Luca Aceto
Monika Henzinger
Jiří Sgall (Eds.)

LNCS 6755
ARCoSS

Automata, Languages and Programming

38th International Colloquium, ICALP 2011
Zurich, Switzerland, July 2011
Proceedings, Part I

1 Part I

Springer

Lecture Notes in Computer Science 6755

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

Advanced Research in Computing and Software Science
Subline of Lectures Notes in Computer Science

Luca Aceto Monika Henzinger
Jiří Sgall (Eds.)

Automata, Languages and Programming

38th International Colloquium, ICALP 2011
Zurich, Switzerland, July 4-8, 2011
Proceedings, Part I

Springer

Volume Editors

Luca Aceto
Reykjavik University, School of Computer Science
101 Reykjavík, Iceland
E-mail: luca@ru.is

Monika Henzinger
Universität Wien, Fakultät für Informatik
Universitätsstraße10/9, 1090 Wien, Österreich
E-mail: monika.henzinger@univie.ac.at

Jiří Sgall
Charles University, Department of Applied Mathematics
Malostranské nám. 25, 118 00 Praha 1, Czech Republic
E-mail: sgall@kam.mff.cuni.cz

ISSN 0302-9743 e-ISSN 1611-3349
ISBN 978-3-642-22005-0 e-ISBN 978-3-642-22006-7
DOI 10.1007/978-3-642-22006-7
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2011930001

CR Subject Classification (1998): F.2, F.1, C.2, H.3, G.2, I.2

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

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

Printed on acid-free paper

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

Preface

ICALP 2011, the 38th edition of the International Colloquium on Automata, Languages and Programming, was held in Zürich, Switzerland, during July 4–8, 2011. ICALP is a series of annual conferences of the European Association for Theoretical Computer Science (EATCS) which first took place in 1972. This year, the ICALP program consisted of three tracks: the established Track A (focusing on Algorithms, Complexity and Games) and Track B (focusing on Logic, Semantics, Automata and Theory of Programming), and a Track C focusing on Foundations of Networked Computation: Models, Algorithms and Information Management.

In response to the call for papers, the Program Committee received 398 submissions: 243 for Track A (three of which were later withdrawn), 103 for Track B and 52 for Track C. Out of these, 114 papers were selected for inclusion in the scientific program: 68 papers for Track A, 29 for Track B, and 17 for Track C. The selection was made by the Program Committees based on originality, quality, and relevance to theoretical computer science. The quality of the manuscripts was very high indeed, and many deserving papers could not be selected.

The EATCS sponsored both a best paper and a best student paper (all authors are students) award for each of the three tracks, to be selected by the Program Committees. The best paper awards were given to Malte Beecken, Johannes Mittmann, and Nitin Saxena for their paper "Algebraic Independence and Blackbox Identity Testing" (Track A), to Olivier Carton, Thomas Colcombet, and Gabriele Puppis for their paper "Regular Languages of Words Over Countable Linear Orderings" (Track B), and to Martin Hoefer for his paper "Local Matching Dynamics in Social Networks" (Track C). The best student paper awards were given to Shi Li for his paper "A 1.488-Approximation Algorithm for the Uncapacitated Facility Location Problem" (Track A), to Martin Delacourt for his paper "Rice's Theorem for mu-Limit Sets of Cellular Automata" (Track B), and to Shiri Chechik for her paper "Fault-Tolerant Compact Routing Schemes for General Graphs" (Track C).

ICALP 2011 consisted of five invited lectures and the contributed papers. This volume of the proceedings contains all contributed papers from Track A, except for the two papers that received one of the best paper awards. These two papers are contained in a companion volume, together with all contributed papers from Track B and Track C, and the papers by four of the invited speakers: Rajeev Alur (University of Pennsylvania, USA), Thore Husfeldt (IT University of Copenhagen, Denmark), Catuscia Palamidessi (INRIA Saclay and LIX, France), and Ronen Shaltiel (University of Haifa, Israel). The program had an additional invited lecture by Éva Tardos (Cornell University, USA), which does not appear in the proceedings.

The following workshops were held as satellite events of ICALP 2011:
GA - Graph algorithms and Applications
GT - Group Testing
DCM - 7th International Workshop on Developments of Computational Models
SDKB - 5th International Workshop on Semantics in Data and Knowledge Bases

We wish to thank all the authors who submitted extended abstracts for consideration, the Program Committees for their scholarly effort, and all referees who assisted the Program Committees in the evaluation process.

Thanks to the sponsors (Swiss National Science Foundation and Google) for their support, and to ETH Zürich for hosting ICALP 2011. We are also grateful to all members of the Organizing Committee and to their support staff in the Institute of Theoretical Computer Science at ETH Zürich. The conference-management system EasyChair was used in handling the submissions and the electronic Program Committee meeting, as well as in assisting in the assembly of the proceedings.

May 2011

Luca Aceto
Monika Henzinger
Jiří Sgall

Organization

Program Committee

Track A

Nikhil Bansal	IBM Research, USA
Harry Buhrman	University of Amsterdam, The Netherlands
Marek Chrobak	University of California, Riverside, USA
Martin Dietzfelbinger	Ilmenau University of Technology, Germany
Thomas Erlebach	University of Leicester, UK
Fedor Fomin	University of Bergen, Norway
Dimitris Fotakis	National Technical University of Athens, Greece
Ricard Gavaldà	UPC Barcelona, Spain
Russell Impagliazzo	University of California, San Diego, USA
Juhani Karhumäki	University of Turku, Finland
Howard Karloff	AT&T Labs–Research, USA
Michal Koucký	Academy of Sciences of the Czech Republic, Czech Republic
Dariusz Kowalski	University of Liverpool, UK
Stefano Leonardi	Sapienza University of Rome, Italy
Gonzalo Navarro	University of Chile, Chile
Rolf Niedermeier	Technische Universität Berlin, Germany
Rafail Ostrovsky	University of California, Los Angeles, USA
Günter Rote	Freie Universität Berlin, Germany
Christian Scheideler	University of Paderborn, Germany
Maria Serna	UPC Barcelona, Spain
Jiří Sgall	Charles University in Prague (Chair), Czech Republic
Gábor Tardos	Simon Fraser University, Canada
Jan Vondrák	IBM Research, USA
Uli Wagner	ETH Zürich, Switzerland
Prudence Wong	University of Liverpool, UK

Track B

Luca Aceto	Reykjavik University (Chair), Iceland
Anuj Dawar	University of Cambridge, UK
Rocco De Nicola	University of Florence, Italy
Zoltán Ésik	University of Szeged, Hungary
Wan Fokkink	VU University Amsterdam, The Netherlands
Herman Geuvers	Radboud University Nijmegen, The Netherlands
Radha Jagadeesan	DePaul University, USA

Jarkko Kari	University of Turku, Finland
Joost-Pieter Katoen	RWTH Aachen, Germany
Orna Kupferman	Hebrew University, Israel
Francois Laroussinie	LIAFA, University Paris Diderot, France
Carroll Morgan	University of New South Wales, Australia
Anca Muscholl	LaBRI, University of Bordeaux, France
Hanne Riis Nielson	Technical University of Denmark, Denmark
Prakash Panangaden	McGill University, Canada
Joachim Parrow	Uppsala University, Sweden
Reinhard Pichler	Vienna University of Technology, Austria
Roberto Segala	University of Verona, Italy
Helmut Seidl	Technische Universität München, Germany
Alex Simpson	University of Edinburgh, UK
Pawel Urzyczyn	University of Warsaw, Poland

Track C

Gilles Barthe	IMDEA Software, Spain
András Benczúr	Hungarian Academy of Sciences, Hungary
Edith Cohen	AT&T Labs–Research, USA
Joan Feigenbaum	Yale University, USA
Amos Fiat	Tel-Aviv University, Israel
Lisa Fleischer	Dartmouth College, USA
Georg Gottlob	Oxford University, UK
Monika Henzinger	University of Vienna (Chair), Austria
Bruce Maggs	Carnegie Mellon and Duke University, USA
Massimo Merro	University of Verona, Italy
Vahab Mirrokni	Google Inc., USA
Alessandro Panconesi	Sapienza University of Rome, Italy
Giuseppe Persiano	University of Salerno, Italy
Anna Philippou	University of Cyprus, Cyprus
Davide Sangiorgi	University of Bologna, Italy
Vladimiro Sassone	University of Southampton, UK
Andrew Tomkins	Google Inc., USA
Dorothea Wagner	Karlsruhe Institute of Technology, Germany
Roger Wattenhofer	ETH Zürich, Switzerland
Ingmar Weber	Yahoo! Research Barcelona, Spain

Conference Chairs

Michael Hoffmann	ETH Zürich
Juraj Hromkovič	ETH Zürich
Ueli Maurer	ETH Zürich
Angelika Steger	ETH Zürich
Emo Welzl	ETH Zürich
Peter Widmayer	ETH Zürich

Referees

Farid Ablayev
Lucia Acciai
Pankaj Agarwal
Jae Hyun Ahn
Saeed Alaei
Susanne Albers
Boris Alexeev
Eric Allender
Shaull Almagor
Andris Ambainis
Aris Anagnostopoulos
Alexandr Andoni
Andrea Frosini
Stefan Andrei
Aaron Archer
Argimiro Arratia
Eugene Asarin
James Aspnes
Mohamed Faouzi Atig
Albert Atserias
Jaume Baixeries
Borja Balle
Vince Bárány
Jérémy Barbay
Pablo Barceló
David Mix Barrington
Libor Barto
Mohammadhossein Bateni
Tugkan Batu
Reinhard Bauer
Mohsen Bayati
Paul Beame
Martin Beaudry
Luca Becchetti
Nicolas Bedon
Piotr Berman
Marco Bernardo
Sandford Bessler
Nadja Betzler
René van Bevern
Umang Bhaskar
Binay Bhattacharya
Marcin Bienkowski
Henrik Björklund
Markus Bläser
Ed Blakey
Jens Blanck
Avrim Blum
Hans-Joachim Böckenhauer
Andrej Bogdanov
Paolo Boldi
Francesco Bonchi
Marcello Bonsangue
Johannes Borgström
Glencora Borradaile
Patricia Bouyer
Julian Bradfield
Vasco Brattka
Vladimir Braverman
Robert Bredereck
Davide Bresolin
Franck van Breugel
Patrick Briest
Thomas Brihaye
Gerth Stølting Brodal
Václav Brožek
Janusz Brzozowski
Andrei Bulatov
Sebastian Burckhardt
Sergiu Bursuc
Jaroslaw Byrka
Sergio Cabello
Jin-Yi Cai
Gruia Calinescu
Cristian S. Calude
Philippe Camacho
Silvio Capobianco
Alberto Caprara
Venanzio Capretta
Ioannis Caragiannis
Arnaud Carayol
Olivier Carton
Sourav Chakraborty
Ho-Leung Chan
Joseph Chan
Witold Charatonik
Krishnendu Chatterjee
Arkadev Chattopadhyay
Prasad Chebolu
Shiri Chechik
Panagiotis Cheilaris
Jianer Chen
Jiehua Chen
Joe Cheriyan
Sherman S.M. Chow
Tobias Christ
Giorgios Christodoulou
Richard Cleve
Thomas Colcombet
Florin Constantin
Graham Cormode
José Correa
László Csirmaz
Marek Cygan
Artur Czumaj
Victor Dalmau
Peter Damaschke
Philippe Darondeau
Sanjeeb Dash
Samir Datta
Tugrul Dayar
Brian Dean
Aldric Degorre
Alberto Del Pia
Stephanie Delaune
Holger Dell
Giorgio Delzanno
Yuxin Deng
Josee Desharnais
Mariangiola Dezani
Ilias Diakonikolas
Volker Diekert
Michael Dinitz
Laurent Doyen
Feodor Dragan
Martin Dyer
Stefan Dziembowski

Josep Díaz
György Dósa
Jeff Egger
Pavlos Eirinakis
Christian Eisentraut
Tinaz Ekim
Khaled Elbassioni
Michael Elkin
Yuval Emek
Alina Ene
David Eppstein
Leah Epstein
Guy Even
Rolf Fagerberg
Jean Fanchon
Arash Farzan
Tomás Feder
Ingo Feinerer
Michael Fellows
Henning Fernau
Diodato Ferraioli
Esteban Feuerstein
Piotr Filipiuk
Irene Finocchi
Dario Fiore
Luca Fossati
Pierre Fraignaud
Gianmarco
De Francisci Morales
Rusins Freivalds
Tom Friedetzky
Ehud Friedgut
Hongfei Fu
Takuro Fukunaga
Stanley Fung
Joaquim Gabarro
Murdoch Gabbay
Bernd Gärtner
Martin Gairing
Anna Găl
Nicola Galesi
Pierre Ganty
Leszek Gasieniec
Serge Gaspers
Qi Ge
Heidi Gebauer
Ran Gelles
Blaise Genest
Chryssis Georgiou
George Giakkoupis
Panos Giannopoulos
Hugo Gimbert
Aristides Gionis
Vasilis Gkatzelis
Rob van Glabbeek
Andreas-Nikolas Gobel
Andreas Goerdt
Leslie Ann Goldberg
Paul Goldberg
Oded Goldreich
Mordecai J. Golin
Petr Golovach
Renato Gomes
Gosta Grahne
Fabrizio Grandoni
Vince Grolmusz
Jan Friso Groote
Romain Grunert
Erich Grädel
Sudipto Guha
Oktay Gunluk
Jiong Guo
Dan Gutfreund
Gregory Gutin
Robert Görke
Annegret Habel
Serge Haddad
Mohammadtaghi
Hajiaghayi
Magnús M. Halldórsson
Sean Hallgren
Nir Halman
Sardouna Hamadou
Xin Han
Chris Hankin
Kristoffer
Arnsfelt Hansen
Nicolas Hanusse
Sariel Har-Peled
Tero Harju
Tobias Harks
Sepp Hartung
Ichiro Hasuo
Pinar Heggernes
Brett Hemenway
Matthew Hennessy
Alejandro Hernandez
Timon Hertli
Jane Hillston
Edward A. Hirsch
Mika Hirvensalo
John Hitchcock
Petr Hliněný
Martin Hoefer
Frank Hoffmann
Thomas Holenstein
Lukas Holik
Wing Kai Hon
Iiro Honkala
Hendrik Jan Hoogeboom
Juraj Hromkovič
Pierre Hyvernat
Falk Hüffner
Martina Hüllmann
Nicole Immorlica
Yuval Ishai
Giuseppe F. Italiano
Szabolcs Iván
Kazuo Iwama
Andreas Jakoby
David N. Jansen
Klaus Jansen
Emmanuel Jeandel
Mark Jerrum
Ranjit Jhala
Albert Xin Jiang
David Johnson
Peter Jonsson
Hossein Jowhari
David Juedes
Joanna Jędrzejowicz
Valentine Kabanets
Mark Kambites
Marcin Kamiński
Ohad Kammar

Frank Kammer
Mong-Jen Kao
Alexis Kaporis
Michael Kapralov
George Karakostas
David Karger
Lila Kari
Juha Karkkainen
Dmitriy Katz
Telikepalli Kavitha
Ken-Ichi Kawarabayashi
Bart de Keijzer
Alexander Kesselman
Andrew King
Daniel Kirsten
Tamás Király
Hartmut Klauck
Philip Klein
Marek Klonowski
Christian Knauer
Sebastian Kniesburges
Naoki Kobayashi
Johannes Köbler
Jochen Könemann
Boris Köpf
Pascal Koiran
Stavros Kolliopoulos
Petr Kolman
Christian Komusiewicz
Stavros Konstantinidis
Spyros Kontogiannis
Eryk Kopczynski
Guy Kortsarz
Nitish Korula
Adrian Kosowski
Paraschos Koutris
Andreas Koutsopoulos
Łukasz Kowalik
Miroslaw Kowaluk
Marcin Kozik
Jan Krajíček
Dieter Kratsch
Robbert Krebbers
Klaus Kriegel
Ravishankar Krishnaswamy
Andrei Krokhin
Marcus Krug
Piotr Krysta
Manfred Kudlek
Fabian Kuhn
Oliver Kullmann
Ravi Kumar
Gábor Kun
Alexander Kurz
Martin Kutrib
Tomi Kärki
Juha Kärkkäinen
Oded Lachish
Pascal Lafourcade
Tak-Wah Lam
Michael Lampis
Lawrence Larmore
Kasper Green Larsen
Sławomir Lasota
Diego Latella
Silvio Lattanzi
Lap Chi Lau
Luigi Laura
Lap-Kei Lee
Troy Lee
Erik Jan van Leeuwen
Pietro Di Lena
Jérôme Leroux
Adam Letchford
Peter Leupold
Jian Li
Ugo de'Liguoro
Andrzej Lingas
Ben Lippmeier
Jiamou Liu
Ivana Ljubic
Martin Loebl
Bruno Loff
Markus Lohrey
Daniel Lokshtanov
Sylvain Lombardy
Violetta Lonati
John Longley
Michele Loreti
Antoni Lozano
Alessandro De Luca
Edward Lui
Christof Löding
Benedikt Löwe
Christoph Lüth
Klaus Madlener
Aleksander Madry
Mohammad Mahdian
Hemanta Maji
Konstantin Makarychev
Yury Makarychev
David F. Manlove
Rajsekar Manokaran
Giulio Manzonetto
Martin Mareš
Nicolas Markey
Bruno Marnette
Barnaby Martin
Russell Martin
Conrado Martinez
Dániel Marx
Mieke Massink
Monaldo Mastrolilli
Claire Mathieu
Arie Matsliah
Elvira Mayordomo
Andrew McGregor
Annabelle McIver
Pierre McKenzie
Larissa Meinicke
Daniel Meister
Páll Melsted
Wolfgang Merkle
George B. Mertzios
Stephan Merz
Julián Mestre
Roland Meyer
Tillmann Miltzow
Matteo Mio
Neeldhara Misra
Michael Mitzenmacher

Xavier Molinero
Ashley Montanaro
Georg Moser
Elchanan Mossel
Achour Mostefaoui
David Mount
Haiko Müller
Wolfgang Mulzer
Marcelo Mydlarz
Sebastian A Mödersheim
Rasmus Ejlers Møgelberg
Viswanath Nagarajan
Mark-Jan Nederhof
Alantha Newman
Ilan Newman
Cyril Nicaud
André Nichterlein
Dejan Nickovic
Flemming Nielson
Damian Niwinski
Martin Nöllenburg
Thomas Noll
Gethin Norman
Dirk Nowotka
Zeev Nutov
Jan Obdržálek
Alexander Okhotin
Sergi Oliva
Alexander Olshevsky
Krzysztof Onak
Adrian Onet
Luke Ong
Rotem Oshman
Friedrich Otto
Shayan Oveis Gharan
Scott Owens
Rasmus Pagh
Thomas Pajor
Katarzyna Paluch
Konstantinos Panagiotou
Debmalya Panigrahi
Charis Papadopoulos
Vicky Papadopoulou
Srinivasan Parthasarathy
Pawel Parys
Francesco Pasquale
Balázs Patkós
Sriram Pemmaraju
Holger Petersen
Ion Petre
Cynthia Phillips
Claudine Picaronny
George Pierrakos
Giovanni Pighizzini
Marcin Pilipczuk
Adolfo Piperno
Giuseppe Pirillo
Nir Piterman
Wojciech Plandowski
Jaco van de Pol
C.K. Poon
Alexandru Popa
Viorel Preoteasa
Christian W. Probst
Kirk Pruhs
Rosario Pugliese
Dömötör Pálvölgyi
Karin Quaas
Jose Quaresma
Femke van Raamsdonk
Alexander Rabinovich
Yuri Rabinovich
Luis Rademacher
Tomasz Radzik
Vijaya Ramachandran
Rajeev Raman
Venkatesh Raman
Narad Rampersad
Julian Rathke
Dror Rawitz
Ran Raz
Igor Razgon
Oded Regev
Klaus Reinhardt
Tzachy Reinman
Michel Reniers
Pierre-Alain Reynier
Mark Reynolds
Michael Rink
Liam Roditty
Heiko Röglin
Dana Ron
Mads Rosendahl
Alexander Russell
Ignaz Rutter
Andrey Rybalchenko
Wojciech Rytter
Harald Räcke
Stefan Rümmele
Joshua Sack
Mert Saglam
Jacques Sakarovitch
Mohammad Salavatipour
Kai Salomaa
Alex Samorodnitsky
Peter Sanders
Miklos Santha
Rahul Santhanam
Rik Sarkar
Saket Saurabh
Vadim Savenkov
Nitin Saxena
Chrstian Schaffner
Dominik Scheder
Marc Scherfenberg
Saul Schleimer
Lena Schlipf
Philippe Schnoebelen
Aleksy Schubert
Warren Schudy
Daria Schymura
Géraud Sénizergues
Marco Serafini
Olivier Serre
Rocco Servedio
C. Seshadhri
Mordechai Shalom
Arpit Sharma
Sarai Sheinvald
Akiyoshi Shioura
David Shmoys
Igor Shparlinski
Amir Shpilka
Anastasios Sidiropoulos

Mihaela Sighireanu
Pedro V. Silva
Riccardo Silvestri
Hans Ulrich Simon
Sebastian Skritek
Nataliya Skrypnyuk
Martin Skutella
Michael Smith
Christian Sohler
Ana Sokolova
Manuel Sorge
Paul Spirakis
Jeremy Sproston
Jiří Srba
Kannan Srinathan
Juraj Stacho
Julinda Stefa
Daniel Štefankovič
Fabian Stehn
Colin Stirling
Alejandro Strejilevich de Loma
David Steurer
Kristian Støvring
Ondřej Suchý
Ola Svensson
Maxim Sviridenko
Chaitanya Swamy
Vasilis Syrgkanis
Stefan Szeider
Nicolas Tabareau
Kunal Talwar
Till Tantau
Éva Tardos
Nina Taslaman
Orestis Telelis
Balder Ten Cate
Lidia Tendera
Michał Terepeta
Pascal Tesson
Nithum Thain
Dimitrios Thilikos
Mikkel Thorup
Francesco Tiezzi
Hingfung Ting
Ashish Tiwari
Szymon Toruńczyk
Mirco Tribastone
Stavros Tripakis
Rahul Tripathi
Enrico Tronci
Madhur Tulsiani
Christos Tzamos
Nikos Tzevelekos
Géza Tóth
Ryuhei Uehara
Marc Uetz
Johannes Uhlmann
Irek Ulidowski
Ugo Vaccaro
Sándor Vágvölgyi
Vasco T. Vasconcelos
Andrea Vattani
Roope Vehkalahti
Helmut Veith
Santosh S. Vempala
Betti Venneri
Carmine Ventre
Elad Verbin
Oleg Verbitsky
Nikolay Vereshchagin
Ivan Visconti
Mahesh Viswanathan
Berthold Vöcking
Markus Völker
Walter Vogler
Björn Wachter
Lei Wang
John Watrous
Mathias Weller
Daniel Werner
Matthias Westermann
Andreas Wiese
Thomas Wilke
Gerhard J. Woeginger
Ronald de Wolf
Paul Wollan
Frank Wolter
David Woodruff
James Worrell
Yi Wu
Dachuan Xu
Eran Yahav
Li Yan
Qiqi Yan
Mu Yang
Ke Yi
Neal Young
Raphael Yuster
Morteza Zadimoghaddam
Michael Zakharyaschev
Hans Zantema
Christos Zaroliagis
Konrad Zdanowski
Marc Zeitoun
Alex Zelikovsky
Rico Zenklusen
Fuyuan Zhang
Lijun Zhang
Min Zhang
Yong Zhang
Chunlai Zhou
Stanislav Živný
Florian Zuleger

Table of Contents – Part I

Session A1: Network Design Problems

Session A2: Quantum Computing

Session A3: Graph Algorithms

Session A4: Games, Approximation Schemes, Smoothed Analysis

Session A5: Online Algorithms

Session A6: Data Structures, Distributed Computing

Session A7: Complexity, Randomness

Session A8: Submodular Optimization, Matroids

Session A9: Cryptography, Learning

Session A10: Fixed Parameter Tractability

Session A11: Hardness of Approximation

Session A12: Counting, Testing

Session A13: Complexity

Session A14: Proof Complexity

Session A15: Sorting, Matchings, Paths

Session A16: Constraint Satisfaction, Algebraic Complexity

Session A17: Steiner Problems, Clustering

Table of Contents – Part II

Invited Lectures

Best Student Papers

Best Papers

Session B1: Foundations of Program Semantics

Session B2: Automata and Formal Languages

Session B3: Model Checking

Session B4: Probabilistic Systems

Session B5: Logic in Computer Science

Session B6: Hybrid Systems

Session B7: Specification and Verification

Session C1: Graphs

Session C2: Matchings and Equilibria

Session C3: Privacy and Content Search

Session C4: Distributed Computation

Improved Approximation for the Directed Spanner Problem

Piotr Berman[1], Arnab Bhattacharyya[2,⋆], Konstantin Makarychev[3], Sofya Raskhodnikova[1,⋆⋆], and Grigory Yaroslavtsev[1,⋆⋆⋆]

[1] Pennsylvania State University
{berman,sofya,grigory}@cse.psu.edu
[2] Massachusetts Institute of Technology
abhatt@mit.edu
[3] IBM T.J. Watson Research Center
konstantin@us.ibm.com

Abstract. We give an $O(\sqrt{n}\log n)$-approximation algorithm for the problem of finding the sparsest spanner of a given *directed* graph G on n vertices. A spanner of a graph is a sparse subgraph that approximately preserves distances in the original graph. More precisely, given a graph $G = (V, E)$ with nonnegative edge lengths $d : E \to \mathbb{R}^{\geq 0}$ and a *stretch* $k \geq 1$, a subgraph $H = (V, E_H)$ is a *k-spanner* of G if for every edge $(u, v) \in E$, the graph H contains a path from u to v of length at most $k \cdot d(u, v)$. The previous best approximation ratio was $\tilde{O}(n^{2/3})$, due to Dinitz and Krauthgamer (STOC '11).

We also present an improved algorithm for the important special case of directed 3-spanners with unit edge lengths. The approximation ratio of our algorithm is $\tilde{O}(n^{1/3})$ which almost matches the lower bound shown by Dinitz and Krauthgamer for the integrality gap of a natural linear programming relaxation. The best previously known algorithms for this problem, due to Berman, Raskhodnikova and Ruan (FSTTCS '10) and Dinitz and Krauthgamer, had approximation ratio $\tilde{O}(\sqrt{n})$.

1 Introduction

A spanner of a graph is a sparse subgraph that approximately preserves pairwise distances in the original graph. This notion was first used by Awerbuch [2] and explicitly introduced by Peleg and Schäffer [23].

Definition 1.1 (k-spanner, [2,23]). *Given a graph $G = (V, E)$ with nonnegative edge lengths $d : E \to \mathbb{R}^{\geq 0}$ and a real number $k \geq 1$, a subgraph $H = (V, E_H)$ is a **k-spanner** of G if for all edges $(u, v) \in E$, the graph H contains a path from u to v of length at most $k \cdot d(u, v)$. The parameter k is called the **stretch**.*

⋆ Supported by National Science Foundation grants CCF-1065125 and CCF-0728645.
⋆⋆ Supported by National Science Foundation (NSF/CCF CAREER award 0845701).
⋆⋆⋆ Supported by NSF / CCF CAREER award 0845701, University Graduate Fellowship and College of Engineering Fellowship.

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 1–12, 2011.

Spanners have numerous applications, such as efficient routing [9,10,25,27,28], simulating synchronized protocols in unsynchronized networks [24], parallel, distributed and streaming algorithms for approximating shortest paths [7,8,13,18], algorithms for distance oracles [3,29], property testing, property reconstruction and key management in access control hierarchies (see [6,5,20], the survey in [26] and references therein).

We study the computational problem of finding the sparsest spanner of a given *directed* graph G and stretch k, that is, a k-spanner of G with the smallest number of edges. We refer to this problem as DIRECTED k-SPANNER and distinguish between the case of unit edge lengths (i.e., $d(e) = 1$ for all $e \in E$) and arbitrary edge lengths. The UNDIRECTED k-SPANNER problem refers to the task of finding the sparsest k-spanner of a given undirected graph. The natural reduction from UNDIRECTED k-SPANNER to DIRECTED k-SPANNER preserves approximation ratio.

Our main results are an algorithm with approximation ratio $O(\sqrt{n}\log n)$ for DIRECTED k-SPANNER with arbitrary edge lengths and an algorithm with approximation ratio $O(n^{1/3}\log^2 n)$ for DIRECTED 3-SPANNER with unit edge lengths, where n is the number of nodes in the input graph G. Our approximation guarantee for DIRECTED 3-SPANNER almost matches the integrality gap of $\Omega(n^{1/3-\epsilon})$ of Dinitz and Krauthgamer [11] for a natural linear programming relaxation of the problem. Our result also directly implies the same approximation ratio for the UNDIRECTED 3-SPANNER problem with unit edge lengths.

Relation to Previous Work. DIRECTED k-SPANNER with *unit edge lengths* has been extensively studied. Note that in this case, we can assume that k is a positive integer. For $k = 2$, the problem has been completely resolved: Kortsarz and Peleg [21] and Elkin and Peleg [15] gave $O(\log n)$-approximation, and Kortsarz [22] proved that the approximation cannot be improved unless P=NP. Elkin and Peleg [14] gave $\tilde{O}(n^{2/3})$-approximation for DIRECTED 3-SPANNER. For general $k \geq 3$, Bhattacharyya *et al.* [6] presented $\tilde{O}(n^{1-1/k})$-approximation; then, Berman, Raskhodnikova and Ruan [4] improved it to $\tilde{O}(n^{1-1/\lceil k/2\rceil})$, and recently Dinitz and Krauthgamer [11] gave $\tilde{O}(n^{2/3})$-approximation, presenting the first algorithm with approximation ratio independent of k. For the special cases of $k = 3$ and $k = 4$, Berman, Raskhodnikova and Ruan showed an $\tilde{O}(\sqrt{n})$-approximation. Independently, Dinitz and Krauthgamer also gave an $\tilde{O}(\sqrt{n})$-approximation for the case $k = 3$. Thus, our algorithms improve on [4] for all $k \geq 3$, where $k \neq 4$, and on [11] for all $k \geq 3$.

Dinitz and Krauthgamer gave the first approximation algorithm for the problem for arbitrary edge lengths. For this case, one can no longer assume that k is an integer. Dinitz and Krauthgamer achieved $\tilde{O}(n^{2/3})$-approximation for arbitrary edge lengths for all $k > 1$. We improve this approximation to $\tilde{O}(n^{1/2})$ for all $k > 1$.

Spanners for undirected graphs behave somewhat differently in terms of their approximability. For all integer k and for all undirected graphs G with arbitrary edge lengths, it is known [23,1] that a k-spanner of G with at most $n \cdot \lceil n^{2/(k+1)}\rceil$ edges can be constructed in polynomial time. Since a k-spanner of a connected

graph must have at least $n-1$ edges, an approximation ratio of $O(n^{2/(k+1)})$ trivially follows. In particular, for $k=3$, this argument yields an approximation ratio of $O(\sqrt{n})$. Our result improves the ratio for Undirected 3-Spanner to $\tilde{O}(n^{1/3})$ in the case of unit-length edges.

Elkin and Peleg [14,17], improving on [22], showed that it is quasi-NP-hard to approximate Directed k-Spanner, even when restricted to unit edge lengths, with ratio better than $2^{\log^{1-\epsilon} n}$ for $k \in (3, n^{1-\delta})$ and all $\delta, \epsilon \in (0,1)$. For Undirected k-Spanner with unit-length edges, such a strong hardness result does not hold since the problem is $O(1)$-approximable when $k = \Omega(\log n)$. However, for constant $k \geq 3$, it is still quasi-NP-hard to approximate with a ratio better than $2^{\log^{1-\epsilon} n}$ [14,12]. When the edge lengths are arbitrary, the same inapproximability also holds for $k \in (1,3)$, even for the undirected case [17].

Our Techniques. Our algorithms operate by combining two graphs: the first obtained from randomized rounding of a fractional solution to a flow-based linear programming relaxation of the problem and the second obtained by growing shortest-path trees from randomly selected vertices. The idea of combining a linear programming approach with sampling to solve Directed k-Spanner first appeared in [6]. Dinitz and Krauthgamer [11] used the same approach, but with a novel, flow-based linear program (LP). Our main insight is to use randomized LP rounding schemes. We also give a new LP relaxation, slightly simpler than that in [11]. In the case of unit edge lengths, this LP has an extra advantage: it can be solved quickly without using the ellipsoid algorithm. We note, however, that our method would yield the same approximation ratios with the LP of Dinitz and Krauthgamer [11] as well.

Directed Steiner Forest. Consider the Directed Steiner Forest (DSF) problem, a basic network design problem for directed graphs: given a directed graph $G = (V, E)$ with edge costs and a collection $D \subseteq V \times V$ of vertex pairs, find a minimum-cost subgraph of G that contains a path from u to v for every pair $(u, v) \in D$. DSF is an NP-hard problem and is known [12] to be quasi-NP-hard to approximate with ratio better than $2^{\log^{1-\epsilon} n}$ for all $\epsilon \in (0,1)$. The best known approximation ratio for this problem is $O(n^{\epsilon} \cdot \min(n^{4/5}, m^{2/3}))$, due to Feldman, Kortsarz and Nutov [19]. Their algorithm has the same structure as the algorithms for Directed k-Spanner in [6] and [11]. Specifically, the LP relaxation that they formulate is closely related to that developed by Dinitz and Krauthgamer, if we replace edge costs by edge lengths. Our technique for the spanner problem also applies to the DSF problem, yielding an improved approximation ratio of $\tilde{O}(n^{2/3+\epsilon})$. We defer details to the full version.

2 An $\tilde{O}(\sqrt{n})$-Approximation for Directed k-Spanner

Our main result is stated in the following theorem.

Theorem 2.1. *There is a polynomial time randomized algorithm for* Directed k-Spanner *with expected approximation ratio* $O(\sqrt{n} \log n)$.

We present two algorithms to prove Theorem 2.1: an algorithm for the general case (whose description is completed in Section 2.2) and a simpler and more efficient algorithm for the special case when all edges have unit length (whose description is completed in Section 3).

Let $G = (V, E)$ be a directed graph with edge lengths $d : E \to \mathbb{R}^{\geq 0}$, given as input to our algorithm, and OPT be the size of its sparsest k-spanner. We assume that G is weakly connected. Otherwise, our algorithm should be executed for each weakly connected component separately.

Definition 2.1. *For an edge $(s,t) \in E$, let $G^{s,t} = (V^{s,t}, E^{s,t})$ be the subgraph of G induced by the vertices on paths from s to t of length at most $k \cdot d(s,t)$.*

Definition 2.2 (Thick and thin edges). *Let β be a parameter in $[1, n]$. If $|V^{s,t}| \geq n/\beta$, the corresponding edge (s,t) is* thick*, and otherwise, it is* thin*. The set of all thin edges is denoted by $\mathcal{E}$. In Sections 2.1–3, we shall always assume that $\beta = \sqrt{n}$.*

Our general strategy is to solve the problem separately for thick and thin edges. We find two sets of edges, E' and E'', such that for each edge $(s,t) \in E$, the required path of length at most $k \cdot d(s,t)$ from s to t is contained in E' if (s,t) is thick and in E'' if (s,t) is thin. The expected size of both sets is $O(\beta \log n \cdot OPT)$.

In Section 2.1, we describe how to obtain E' using random sampling. In Section 2.2, we describe how to obtain E'' in the general case, using randomized rounding of a fractional solution to an LP, thus completing the proof of Theorem 2.1. For graphs with unit edge lengths, the general method is the same, but we use a different LP (see Section 3).

2.1 Sampling

We say that an edge $(s,t) \in E$ is *settled* if the k-spanner property for this edge is satisfied, i.e., the selected set of edges contains a path of length at most $k \cdot d(s,t)$ from s to t. The following procedure uses random sampling to construct E'.

Algorithm 1. SAMPLE(β)

1. $E' \leftarrow \varnothing$, $S \leftarrow \varnothing$;
2. **for** $i = 1$ to $\beta \ln n$ **do**
3. $\quad v \leftarrow$ a uniformly random element of V;
4. $\quad T_v^{in} \leftarrow$ a shortest path in-arborescence rooted at v;
5. $\quad T_v^{out} \leftarrow$ a shortest path out-arborescence rooted at v;
6. $\quad E' \leftarrow E' \cup T_v^{in} \cup T_v^{out}$, $S \leftarrow S \cup \{v\}$; //Set S is used only in the analysis.
7. **end for**
8. Add all unsettled thick edges to E';
9. **return** E'.

Lemma 2.1. *Algorithm 1, in polynomial time, computes a set E' that settles all thick edges and has expected size at most $3\beta \ln n \cdot OPT$.*

Proof. After the execution of the **for**-loop in Algorithm 1, $|E'| \leq 2(n-1)\beta \ln n \leq 2\beta \ln n \cdot OPT$. The last inequality holds because $OPT \geq n-1$ for weakly connected graphs G.

If some vertex v from a set $V^{s,t}$ appears in the set S of vertices selected by Sample, then T_v^{in} and T_v^{out} contain shortest paths from s to v and from v to t, respectively. Thus, both paths are contained in E'. Since $v \in V^{s,t}$, the sum of lengths of these two paths is at most $k \cdot d(s,t)$. Therefore, if $S \cap V^{s,t} \neq \varnothing$, then the edge (s,t) is settled. For a thick edge (s,t), the set $S \cap V^{s,t}$ is empty with probability at most $(1-1/\beta)^{\beta \ln n} \leq e^{-\ln n} = 1/n$. Thus, the expected number of unsettled thick edges added to E' in Step 8 of Sample is at most $|E|/n \leq n-1 \leq OPT$.

Step 8 ensures that E', returned by the algorithm, settles all thick edges. Computing shortest path in- and out-arborescences and determining whether an edge is thick can be done in polynomial time. □

2.2 Antispanners, LP and the Separation Oracle

In this section, we introduce *antispanners*, a notion used in the description of our algorithm for Directed k-Spanner and essential in the analysis of all algorithms. It is needed in the parts of the algorithms that settle thin edges. Then, we formulate an LP relaxation of the problem of settling thin edges and present our approximation algorithm, proving Theorem 2.1.

Antispanners. For a given edge (s,t), we define an antispanner to be a subset of edges of G, such that if we remove this subset of edges from G, the length of the shortest path from s to t becomes larger than $k \cdot d(s,t)$.

Definition 2.3 (Antispanners). *A set $C \subseteq E$ is an antispanner for an edge $(s,t) \in E$ if $G' = (V, E \setminus C)$ contains no path from s to t of length at most $k \cdot d(s,t)$. If no proper subset of an antispanner C is an antispanner, we say that C is minimal.*

Thus, the edge set of a k-spanner of G must intersect all antispanners for all edges of G. In other words, it has to be a hitting set for all minimal antispanners.

We now prove that if a graph $(V, E' \cup E'')$ is not a k-spanner, then we can efficiently find a thin edge $(s,t) \in E$ and a minimal antispanner C that does not intersect E''.

Lemma 2.2. *There exists a polynomial time algorithm that, given a set of edges $E'' \subset E$ and a thin edge $(s,t) \in E$, outputs a minimal antispanner $C \subset E^{s,t} \setminus E''$ if there is no directed path from s to t of length at most $k \cdot d(s,t)$ in E''.*

Proof. The algorithm first checks if there exists a directed path from s to t of length at most $k \cdot d(s,t)$ in E''. If there is no such path then $E^{s,t} \setminus E''$ is an antispanner. (Note that all paths between s and t of lengths at most $k \cdot d(s,t)$ in G lie in the subgraph $G^{s,t} = (V^{s,t}, E^{s,t})$). The algorithm sets $C = E^{s,t} \setminus E''$ and then sequentially deletes all edges $(u,v) \in C$ such that $C \setminus \{(u,v)\}$ is an antispanner. When no more such edges are left, the algorithm returns C. □

$$\text{Minimize} \sum_{e \in E} x_e \text{ subject to:} \tag{1}$$

$$\sum_{e \in C} x_e \geq 1 \qquad \forall C \in \mathcal{S} \tag{2}$$

$$x_e \geq 0 \qquad \forall e \in E \tag{3}$$

Fig. 1. Linear program for the arbitrary-length case, LP-A

Linear Program. Since E' from Section 2.1 already settles the thick edges, our goal is to design a randomized procedure that finds a subset of edges $E'' \subset E$ that intersects all minimal antispanners for all thin edges. This condition can be expressed using linear program LP-A (see Fig. 1). This LP has a variable x_e for each edge $e \in E$ and a constraint (2) for each minimal antispanner C for thin edges. Set $\mathcal{S}$ is the set of all minimal antispanners for thin edges. In the integral solution $\{x_e^{int}\}$ corresponding to a k-spanner with edge set $E'' \subset E$, $x_e^{int} = 1$ if $e \in E''$ and $x_e^{int} = 0$ otherwise. All constraints (2) are satisfied for $\{x_e^{int}\}$ since E'' intersects every antispanner. The value of the objective function $\sum_e x_e^{int}$ equals the size of E''. Hence, the LP is a valid relaxation.

For ease of presentation, we assume that we have guessed OPT, the size of the optimal spanner. (We can try all values in $\{n-1, \ldots, n^2\}$ for OPT and output the best spanner found in all iterations). We replace the objective function (1) with

$$\sum_{e \in E} x_e \leq OPT. \tag{4}$$

Separation Oracle. Our LP has polynomially many variables and exponentially many constraints. We solve it using the ellipsoid algorithm with a separation oracle. Our separation oracle receives a fractional vector $\{x_e^*\}$ (satisfying (3), (4)) and outputs either a violated constraint (2) for some antispanner C or a set E'' of size at most $2OPT \cdot \sqrt{n} \ln n$ such that $E' \cup E''$ is the edge set of a k-spanner. Specifically, if $\{x_e^*\}$ is a feasible solution, then the separation oracle returns a set E''.

The separation oracle works as follows: it first samples a random set of edges E'' picking each $e \in E$ with probability $\min(x_e^* \sqrt{n} \ln n, 1)$:

Algorithm 2. RandomizedSelection(x_e^*)

1. $E'' \leftarrow \varnothing$;
2. **for** each edge $e \in E$ **do**
3. $\quad p_e \leftarrow \min(1, \sqrt{n} \ln n \cdot x_e^*)$;
4. $\quad$ Add e to E'' with probability p_e;
5. **end for**
6. **return** E''.

Then if $(V, E' \cup E'')$ is a spanner and $|E''| \leq 2OPT \cdot \sqrt{n} \ln n$, it outputs E''. If $|E''| > 2OPT \cdot \sqrt{n} \ln n$, the separation oracle fails. If $(V, E' \cup E'')$ is not a spanner, the algorithm finds a thin edge (s,t) such that there is no directed path of length $k \cdot d(s,t)$ from s to t in (V, E''), then it finds a minimal antispanner $C \subset E^{s,t} \setminus E''$ using a greedy algorithm (see Lemma 2.2 below for details; note that $E^{s,t} \setminus E''$ is an antispanner) and if $\sum_{e \in C} x_e < 1$, outputs this violated constraint. If $\sum_{e \in C} x_e \geq 1$, the separation oracle fails.

We now show that the probability that the separation oracle fails during an execution of the ellipsoid algorithm is small.

Theorem 2.2. *The probability that during an execution of the ellipsoid algorithm the separation oracle fails is exponentially small in n.*

Proof. As discussed above, there are two different events, which can cause the separation oracle to fail:

1. The size of the sampled set E'' is too large. The expected size of E'' is at most $\sqrt{n} \ln n \sum_{e \in E} x_e \leq OPT \cdot \sqrt{n} \ln n$. By the Chernoff bound, $\Pr(|E''| > 2OPT \cdot \sqrt{n} \ln n) \leq e^{-c \cdot OPT \cdot \sqrt{n} \ln n} = e^{-\Omega(n \cdot \sqrt{n} \ln n)}$. Thus, the probability that the separation oracle fails because $|E''| > 2OPT \cdot \sqrt{n} \ln n$ is exponentially small.
2. The minimal antispanner found by the oracle doesn't correspond to a violated constraint (see discussion below). We prove that the probability that the separation oracle fails because $\sum_{e \in C} x^*_e \geq 1$ is exponentially small in Lemma 2.3.

Lemma 2.3. *The probability that there exists an edge (s,t) and a minimal antispanner C for it such that $\sum_{e \in C} x^*_e \geq 1$, but $C \subset E^{s,t} \setminus E''$ is at most $|E| \cdot e^{-\frac{1}{2}\sqrt{n} \ln n}$.*

Proof. First, we bound the total number of minimal antispanners for thin edges.

Proposition 2.1. *If (s,t) is a thin edge, then there are at most $(n/\beta)^{n/\beta}$ minimal antispanners for (s,t). In particular, if $\beta = \sqrt{n}$, then there are at most $\sqrt{n}^{\sqrt{n}}$ minimal antispanners.*

Proof. Fix a thin edge (s,t) and consider an arbitrary minimal antispanner C for (s,t). Let A_C be the outward shortest path tree (arborescence) rooted at s in the graph $(V^{s,t}, E^{s,t} \setminus C)$. Denote by $f_{A_C}(u)$ the distance from s to u in the tree A_C. If there is no directed path from s to u in A_C, we let $f_{A_C}(u) = \infty$. We show that $C = \{(u,v) \in E^{s,t} : f_{A_C}(u) + d(u,v) < f_{A_C}(v)\}$, and, thus A_C uniquely determines C for a given thin edge (s,t). If $(u,v) \in C$, then, since C is a *minimal* antispanner, there exists a path from s to t of length at most $kd(s,t)$ in the graph $(V, E \setminus C \cup \{(u,v)\})$, this path must lie in $(V^{s,t}, E^{s,t} \setminus C \cup \{(u,v)\})$ and must contain the edge (u,v). Thus, the distance from s to t in the graph $(V^{s,t}, E^{s,t} \setminus C \cup \{(u,v)\})$ is at most $k \cdot d(s,t)$ and is strictly less than $f_{A_C}(t)$. Hence, A_C is not the shortest path tree in the graph $(V^{s,t}, E^{s,t} \setminus C \cup \{(u,v)\})$. Therefore, $f_{A_C}(u) + d(u,v) < f_{A_C}(v)$. If $(u,v) \in E^{s,t}$ satisfies the condition

$f_{A_C}(u) + d(u,v) < f_{A_C}(v)$, then $(u,v) \notin E^{s,t} \setminus C$, otherwise A_C would not be the shortest path tree, hence $(u,v) \in C$.

We now count the number of outward trees rooted at s in $(V^{s,t}, E^{s,t} \setminus C)$. For every vertex $u \in V^{s,t}$ we may choose the parent vertex in at most $|V^{s,t}|$ possible ways (if a vertex is isolated we assume that it is its own parent), thus the total number of trees is at most $|V^{s,t}|^{|V^{s,t}|} \leq (n/\beta)^{n/\beta}$. □

Proposition 2.2. *For an edge $(s,t) \in E$ and a minimal antispanner C for (s,t) satisfying $\sum_{e \in C} x^*_e \geq 1$, the probability that $E'' \cap C = \varnothing$ is at most $e^{-\sqrt{n} \ln n}$.*

Proof. Suppose there exists $(u,v) \in C$ such that $x^*_e \geq (\sqrt{n} \ln n)^{-1}$. In this case, $(u,v) \in E''$ with probability 1, and we are already done. Otherwise, for $(u,v) \in C$, the probability that $(u,v) \in E''$ is exactly $\sqrt{n} \ln n \cdot x_e$. The probability that no edges of C are in E'' is, therefore,

$$\prod_{e \in C} (1 - \sqrt{n} \ln n \cdot x^*_e) < \exp\left(- \sum_{e \in C} \sqrt{n} \ln n \cdot x^*_e \right) \leq e^{-\sqrt{n} \ln n}.$$

The first inequality above follows from the fact that $1 - x < \exp(-x)$ for $x > 0$. The second one holds because $\sum_{e \in C} x^*_e \geq 1$. □

The proof of Lemma 2.3 is completed by using Proposition 2.2 and Proposition 2.1 and taking a union bound over all minimal antispanners for all thin edges. □

The proof of Theorem 2.2 is completed by using Lemma 2.3 and taking a union bound over all iterations of the ellipsoid algorithm, the number of which is bounded by a polynomial. □

Proof of Theorem 2.1.

Proof. The thick edges can be settled by running SAMPLE($\sqrt{n}$), according to Lemma 2.1. The thin edges can be settled by running the ellipsoid algorithm as described above. The ellipsoid algorithm terminates in polynomial time. With exponentially small probability, we allow the separation oracle to fail (as shown in Theorem 2.2), in which case we output a spanner containing all edges E. Thus, the expected size of the set E'' is at most $2OPT \cdot \sqrt{n} \ln n + o(1)$ and the resulting approximation ratio of the algorithm is $O(\sqrt{n} \ln n)$.

3 LP and Rounding for Graphs with Unit-Length Edges

In this section, we describe how to settle the thin edges, and thus prove Theorem 2.1, for the case of unit-length edges. Our motivation for presenting this special case is two-fold. First, we show that for the unit-length case, one can directly formulate a polynomial-sized LP relaxation, and this makes the approximation algorithm more efficient. Second, the LP used here will be convenient in presenting the improved approximation for 3-spanners in Section 4.

In order to define and analyze the LP, we need to introduce some notation.

Definition 3.1 (Layered expansion). *Given a directed graph $G = (V, E)$, its* layered expansion *is a directed graph $\hat{G} = (\hat{V}, \hat{E})$, satisfying the following:*

1. *Let $\hat{V} = \{v_i : v \in V \text{ and } i \in \mathbb{Z}^{\geq 0}\}$, where v_i denotes the i-th copy of v. The set of all the i-th copies of nodes in V is the i-th layer of $\hat{V}$.*
2. *Let $L = \{(u, u) : u \in V\}$ be the set of loops. Define the i-th copy of an edge $e = (u, v)$ to be $e_i = (u_i, v_{i+1})$, and the i-th copy of a loop $e = (u, u)$ to be $e_i = (u_i, u_{i+1})$. Let $\hat{E} = \{e_i : e \in E \cup L \text{ and } i \in \mathbb{Z}^{\geq 0}\}$.*

We use layered expansion $\hat{G}$ to describe paths in G. Note that $\hat{G}$ contains a path from u_0 to v_ℓ if and only if G contains a path from u to v of length at most ℓ.

Recall that $\mathcal{E}$ denotes the set of thin edges. For $(s, t) \in \mathcal{E}$, we consider the subgraph of $\hat{G}$ consisting of all paths that can be used by a k-spanner:

Definition 3.2 (Edge network). *For an edge $(s, t) \in \mathcal{E}$ and $k \geq 1$, the* edge network *is a subgraph $\hat{G}_k^{s,t} = (\hat{V}_k^{s,t}, \hat{E}_k^{s,t})$ of $\hat{G}$ with a source $\bar{s} = s_0$ and a sink $\bar{t} = t_{k \cdot d(s,t)}$, such that $\hat{G}_k^{s,t}$ contains all nodes and edges on paths from $\bar{s}$ to $\bar{t}$.*

Now, consider the linear program LP-U defined in Figure 2 below. LP-U has variables of two types: x_e, where $e \in E$, and $f_{e_i}^{s,t}$, where $(s, t) \in \mathcal{E}$ and $e_i \in \hat{E}_k^{s,t}$. A variable x_e represents whether the edge e is included in the k-spanner. A variable $f_{e_i}^{s,t}$ represents flow along the edge e_i in $\hat{G}_k^{s,t}$ (integer flow in $\hat{G}_k^{s,t}$ with value 1 is simply a path of length at most k). We denote the sets of incoming and outgoing edges for a vertex $v_i \in \hat{G}_k^{s,t}$ by $In(v_i)$ and $Out(v_i)$, respectively.

Minimize $\sum_{e \in E} x_e$ subject to:

$$\text{Flow requirement} \quad \sum_{e_0 \in Out(s_0)} f_{e_0}^{s,t} \geq 1 \quad \forall (s,t) \in \mathcal{E}$$

$$\text{Flow conservation} \quad \sum_{e_{i-1} \in In(v_i)} f_{e_{i-1}}^{s,t} - \sum_{e_i \in Out(v_i)} f_{e_i}^{s,t} = 0 \quad \forall (s,t) \in \mathcal{E}, \forall v_i \in \hat{V}_k^{s,t} \setminus \{\bar{s}, \bar{t}\}$$

$$\text{Capacity constraints} \quad x_e - \sum_{i=0}^{k-1} f_{e_i}^{s,t} \geq 0 \quad \forall (s,t) \in \mathcal{E}, \forall e \in E$$

$$x_e \geq 0 \quad \forall e \in E$$

$$f_{e_i}^{s,t} \geq 0 \quad \forall (s,t) \in \mathcal{E}, \forall e_i \in \hat{E}_k^{s,t}$$

Fig. 2. Linear program for the unit-length case, LP-U

Note that to write down LP-U, we only need to know V, E, k and the set of thin edges, $\mathcal{E}$. The first three are inputs to the algorithm, and $\mathcal{E}$ can be computed in polynomial time. LP-U can be written down and solved in polynomial time because it has $O(|E|^2 \cdot k) = O(n^5)$ variables and constraints.[1] Thus, unlike the case of arbitrary lengths, one does not need to invoke the ellipsoid algorithm here.

[1] More precisely, LP-U has $O(|\mathcal{E}| \times |V^{s,t}|^3) = O(n^{3.5})$ variables and constraints.

Given x_e^*, the fractional solution of LP-U, we construct the set E'' by first running Algorithm 2 and then adding all unsettled thin edges. Because sets of fractional solutions x_e^* to LP-U and LP-A are equal, one can show that the set $E' \cup E''$ forms a k-spanner with high probability and the size of this spanner is $O(OPT \cdot \sqrt{n} \log n)$. We give a direct proof of this fact in the full version.

4 An $\tilde{O}(n^{1/3})$-Approximation for Directed 3-Spanner with Unit-Length Edges

In this section, we show an improved approximation for the special case of Directed 3-Spanner with unit length edges. At a high level, our analysis is a combination of the technique we described for Directed k-Spanner with the technique of Dinitz and Krauthgamer [11] for Directed 3-Spanner. The algorithm for Directed 3-Spanner in [11] does not use sampling, which makes their result applicable to the problem on graphs with arbitrary edge *cost*, where the total edge cost is minimized rather than the total number of edges in the spanner. By combining with sampling, we can improve the approximation ratio for graphs with unit edge costs and lengths.

Theorem 4.1. *There is a polynomial time randomized algorithm for* Directed 3-Spanner *for graphs with unit edge lengths with expected approximation ratio* $O(n^{1/3} \log^2 n)$.

Proof. We define thick and thin edges as in Definition 2.2, with $\beta = n^{1/3}$, and we run Sample($n^{1/3}$). By Lemma 2.1, this settles all thick edges with edge set E' that on the average has size at most $3n^{1/3} \ln n \cdot OPT$. Then we obtain solution x^* of the linear program LP-U from Fig. 2 and use randomized rounding to obtain edge set E'' that settles all thin edges with high probability. However, we need to use a different method of rounding that takes advantage of the fact that our spanners provides paths of length 3. We could have used Algorithm 2 from [11] with $\rho = \tilde{\Theta}(n^{1/3})$, but instead we give a simplified rounding scheme.

Algorithm 3. Randomized3SpannerSelection(x_e^*)

1. $E'' \leftarrow \varnothing$;
2. **for** each vertex $u \in V$ **do**
3. Let r_u be chosen i.i.d. uniformly from $[0, 1]$;
4. **end for**
5. **for** each edge $e = (u, v) \in E$ **do**
6. Add e to E'' if $r_u r_v \leq x^*_{u,v} \alpha\, n^{1/3} \ln n$; //$\alpha$ is a constant less than 10
7. **end for**
8. **return** E''.

It suffices to prove the following two lemmas. Lemma 4.1 bounds the expected size of E''. Lemma 4.2 shows that E'' settles almost all thin edges

Lemma 4.1 (analog of Lemma 4.1 in [11]). $\mathbb{E}[|E''|] = O(OPT n^{1/3} \ln^2 n)$.

Lemma 4.2 (analog of Lemma 4.2 in [11]). *If (s,t) is a thin edge, E'' contains a path from s to t of length at most 3 with probability at least $1-1/n$.*

By this lemma, the expected number of unsettled thin edges is at most $|E|/n \leq n \leq OPT$, so one can simply add the unsettled edges to the solution.

It remains to prove Lemmas 4.1 and 4.2. In the proof of Lemma 4.1, we use the following fact whose proof we omit for space considerations:

Lemma 4.3. *If $q \leq 1$, $\mathbf{P}r[r_u r_v \leq q] = q(1-\ln q)$.*

Proof (of Lemma 4.1). Let $A = \{e \in E : x^*_e \alpha\, n^{1/3} \ln n \geq 1/n\}$ and $B = E \setminus A$. We use two estimates for OPT: $OPT_1 = \sum_e x^*_e$ and $OPT_2 = |B|/n$. Clearly,
$\mathbb{E}[|E'' |] = \mathbb{E}[|E'' \cap A|] + \mathbb{E}[|E'' \cap B|]$;
$\mathbb{E}[|E'' \cap A|] \leq OPT_1 \times \alpha\, n^{1/3} \ln n(1+\ln n)$;
$\mathbb{E}[|E'' \cap B|] \leq OPT_2 \times (1+\ln n)$.
Both inequalities follow from Lemma 4.3. □

We defer the proof of Lemma 4.2 to the full version. □

5 Conclusion

We gave approximation algorithms with ratio $\tilde{O}(\sqrt{n})$ for Directed k-Spanner and with ratio $\tilde{O}(n^{1/3})$ for Directed 3-Spanner with unit length edges. It remains an interesting open question whether one improve the approximation ratio to $\tilde{O}(n^{1/3})$ for arbitrary lengths and larger k, thus matching the integrality gap shown by Dinitz and Krauthgamer. Our algorithm for Directed k-Spanner applies to the k-Transitive-Closure Spanner problem [6], which can be reformulated as a special case of Directed k-Spanner. It also straightforwardly extends to the Client-Server k-Spanner problem and the k-Diameter Spanning Subgraph problem [16].

References

1. Althöfer, I., Das, G., Dobkin, D., Joseph, D., Soares, J.: On sparse spanners of weighted graphs. Discrete & Computational Geometry 9(1), 81–100 (1993)
2. Awerbuch, B.: Communication-time trade-offs in network synchronization. In: PODC, pp. 272–276 (1985)
3. Baswana, S., Sen, S.: Approximate distance oracles for unweighted graphs in expected $\tilde{O}(n^2)$ time. ACM Transactions on Algorithms 2(4), 557–577 (2006)
4. Berman, P., Raskhodnikova, S., Ruan, G.: Finding sparser directed spanners. In: Lodaya, K., Mahajan, M. (eds.) FSTTCS. LIPIcs, vol. 8, pp. 424–435. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2010)
5. Bhattacharyya, A., Grigorescu, E., Jha, M., Jung, K., Raskhodnikova, S., Woodruff, D.P.: Lower bounds for local monotonicity reconstruction from transitive-closure spanners. In: Serna, M.J., Shaltiel, R., Jansen, K., Rolim, J.D.P. (eds.) APPROX 2010, LNCS, vol. 6302, pp. 448–461. Springer, Heidelberg (2010)

6. Bhattacharyya, A., Grigorescu, E., Jung, K., Raskhodnikova, S., Woodruff, D.: Transitive-closure spanners. In: SODA, pp. 932–941 (2009)
7. Cohen, E.: Fast algorithms for constructing t-spanners and paths with stretch t. SIAM J. Comput. 28(1), 210–236 (1998)
8. Cohen, E.: Polylog-time and near-linear work approximation scheme for undirected shortest paths. JACM 47(1), 132–166 (2000)
9. Cowen, L.: Compact routing with minimum stretch. J. Algorithms 38(1), 170–183 (2001)
10. Cowen, L., Wagner, C.G.: Compact roundtrip routing in directed networks. J. Algorithms 50(1), 79–95 (2004)
11. Dinitz, M., Krauthgamer, R.: Directed spanners via flow-based linear programs. In: Vadhan, S. (ed.) STOC. ACM, New York (to appear, 2011)
12. Dodis, Y., Khanna, S.: Designing networks with bounded pairwise distance. In: STOC, pp. 750–759 (1999)
13. Elkin, M.: Computing almost shortest paths. In: PODC, pp. 53–62 (2001)
14. Elkin, M., Peleg, D.: Strong inapproximability of the basic k-spanner problem. In: Welzl, E., Montanari, U., Rolim, J.D.P. (eds.) ICALP 2000. LNCS, vol. 1853, pp. 636–647. Springer, Heidelberg (2000)
15. Elkin, M., Peleg, D.: The client-server 2-spanner problem with applications to network design. In: SIROCCO, pp. 117–132 (2001)
16. Elkin, M., Peleg, D.: Approximating k-spanner problems for $k \geq 2$. Theor. Comput. Sci. 337(1-3), 249–277 (2005)
17. Elkin, M., Peleg, D.: The hardness of approximating spanner problems. Theory Comput. Syst. 41(4), 691–729 (2007)
18. Feigenbaum, J., Kannan, S., McGregor, A., Suri, S., Zhang, J.: Graph distances in the data-stream model. SIAM J. Comput. 38(5), 1709–1727 (2008)
19. Feldman, M., Kortsarz, G., Nutov, Z.: Improved approximating algorithms for Directed Steiner Forest. In: Mathieu, C. (ed.) SODA, pp. 922–931. SIAM, Philadelphia (2009)
20. Jha, M., Raskhodnikova, S.: Testing and reconstruction of Lipschitz functions with applications to data privacy. Electronic Colloquium on Computational Complexity (ECCC) TR11-057 (2011)
21. Kortsarz, G., Peleg, D.: Generating sparse 2-spanners. J. Algorithms 17(2), 222–236 (1994)
22. Kortsarz, G.: On the hardness of approximating spanners. Algorithmica 30(3), 432–450 (2001)
23. Peleg, D., Schäffer, A.A.: Graph spanners. J. Graph Theory 13(1), 99–116 (1989)
24. Peleg, D., Ullman, J.D.: An optimal synchronizer for the hypercube. SIAM J. Comput. 18(4), 740–747 (1989)
25. Peleg, D., Upfal, E.: A trade-off between space and efficiency for routing tables. JACM 36(3), 510–530 (1989)
26. Raskhodnikova, S.: Transitive-closure spanners: a survey. In: Goldreich, O. (ed.) Property Testing. LNCS, vol. 6390, pp. 167–196. Springer, Heidelberg (2010)
27. Roditty, L., Thorup, M., Zwick, U.: Roundtrip spanners and roundtrip routing in directed graphs. ACM Transactions on Algorithms 4(3) (2008)
28. Thorup, M., Zwick, U.: Compact routing schemes. In: SPAA, pp. 1–10. ACM, New York (2001)
29. Thorup, M., Zwick, U.: Approximate distance oracles. JACM 52(1), 1–24 (2005)

An Improved Approximation Algorithm for Minimum-Cost Subset k-Connectivity
(Extended Abstract)

Bundit Laekhanukit

School of Computer Science, McGill University
blaekh@cs.mcgill.ca

Abstract. The minimum-cost subset k-connected subgraph problem is a cornerstone problem in the area of network design with vertex connectivity requirements. In this problem, we are given a graph $G = (V, E)$ with costs on edges and a set of terminals T. The goal is to find a minimum cost subgraph such that every pair of terminals are connected by k openly (vertex) disjoint paths. In this paper, we present an approximation algorithm for the subset k-connected subgraph problem which improves on the previous best approximation guarantee of $O(k^2 \log k)$ by Nutov (FOCS 2009). Our approximation guarantee, $\alpha(|T|)$, depends upon the number of terminals:

$$\alpha(|T|) = \begin{cases} O(|T|^2) & \text{if } |T| < 2k \\ O(k \log^2 k) & \text{if } 2k \leq |T| < k^2 \\ O(k \log k) & \text{if } |T| \geq k^2 \end{cases}$$

So, when the number of terminals is *large enough*, the approximation guarantee improves significantly. Moreover, we show that, given an approximation algorithm for $|T| = k$, we can obtain almost the same approximation guarantee for any instances with $|T| > k$. This suggests that the hardest instances of the problem are when $|T| \approx k$.

1 Introduction

We present an improved approximation algorithm for the *minimum cost subset k-connected subgraph problem*. In this problem (subset k-connectivity, for short), we are given a graph $G = (V, E)$ with edge costs and a set of terminals $T \subseteq V$. The goal is to find a minimum cost subgraph such that each pair of terminals is connected by k openly (vertex) disjoint paths. This is a fundamental problem in network design which includes as special cases the minimum-cost Steiner tree problem (the case $k = 1$) and the minimum-cost k vertex-connected spanning subgraph problem (the case $T = V$). However, the subset k-connectivity problem is significantly harder than these two special cases. Specifically, an important result of Kortsarz, Krauthgamer and Lee [12] shows that the problem does not admit an approximation guarantee better than $O(2^{\log^{1-\epsilon} n})$ for any $\epsilon > 0$ unless

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 13–24, 2011.

NP $\subseteq$ DTIME($n^{O(\text{polylog}(n))}$). In contrast, polylogarithmic approximation guarantees are known for the minimum-cost k-vertex connected spanning subgraph problem. The first such result was obtained by Fakcharoenphol and Laekhanukit [8] using a technique that we will call the *Halo-set method*. Subsequently, Nutov [16] refined the algorithm and analysis of [8] to obtain the current best approximation guarantee of $O\left(\log k \cdot \log \frac{n}{n-k}\right)$.

Despite being studied for a decade, no non-trivial approximation algorithm was known for the subset-k-connectivity problem until the ground-breaking work of Chakraborty, Chuzhoy and Khanna [2]. They presented an $f(k,|T|)$-approximation algorithm for the rooted version of our problem, namely the *rooted subset k-connectivity problem* (where $f(k,|T|) = O(k^{O(k^2)} \cdot \log^4 |T|)$). There, given a root vertex r and a set of terminals T, the goal is to find a minimum cost subgraph that has k openly disjoint paths from the root vertex r to every terminal in T. Chakraborty et al. showed how to solve the subset k-connectivity problem by applying the rooted subset k-connectivity algorithm k times, thus obtaining an $(k \cdot f(k,|T|))$-approximation algorithm. Recently, in a series of developments [2,6,7,3,17,18], the approximation guarantee, $f(k,|T|)$, for the rooted subset k-connectivity has been steadily improved. This has culminated in an $O(k \log k)$ guarantee due to Nutov [17]. Consequently, the current best known approximation guarantee for the subset k-connectivity problem is $O(k^2 \log k)$.

As we will show, there is a trivial way to obtain an approximation bound of $O(|T|^2)$. So, with the current progress on the rooted subset k-connectivity problem, the application of the rooted subroutine is only useful when the number of terminals is large enough, say $|T| \geq 2k$. The main contribution of this paper is to show that, in this case, only a polylogarithmic number of applications of the rooted subset k-connectivity algorithm are required to solve the subset k-connectivity problem. Given an approximation algorithm for the rooted subset k-connectivity problem, we show that only $O(\log^2 k)$ applications of the algorithm are required. Moreover, as the number of terminal increases above k^2, we are able to save an additional $O(\log k)$ factor. Thus, given an approximation algorithm for rooted subset k-connectivity problem in [17] due to Nutov (and with careful analysis), we achieve an $\alpha(|T|)$-approximation guarantee where:

$$\alpha(|T|) \quad = \quad \begin{cases} O(|T|^2) & \text{if } |T| < 2k \\ O(k \log^2 k) & \text{if } 2k \leq |T| < k^2 \\ O(k \log k) & \text{if } |T| \geq k^2 \end{cases}$$

Interestingly, the key to achieving this result is a very careful application of the Halo-set method. This method was introduced by Kortsarz and Nutov in [13] and developed further by Fakcharoenphol and Laekhanukit in [8]. The technique has been successfully used in [13,8,16,17,18,15]. The original method applies only when certain subsets of vertices, called *cores*, are disjoint on the set of terminals. However, for the subset k-connectivity problem, the cores are not disjoint even on the set of terminals. Overcoming this difficulty is the main technical contribution of this paper.

Consequently, we improve upon the current best approximation guarantee of $O(k^2 \log k)$ in all cases. In general, we obtain a significant improvement of a factor of k. Observe, however, that for the case of $|T| \approx k$ our guarantee is still quadratic. At first, this may seem paradoxical since we may hope that the problem is easier when the number of terminals is small. Our results suggest that this is not the case. Indeed, it appears that the hardest instances of subset k-connectivity may have at most k terminals. Precisely, we show that, given an $\alpha(k)$-approximation algorithm for the subset k-connectivity problem with $|T| = k$, there is an $(\alpha(k) + f(k))$-approximation algorithm for any instance with $|T| > k$, where $f(k)$ is the best known approximation guarantee for the rooted subset k-connectivity problem.

Related Work. Some very special cases of the subset k-connectivity problem are known to have constant factor approximation algorithms. For $k = 1$, the minimum-cost Steiner tree problem, the best known approximation guarantee is 1.39 due to Byrka, Grandoni, Rothvoß and Sanità [1]. For $k = 2$, a factor two approximation algorithm algorithm was given by Fleischer, Jain and Williamson [9]. The subset k-connectivity problem also has an $O(1)$-approximation algorithm when edge costs satisfy the triangle inequality; see Cheriyan and Vetta [5]. The most general problem in this area is the vertex-connectivity survivable network design problem (VC-SNDP). In VC-SNDP, the connectivity requirement for each pair of vertices can be arbitrary. Recently, Chuzhoy and Khanna [7] showed that there is an $O(k^3 \log n)$-approximation algorithm for VC-SNDP. The problems where requirements are edge and element connectivity (EC-SNDP and Element-SNDP) are also very well studied. Both problems admit 2-approximation algorithms via iterative rounding. For EC-SNDP, a 2-approximation algorithm was given by Jain [11]. For element-SNDP a 2-approximation algorithm was given by Fleischer, Jain and Williamson [9].

2 Preliminaries and Results

We begin with some formal definitions. Let $G = (V, E)$ denote the graph for an instance of the problem. For a set of edges F, the graph $G' = (V, E \cup F)$ is denoted by $G + F$; for a vertex v, the graph obtained from G by removing v is denoted by $G - v$. For any set of vertices $U \subseteq V$, let $\Gamma(U)$ denote the set of *neighbors* of U; that is, $\Gamma(U) = \{v \in V - U : \exists (u, v) \in E : u \in U\}$. Define a set U^* to be $V - (U \cup \Gamma(U))$, which is the *vertex-complement* of U. For any pair of vertices $s, t \in V$, two s,t-paths are *openly disjoint* if they have no vertices except s and t in common. Let $T \subseteq V$ be a set of vertices called *terminals*. Without loss of generality, assume that no two terminals of T are adjacent in G. This assumption can be easily justified by subdividing every edge joining two terminals; that is, if there is an edge (s, t) joining two terminals, then we replace (s, t) by two new edges (s, u) and (u, t) and set cost of the new edges so that $c(s, t) = c(s, u) + c(u, t)$, where $c(.)$ is a cost function. The graph G is *subset k-connected on T* if G has k openly disjoint s, t-paths between every pair

of terminals $s, t \in T$. Thus, by Menger-Whitney Theorem, the removal of any set of vertices of size at most $k-1$ leaves all the remaining terminals in the same component of the remaining graph. By the *subset connectivity* of G on T, we mean the maximum integer ℓ such that G is ℓ-connected on T. A *deficient set* is a subset of vertices $U \subseteq V$ such that both U and U^* contain terminals of T and $|\Gamma(U)| < k$. Observe that the vertex-complement U^* is also a deficient set. Similarly, given a designated *root* vertex r, the graph is *r-rooted subset k-connected to T* if G has k openly disjoint r,t-paths for every terminal $t \in T$ (r may or may not be in T). By the *rooted connectivity* of G from r to T, we mean the maximum integer ℓ such that G is r-rooted subset ℓ-connected to T.

In the *subset k-connectivity problem*, we are given a graph $G = (V, E)$ with a cost $c(e)$ on each edge $e \in E$, a set of terminals $T \subseteq V$, and an integer $k \geq 0$. The goal is to find a set of edges $\widehat{E} \subseteq E$ of minimum cost such that the subgraph $\widehat{G} = (V, \widehat{E})$ is subset k-connected on T. In the *rooted subset k-connectivity problem*, our goal is to find a set of edges $\widehat{E} \subseteq E$ of minimum cost such that the subgraph $\widehat{G} = (V, \widehat{E})$ is r-rooted k-connected to T, for a given root r.

Nutov [17] recently gave an $O(k \log k)$-approximation algorithm for the rooted subset k-connectivity problem. The approximation guarantee improves by a logarithmic factor for the problem of increasing the rooted connectivity by one, and the guarantee also depends on the size of deficient sets.

Theorem 1 (Nutov 2009 [17]). *There is an $O(k \log k)$-approximation algorithm for the rooted subset k-connectivity problem. Moreover, consider the restricted version of the problem where the goal is to increase the rooted connectivity from ℓ to $\ell + 1$. Let $\phi = \min\{|U \cap T| : U \text{ is a deficient set}\}$. Then the approximation guarantee (with respect to a standard LP) is $O(\ell/\phi)$.*

Our focus is upon the subset k-connectivity problem. Nutov's result leads to an $O(k^2 \log k)$-approximation algorithm for this problem. The following theorems are our main results:

Theorem 2. *For any set T of terminals, there is an $\alpha(|T|)$-approximation algorithm for the subset k-connectivity problem where*

$$\alpha(|T|) \quad = \quad \begin{cases} O(|T|^2) & \text{if } |T| < 2k \\ O(k \log^2 k) & \text{if } 2k \leq |T| < k^2 \\ O(k \log k) & \text{if } |T| \geq k^2 \end{cases}$$

Remark 1. Although we do not discuss the case $k < |T| \leq 2k$ in this paper, we also have an approximation algorithm for this case. Our algorithm gives an approximation guarantee of $O(\frac{k^2}{|T|-k} \log^2 k)$ which is strictly better than $O(k^2)$ for $|T| - k > \log^2 k$.

Theorem 3. *Consider the subset k-connectivity problem. Suppose there is an $\alpha(k)$-approximation algorithm for instances with $|T| = k$. Then there is an $(\alpha(k) + f(k))$-approximation algorithm for any instance with $|T| > k$, where $f(k)$ is the best known approximation guarantee for the rooted subset k-connectivity problem.*

Similar results and proofs here have appeared in previous literature. See [4,13,14]. In particular, Lemma 9 and Lemma 11 have appeared in [13] and [14], respectively. Our key new contributions are Lemmas 4, 8 and 7 which allow us to extend the results to the subset k-connectivity problem.

Due to the page limit, most proofs are omitted. The full version containing all proofs is available in [15].

3 An Approximation Algorithm

Our main result in Theorem 2 breaks up into three cases where there are a small number, a moderate number and a large number of terminals, respectively. When there are a small number of terminals ($|T| < 2k$), we apply the following trivial $O(|T|^2)$-approximation algorithm. We find k openly disjoint paths of minimum cost between every pair of terminals, by applying a minimum-cost flow algorithm. Let *opt* denote the cost of the optimal solution to the subset k-connectivity problem. Since any feasible solution to the subset k-connectivity problem has k openly disjoint paths between every pair of terminals, the cost incurred by finding a minimum cost collection of k openly disjoint paths between any pair of terminals is at most *opt*. Since we have at most $|T|^2$ pairs, this incurs a total cost of $O(|T|^2 \cdot opt)$.

The remaining two cases are similar. Things are slightly easier, though, when there are large number of terminals ($|T| \geq k^2$), leading to a slightly better guarantee than when there are a moderate number of terminals ($2k \leq |T|$). We devote most of this section to presenting an approximation algorithm for the moderate case. (In Section 3.6, we show the improvement for the case of a large number of terminals.)

Our algorithm is based on the Halo-set method, which is an effective algorithmic paradigm for problems in network design with vertex connectivity requirements; see, for example, [13,8,16,17,18].

The algorithm works by repeatedly increasing the subset connectivity of a graph by one. We start with a graph that has no edges. Then we apply k *outer* iterations. Each outer iteration increases the subset connectivity (of the current graph) by one by adding a set of edges of approximately minimum cost. The analysis of the outer iterations applies linear programming (LP) scaling and incurs a factor of $O(\log k)$ in the approximation guarantee for the k outer iterations. The analysis based on LP-scaling can be seen in [19,5,13,8] and also in [10,14].

Lemma 1. *Suppose there is a $\beta(\ell)$-approximation algorithm for the problem of increasing the subset connectivity of a graph from ℓ to $\ell + 1$ with respect to a standard LP, where $\beta(\ell)$ is a non-decreasing function. Then there is an $O(\beta(k) \log k)$-approximation algorithm for the subset k-connectivity problem.*

We are left with the key problem of increasing the subset connectivity (of the current graph) by one by adding a set of edges of approximately minimum cost. Throughout this section, we assume that the current graph is subset ℓ-connected on T, and $|T| \geq 2k \geq 2\ell$. Also, we assume that no two terminals are adjacent in the input graph $G = (V, E)$.

Assumption: The current graph $\widehat{G} = (V, \widehat{E})$ is subset ℓ-connected on T, and $|T| \geq 2k \geq 2\ell$. Moreover, no two terminals are adjacent in the input graph G.

The Halo-set method solves the problem of increasing the subset connectivity of a graph by one by applying a number of so-called *inner* iterations. To describe our algorithm, we need some definitions and subroutines. Thus, we defer the description of our algorithm to Section 3.5. In Section 3.1, we give important definitions and structures of subset ℓ-connected graphs called "cores" and "halo-families". Our algorithm requires two subroutines. The first one is the subroutine that uses the rooted subset $(\ell+1)$-connectivity algorithm to cover halo-families. This subroutine is given in Section 3.2. The second one is the subroutine for decreasing the number of cores to $O(\ell)$, which is given in Section 3.3. Then we introduce a notion of "thickness" in Section 3.4. This notion guides us how to use the rooted subset $(\ell + 1)$-connectivity algorithm efficiently. Finally, in Section 3.5, we present an $O(k \log^2 k)$-approximation algorithm for $|T| \geq 2k$. By slightly modifying the algorithm and analysis, we show in Section 3.6 that our algorithm achieves a better approximation guarantee of $O(k \log k)$ when $|T| \geq k^2$.

3.1 Subset ℓ-Connected Graphs: Deficient Sets, Cores, Halo-Families and Halo-Sets

In this section, we discuss some key properties of deficient sets that will be exploited by our approximation algorithm.

Assume that the graph $G = (V, E)$ is subset ℓ-connected on the set of terminals T. Then G has $|\Gamma(U)| \geq \ell$ for all $U \subseteq V$ such that $U \cap T \neq \emptyset$ and $U^* \cap T \neq \emptyset$. Moreover, by Menger-Whitney Theorem, G is subset $(\ell + 1)$-connected if and only if G has no deficient set.

A key property of vertex neighborhoods is that the function $|\Gamma(\cdot)|$ on subsets of V is submodular. In other words, for any subsets of vertices $U, W \subseteq V$,

$$|\Gamma(U \cup W)| + |\Gamma(U \cap W)| \leq |\Gamma(U)| + |\Gamma(W)|.$$

We call a deficient set $U \subseteq V$ *small* if $|U \cap T| \leq |U^* \cap T|$.

Proposition 1. *For any small deficient set U, $|U \cap T| \leq |T|/2$ and $|U^* \cap T| \geq (|T| - \ell)/2$.*

Lemma 2 (Uncrossing Lemma). *Consider any two distinct deficient sets $U, W \subseteq V$. If $U \cap W \cap T \neq \emptyset$ and $U^* \cap W^* \cap T \neq \emptyset$, then both $U \cap W$ and $U \cup W$ are deficient sets. Moreover, if U or W is a small deficient set, then $U \cap W$ is a small deficient set.*

By a *core* we mean a small deficient set C that is inclusionwise minimal. In other words, C is a core if it is a small deficient set that does not contain another such set. Note that any small deficient set U contains at least one core.

The *halo-family* of a core C, denoted by $\mathrm{Halo}(C)$, is the set of all small deficient sets that contain C and contain no other cores; that is,

$$\mathrm{Halo}(C) = \{U : U \text{ is a small deficient set, } C \subseteq U, U \text{ contains no core } D \neq C\}$$

The *halo-set* of a core C, denoted by $H(C)$, is the union of all the sets in Halo(C); that is,

$$H(C) = \bigcup \{U : U \in \text{Halo}(C)\}$$

We remark that cores and halo-sets can be computed in polynomial-time. See [13,8,14]. Some important properties of cores and halo-families that we will require are stated below.

Lemma 3 (Disjointness Lemma). *Consider any two distinct cores C and D. For any deficient sets $U \in \text{Halo}(C)$ and $W \in \text{Halo}(D)$, either $U \cap W \cap T = \emptyset$ or $U^* \cap W^* \cap T = \emptyset$.*

The next result gives an upper bound on the number of halo-sets that contain a chosen terminal. The next result gives a key bound for the design of our algorithm.

Lemma 4 (Upper bound). *For any terminal $t \in T$, the number of cores C such that $t \in H(C)$ is at most $\frac{2(|T|-1)}{|T|-\ell}$.*

3.2 Covering Halo-Families via Rooted Subset $(\ell+1)$-Connectivity

We say that an edge $e = (u, v)$ *covers* a deficient set U if e connects U and U^*; that is, $u \in U$ and $v \in U^*$. Clearly, e covers U if e covers U^*. Observe that if e covers U, then after adding the edge e to the current graph, U is no longer a deficient set. Now, consider any core C. We say that a set of edges F *covers* the halo-family of C if each deficient set U in Halo(C) is covered by some edge of F. For a terminal $r \in T$, we say that the terminal r *hits* the halo-family Halo(C) if r is in C or r is in the vertex-complement of the halo-set of C; that is, r hits Halo(C) if $r \in C$ or $r \in H(C)^*$. For a set of terminals $S \subseteq T$, we say that S *hits* a halo-family Halo(C) if there is a terminal $r \in S$ that hits Halo(C). The following lemma shows that if r hits the halo-family Halo(C), then we can find a set of edges F that covers Halo(C) by applying the rooted subset $(\ell+1)$-connectivity algorithm with r as the root.

Lemma 5. *Consider a set of edges F whose addition to $\widehat{G}$ makes the resulting graph $\widehat{G}+F$ rooted $(\ell+1)$-connected from a terminal r to T. Let C be any core. If $r \in C$ or $r \in H(C)^*$, then F covers all deficient sets in the halo-family of C.*

3.3 Preprocessing to Decrease the Number of Cores

In this section, we describe the preprocessing algorithm that decreases the number of cores to $O\left(\frac{\ell|T|}{|T|-\ell}\right)$. We apply the following *root padding algorithm* in the preprocessing step.

The root padding algorithm: The algorithm takes as an input a graph $G = (V, E)$ with the given edge costs, a subset of terminals $R \subseteq T$, and a connectivity parameter ρ. We construct a padded graph by adding a new vertex

$\widehat{r}$ and new edges of zero cost from $\widehat{r}$ to each terminal of R. Then we apply the rooted subset ρ-connectivity algorithm to the padded graph with the set of terminals T and the root $\widehat{r}$. We denote a solution subgraph (of the padded graph) by $\widehat{G} = (V + \widehat{r}, F \cup \{(\widehat{r}, t) : t \in R\})$, where $F \subseteq E$. Then the algorithm outputs the subgraph (of the original graph) $\widehat{G} - \widehat{r} = (V, F)$. The following result shows that, in the resulting graph $\widehat{G}$, every deficient set contains at least one terminal of R.

Lemma 6 (root padding). *Suppose we apply the root padding algorithm as above, and it finds a subgraph $\widehat{G} - \widehat{r} = (V, F)$. Then every deficient set of $\widehat{G} - \widehat{r}$ (with respect to subset ρ-connectivity on T) contains at least one terminal of R.*

Next, recall that $|T| \geq 2\ell$. We apply the root padding algorithm in Lemma 6 to any subset R of $(\ell + 1)$ terminals with $\rho = (\ell + 1)$. By Theorem 1, this incurs a cost of at most $O((k \log k) \cdot opt)$ Moreover, the algorithm adds a set of edges to the current graph such that every deficient set of the resulting graph contains at least one terminal of R. Thus, each core of the resulting graph contains at least one terminal of R. By Lemma 4, each terminal is in $O\left(\frac{|T|}{|T|-\ell}\right) = O(1)$ halo-sets. Hence, the number of cores in the resulting graph is at most $O(\ell)$. This gives the next result.

Lemma 7. *Given a subset ℓ-connected graph, where $|T| \geq 2\ell$, there is an $f(k)$-approximation algorithm that decreases the number of cores to $O(\ell)$, where $f(k)$ is the best known approximation guarantee for the rooted subset k-connectivity problem.*

3.4 Thickness of Terminals

Consider a graph that is subset ℓ-connected on T. We define the *thickness* of a terminal $t \in T$ to be the number of halo-families $\text{Halo}(C)$ such that $t \in \Gamma(H(C))$. Thus, the thickness of a terminal t is $|\{\text{Halo}(C) : C \text{ is a core}, t \in \Gamma(H(C))\}|$.

The following lemmas show the existence of a terminal with low thickness.

Lemma 8. *For every core C, $|\Gamma(H(C))| \leq \ell$.*

The following lemma shows the existence of a terminal with low thickness.

Lemma 9. *Consider a subset ℓ-connected graph. Let q denote the number of halo-families. Then there exists a terminal $t \in T$ with thickness at most $\frac{\ell q}{|T|}$.*

3.5 An $O(k \log^2 k)$-Approximation Algorithm for $|T| \geq 2k$

In this section, we describe our approximation algorithm for the case of a moderate number of terminals. Recall that we solve the problem by iteratively increasing the subset connectivity by one. Initially, we apply the algorithm in Section 3.3 to decrease the number of core to $O(\ell)$. Then we apply inner iterations until all

the deficient sets are covered. At the beginning of each inner iteration, we compute the cores and the halo-sets. Then we apply a *covering-procedure* to find a set of edges that covers all the computed halo-families. This completes one inner iteration. Note that an inner iteration may not cover all of the deficient sets because deficient sets that contain two or more of the initial cores (those computed at the start of the inner iteration) may not be covered. So, we have to repeatedly apply inner iterations until no core is present. See Figure 1.

An approximation algorithm for moderate and large number of terminals:

1. For $\ell = 0, 1, \ldots, k-1$ (outer iterations):
2. *(* Increase the subset connectivity of a graph by one. *)*
3. Decrease the number of cores to $O(\ell)$.
4. While the number of cores is greater than 0 (inner iterations):
5. Compute cores and halo-sets.
6. Apply a **covering-procedure** to cover all the halo-families.

Fig. 1. An approximation algorithm for moderate and larger number of terminals

We now describe the covering-procedure. The procedure first finds a set of terminals $S \subseteq T$ that hits all the computed halo-families. Then it applies the rooted subset $(\ell+1)$-connectivity algorithm (Theorem 1) from each terminal of S. Let F be the union of all edges found by the rooted subset $(\ell+1)$-connectivity algorithm. Then, by Lemma 5, F covers all the halo-families.

The key idea of our algorithm is to pick a terminal $\widehat{r}$ of minimum thickness. Observe that a halo-family Halo(C) is not hit by $\widehat{r}$ only if

(1) its halo-set $H(C)$ has $\widehat{r}$ as a neighbour (that is, $\widehat{r} \in \Gamma(H(C))$) or
(2) its halo-set $H(C)$ contains $\widehat{r}$, but its core C does not contain $\widehat{r}$.

The number of halo-families Halo(C) such that $\widehat{r} \in \Gamma(H(C))$ may be large, but the number of halo-families whose halo-sets contain $\widehat{r}$ is $O(1)$, assuming that $|T| \geq 2\ell$. Hence, we only hit halo-families of the second case by picking one terminal from each core C whose halo-set contains $\widehat{r}$. Thus, the number of terminals picked is $O(1)$. We call this a *micro* iteration. Then the remaining halo-families are the halo-families whose halo-sets have $\widehat{r}$ as a neighbour. We repeatedly apply micro iterations until we hit all of the halo-families computed at the start of the inner iteration.

To be precise, initially let $S = \emptyset$. In each micro iteration, we add to S a terminal $\widehat{r}$ of minimum thickness (with respect to halo-families that are not hit by S). Then, for each core C such that $\widehat{r} \in H(C) - C$, we add to S any terminal in $C \cap T$. We repeatedly apply micro iterations until S hits all the halo-families. At the termination, we apply the rooted subset $(\ell+1)$-connectivity algorithm (Theorem 1) from each terminal of S, and we return all the set of edges found by the algorithm as an output. The covering-procedure is presented in Figure 2.

A covering-procedure:

1. $S \leftarrow \emptyset$.
2. While some halo-family is not hit by S (micro iteration):
3. Add to S a terminal $\widehat{r}$ of minimum thickness.
4. For each halo-family Halo(C) (not hit by S) such that $\widehat{r} \in H(C) - C$:
5. Add to S any terminal $r \in C$.
6. For each terminals r in S:
7. Apply the rooted subset $(\ell+1)$-connectivity algorithm from r.

Fig. 2. A covering-procedure

Analysis: The feasibility of a solution directly follows from the condition of the inner iteration; that is, the inner iteration terminates when a current graph has no core. So, at the termination of the inner iteration, the resulting graph has no deficient set. Thus, the subset connectivity of the graph becomes $\ell+1$. Applying the outer iteration k times, the final graph is then subset k-connected.

It remains to analyze the cost of the solution subgraph. First, we analyze the cost incurred by the covering-procedure. Then we analyze the number of inner iterations and the total cost incurred by solving the problem of increasing the subset connectivity of a graph by one. Finally, we apply Theorem 1 to analyze the final approximation guarantee.

Consider any micro iteration of the covering-procedure. By Lemma 4, $\widehat{r}$ is contained in at most $O(1)$ halo-sets, assuming that $|T| \geq 2\ell$. Hence, we have to apply the rooted subset $(\ell+1)$-connectivity algorithm $O(1)$ times.

Thus, Theorem 1 implies that the cost incurred by each micro iteration is $O(\frac{\ell}{\phi} \cdot opt)$, where opt is the cost of the optimal solution to the subset $(\ell+1)$-connectivity problem and $\phi = \min\{|U \cap T| : U \text{ is a deficient set}\}$.

We now analyze the number of micro iterations needed to hit all of the halo-families. Let h_i denote the number of halo-families that are not hit by S at the beginning of the i-th micro iteration. Recall that the number of cores after the preprocessing step is $O(\ell)$. Thus, $h_1 = O(\ell)$. We claim that, at the i-th iteration, the number of halo-families that are not hit by S is at most $h_1/2^{i-1}$.

Lemma 10. *Consider the i-th micro iteration. The number of halo-families that are not hit by terminals of S at the start of the iteration is $h_1/2^{i-1}$.*

Lemma 10 implies that the maximum number of micro iterations (within the covering-procedure) is $O(\log h_1) = O(\log \ell)$. So, the cost incurred by the covering-procedure is $O(\frac{\ell}{\phi} \log \ell \cdot opt)$.

Lastly, we analyze the number of inner iterations by focusing on the *maximum number of cores that are disjoint on T*. Recall to the preprocessing step in Section 3.3. After the preprocessing, every core contains at least one terminal of R. This means that the maximum number of cores that are disjoint on T is $\ell+1$. Consider the cores at any inner iteration. We call cores at the beginning of the iteration *old cores* and call cores at the end of the iteration *new cores*. We

claim that every new core $\widehat{C}$ contains at least two old cores C and D that are disjoint on T. This follows from the following lemma.

Lemma 11. *No small deficient set contains two distinct cores C and D such that $C \cap D \cap T \neq \emptyset$.*

Lemma 11 implies that no new cores contain two old cores that are intersecting on T. This is because new cores are small deficient sets of the old graph. Moreover, since all small deficient sets that contain only one core have been covered, new cores must contain at least two old cores that are disjoint on T. Thus, the number ϕ of the smallest number of terminals containing in any core (and deficient set) increases by a factor of two. More precisely, $\phi \leq 2\phi'$, where ϕ and ϕ' denote the smallest number such that any deficient set U of the new graph has $|U \cap T| \geq \phi$ and any deficient set U' of the old graph has $|U' \cap T| \geq \phi'$. In particular, the number ϕ at the j-th inner iteration is 2^{j-1}.

Combining everything together, the approximation guarantee for the problem of increasing the subset connectivity of a graph by one is

$$O\left(\frac{\ell}{2^0}\log \ell + \frac{\ell}{2^1}\log \ell + \ldots\right) = O(\ell \log \ell).$$

Thus, by Theorem 1, our algorithm achieves an approximation guarantee of $O(k \log^2 k)$, assuming that $|T| \geq 2k$.

3.6 An $O(k \log k)$-Approximation Algorithm for $|T| \geq k^2$

To finish, we show that if the number of terminals is large, we get a slightly better performance guarantee. Observe that if $|T| \geq k^2$ then, by Lemma 9, there is a terminal $\widehat{r}$ with a thickness of at most $\frac{q\ell}{|T|} \leq \frac{2\ell^2}{\ell^2} = 2$. Moreover, by Lemma 4, each terminal is contained in at most $\frac{2|T|}{|T|-\ell} = O(1)$ halo-sets. Thus, the number of halo-families that are not hit by $\widehat{r}$ is $O(1)$. This means that we can hit all the remaining halo-families by choosing $O(1)$ terminals; that is, for each halo-family, we choose one terminal from its core. So, we can skip the micro iterations of the covering-procedure, and the approximation guarantee becomes $O(k \log k)$.

Acknowledgments. We thank Joseph Cheriyan for useful discussions over a year. Also, we thank Parinya Chalermsook, Jittat Fakcharoenphol, Danupon Nanongkai, Adrian Vetta and anonymous referees for useful comments.

References

1. Byrka, J., Grandoni, F., Rothvoß, T., Sanità, L.: An improved LP-based approximation for steiner tree. In: STOC, pp. 583–592 (2010)
2. Chakraborty, T., Chuzhoy, J., Khanna, S.: Network design for vertex connectivity. In: STOC, pp. 167–176 (2008)

3. Chekuri, C., Korula, N.: A graph reduction step preserving element-connectivity and applications. In: Albers, S., Marchetti-Spaccamela, A., Matias, Y., Nikoletseas, S., Thomas, W. (eds.) ICALP 2009. LNCS, vol. 5555, pp. 254–265. Springer, Heidelberg (2009)
4. Cheriyan, J., Vempala, S., Vetta, A.: An approximation algorithm for the minimum-cost k-vertex connected subgraph. SICOMP 32(4), 1050–1055 (2003); Preliminary version in STOC
5. Cheriyan, J., Vetta, A.: Approximation algorithms for network design with metric costs. SIDMA 21(3), 612–636 (2007); Preliminary version in STOC
6. Chuzhoy, J., Khanna, S.: Algorithms for single-source vertex connectivity. In: FOCS, pp. 105–114 (2008)
7. Chuzhoy, J., Khanna, S.: An ($O(k^3 \log n)$-approximation algorithm for vertex-connectivity survivable network design. In: FOCS, pp. 437–441 (2009)
8. Fakcharoenphol, J., Laekhanukit, B.: An $O(\log^2 k)$-approximation algorithm for the k-vertex connected spanning subgraph problem. In: STOC, pp. 153–158 (2008)
9. Fleischer, L., Jain, K., Williamson, D.P.: Iterative rounding 2-approximation algorithms for minimum-cost vertex connectivity problems. JCSS 72(5), 838–867 (2006); Preliminary versions in FOCS and IPCO
10. Goemans, M.X., Goldberg, A.V., Plotkin, S.A., Shmoys, D.B., Tardos, É., Williamson, D.P.: Improved approximation algorithms for network design problems. In: SODA, pp. 223–232 (1994)
11. Jain, K.: A factor 2 approximation algorithm for the generalized steiner network problem. Combinatorica 21(1), 39–60 (2001); Preliminary version in FOCS
12. Kortsarz, G., Krauthgamer, R., Lee, J.R.: Hardness of approximation for vertex-connectivity network design problems. SICOMP 33(3), 704–720 (2004); Preliminary version in APPROX
13. Kortsarz, G., Nutov, Z.: Approximating k-node connected subgraphs via critical graphs. SICOMP 35(1), 247–257 (2005); Preliminary version in STOC
14. Laekhanukit, B.: Approximation algorithms for (S,T)-connectivity problems. Master's thesis, University of Waterloo, Canada (2010)
15. Laekhanukit, B.: An improved approximation algorithm for the minimum-cost subset k-connected subgraph problem (2011) (manuscript), http://arxiv.org/abs/1104.3923
16. Nutov, Z.: An almost O(logk)-approximation for k-connected subgraphs. In: SODA, pp. 912–921 (2009)
17. Nutov, Z.: Approximating minimum cost connectivity problems via uncrossable bifamilies and spider-cover decompositions. In: FOCS, pp. 417–426 (2009)
18. Nutov, Z.: A note on rooted survivable networks. IPL 109(19), 1114–1119 (2009); Preliminary version in SODA
19. Ravi, R., Williamson, D.P.: An approximation algorithm for minimum-cost vertex-connectivity problems. Algorithmica 18(1), 21–43 (1997); Preliminary version in SODA. Erratum [20]
20. Ravi, R., Williamson, D.P.: Erratum: An approximation algorithm for minimum-cost vertex-connectivity problems. Algorithmica 34(1), 98–107 (2002); Preliminary version in SODA

Approximation Schemes for Capacitated Geometric Network Design*

Anna Adamaszek[1], Artur Czumaj[1], Andrzej Lingas[2], and Jakub Onufry Wojtaszczyk[3]

[1] DIMAP and Department of Computer Science, University of Warwick, UK
{A.M.Adamaszek,A.Czumaj}@warwick.ac.uk
[2] Department of Computer Science, Lund University, Sweden
andrzej@cs.lth.se
[3] Google, Inc.
onufry@google.com

Abstract. We study a *capacitated network design problem* in *geometric* setting. We assume that the input consists of an integral link capacity k and two sets of points on a plane, sources and sinks, each source/sink having an associated integral demand (amount of flow to be shipped from/to). The *capacitated geometric network design problem* is to construct a minimum-length network N that allows to route the requested flow from sources to sinks, such that each link in N has capacity k; the flow is splittable and parallel links are allowed in N.

The capacitated geometric network design problem generalizes, among others, the geometric Steiner tree problem, and as such it is NP-hard.

We show that if the demands are polynomially bounded and the link capacity k is not too large, the single-sink capacitated geometric network design problem admits a *polynomial-time approximation scheme*. If the capacity is arbitrarily large, then we design a *quasi-polynomial time approximation scheme* for the capacitated geometric network design problem allowing for arbitrary number of sinks. Our results rely on a derivation of an upper bound on the number of vertices different from sources and sinks (the so called Steiner vertices) in an optimal network. The bound is polynomial in the total demand of the sources.

1 Introduction

The area of *network design*, problems of designing low cost networks satisfying some predefined constraints, plays a fundamental role in operational research and graph algorithms. We consider one important class of problems in this category, the *capacitated network design problem*: for a given input network N, find a multiset of links in N of minimum cost that will allow to route a predetermined amount of flow from a set of sources to a set of sinks subject to the capacity constraints on the links.

While network design problems have been extensively studied in operational research and combinatorial optimization, traditionally the main focus has been on problems modeled by arbitrary graphs. However, as it has been observed in some more applied papers, many applications require to consider network design problems in the

* Research supported in part by the Royal Society IJP-2006/R2, the Centre for Discrete Mathematics and its Applications (DIMAP), EPSRC EP/D063191/1, and VR grant 621-2005-4085.

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 25–36, 2011.

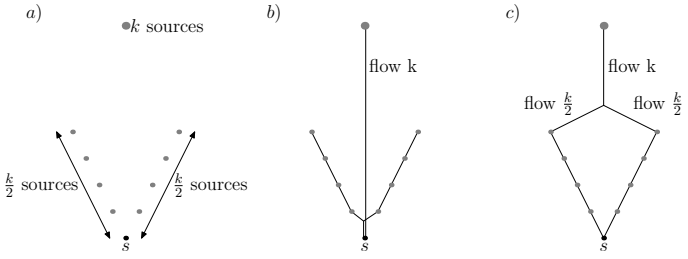


Fig. 1. A network for which the best solution which does not allow splitting the flow (b) has a significantly larger cost than the best solution which allows splitting the flow (c). All sources have unit demands and the single sink s has demand $2k$.

geometric setting. For example, when Salman *et al.* [10] initiated the graph-algorithmic study of the *buy-at-bulk network design problem* (besides the underlying graph with distinguished source and sink nodes, and their demands, there is also given a discrete set of types of links having different capacities and costs that can be used to assemble a network), they considered Euclidean graphs as a central case. Only recently the problem has been considered in the context of *geometric* network design [2], where the only input are the locations of sources and sinks, their demands, and a set of types of links.

In this paper, we study the capacitated network design problem in *geometric* setting. The input consists of an integral *link capacity* k and two sets of points on a plane, *sources* and *sinks*, each source/sink having an associated integral *demand* (amount of flow to be shipped from/to). The capacitated geometric network design problem is to construct a minimum-length network N that allows to route the requested flow from sources to sinks, such that each link in N has capacity k. We assume the flow is *splittable* and *parallel links* are allowed in N. (Observe that this problem can be considered as a special case of the geometric buy-at-bulk network design problem, in which only a single type of links having a capacity constraint is available.)

We term the studied problem as *capacitated geometric network design* ***(CGND)*** or *single-sink capacitated geometric network design* ***(SCGND)*** if there is only one sink.

It is not difficult to observe that even the single-sink variant is NP-hard, as it includes as a special case the minimum Euclidean Steiner tree problem [4][1]. Therefore, we focus on the design of fast approximation schemes.

Unlike in the Steiner tree problem, a near optimal capacitated geometric network is not necessarily a tree (see, e.g., Fig. 1). This complicates the task of deriving efficient approximation schemes for CGND and SCGND. Another major difficulty is caused by the so-called *Steiner vertices*, i.e., vertices different from the sources and sinks where the links meet and branch. Since Steiner points can be arbitrary points on the plane, one can consider all points in the vicinity of sources and sinks as potential candidates for Steiner vertices. In the very special case of minimum Euclidean Steiner tree the number of Steiner vertices in an optimal solution can be easily upper bounded by $n-2$, where n is the number of input points. If the network edges are required to be vertical or horizontal, then the Steiner points can be constrained to the quadratic number of

[1] We can model the Steiner tree problem by the SCGND problem, by taking one of the input points as sink, all other points as sources of unit demand, and set the link capacity to $n-1$.

intersections between vertical and horizontal straight-lines passing through the sources and sinks, the so called Hanan grid [2,11]. However, in the general Euclidean case that we consider here, the problem of bounding the number of Steiner points in terms of the total demand of input sources has been open [2].

Our contributions. We present three results for the CGND problem. First, we study structural properties of (near) optimal solutions for CGND. Our main result here is an upper bound for the number of Steiner points in an optimal solution that is polynomial in the total demand of sources; no upper bound for the number of Steiner points has been previously known. This structural result allows us to design a *quasi-polynomial time approximation scheme (QPTAS)* for CGND with polynomially bounded demands of sources and sinks. Next, we extend this result to derive our main algorithmic result, a *polynomial-time approximation scheme (PTAS)* for SCGND when the link capacity is at most $2^{O(\sqrt{\log n})}$, and the demands of sources and sinks are polynomially bounded.

The QPTAS is obtained by combining our upper bound on the number of Steiner points with Arora's framework [1] for geometric optimization problems. The main result of the paper, the PTAS, relies on a geometric partition of the sources combined with a TSP-based heuristic and our QPTAS applied to a small number of points.

Related work on geometric network design. A related minimum Euclidean Steiner tree problem has been studied extensively in the literature (cf. [7]), with a PTAS provided independently by Arora [1] and Mitchell [8].

Salman *et al.* [10] initiated the algorithmic study of the single-sink buy-at-bulk network design problem. They argued that the problem is especially relevant in practice in the *geometric* case. They provided a polynomial-time approximation algorithm for a variant of buy-at-bulk single-sink network design on an Euclidean graph with an approximation ratio of $O(\log(D/c_1))$, where D is the total capacity of sources and c_1 is the smallest link capacity. Besides allowing the use of many link types, their model differs from ours in that they did not permit a flow to be split, and only admit a given finite set of points in the plane to be used as vertices by the network, whereas we allow splittable flow and we allow Steiner points to be arbitrary points in the plane. Czumaj et al. [2] considered geometric buy-at-bulk network design allowing the entire space to be used. They presented a QPTAS for the rectilinear variant of geometric network design with polynomially bounded total demand, where the links have to be horizontal or vertical. Their QPTAS relies on the observation that in this case Steiner points can be constrained to lie on the Hanan grid. Hassin et al. [6] and Morsy et al. [9] considered the special case of the network design problem raised in [10], where only one type of link occurs. Since their flow is unsplittable, the result is not comparable to our model.

2 Preliminaries

Consider the Euclidean 2-dimensional space $\mathbb{E}^2$. Let $\{s_1, \ldots, s_{n_s}\}$ be a given set of n_s points in $\mathbb{E}^2$ (*sources*), and let $\{t_1, \ldots, t_{n_t}\}$ be a given set of n_t points in $\mathbb{E}^2$ (*sinks*). Let $n = n_s + n_t$. Each source s_i supplies some integral *demand* $\mathfrak{d}(s_i)$ to the sinks, and each sink t_j is required to receive some integral *demand* $\mathfrak{d}(t_j)$ from the sources. We assume that $\sum_i \mathfrak{d}(s_i) = \sum_j \mathfrak{d}(t_j)$ and define $D = \sum_i \mathfrak{d}(s_i)$ to be the *total demand*. Furthermore, there is given a single *link type* with positive integral *capacity* k.

A *geometric network* is a directed, weighted (finite) *multigraph* embedded in $\mathbb{E}^2$. The *cost* of a geometric network G, denoted $c(G)$, is the sum of the lengths of all links used by G, where in the sum we count all copies of parallel edges (multiedges).

A geometric network G is *feasible* with respect to a given set of sources and sinks with demands, and the link type, if the following conditions hold:

- the vertex set of G contains all sources $\{s_1, \ldots, s_{n_s}\}$, all sinks $\{t_1, \ldots, t_{n_t}\}$, and potentially other vertices (called *Steiner vertices*),
- each copy of a multiedge in G is a link of capacity k,
- there is a flow in G from the sources to the sinks that is consistent with the directions of used links, saturates the demands of the sources and the sinks, and never exceeds the capacity k in a link.

If G is feasible then one can find such a flow in polynomial time using standard algorithms for the maximum-flow problem with multiple sources and multiple sinks. Furthermore, since the demands and the link capacity are assumed to be integral, we can presume, without loss of generality, that the flow found is integral.

The objective of the *capacitated geometric network design (**CGND**) problem* is to construct a feasible geometric network that minimizes the total length of all links (i.e., all copies of multiedges) used. If the set of sinks is a singleton then the problem is termed as the *single-sink capacitated geometric network design (**SCGND**) problem*.

3 Bounding the Number of Steiner Points in Optimal Solution

In this section we consider the CGND problem in $\mathbb{E}^2$ with sources and sinks having positive integral demands. If multiple sources or sinks are in the same location, we merge them to create a single source or sink. We shall show that the number of Steiner points in an optimal solution can be upper bounded by a polynomial function of D.

For this purpose, we will consider a special class of multigraphs that are feasible solutions to the CGND problem, which we call *minimizers*. We will show that for any $\varepsilon > 0$ there is a minimizer that gives a $(1+\varepsilon)$-approximation to the CGND instance. We then analyze geometric properties of the minimizers, and show that *each* minimizer has a special geometric structure, namely it can have only three types of Steiner vertices. We then define an operation of shifting a cycle in a minimizer. We show that this operation transforms a minimizer into another minimizer without increasing the cost, can be performed on a minimizer only a finite number of times, and when the operation cannot be performed any more, the resulting minimizer has a small number of Steiner vertices. We get that for any $\varepsilon > 0$ there *exists* a minimizer that gives a $(1+\varepsilon)$-approximation to the CGND instance and has a small number of Steiner vertices. Then we can show that there is an optimal solution which has a small number of Steiner vertices.

We will consider feasible multigraphs together with associate integral flows certifying the feasibility of the multigraphs. For a multigraph $M = (V, E)$ we define the *value* of M as $\mathfrak{val}(M) = \sum_{v \in V}(\deg(v) - 2)$, where the degree of a vertex is counted with multiplicities in the case of multiple edges. Among the solutions with the same cost, we will prefer the one with a smaller value.

Without increasing the cost or the value we can modify the multigraph M and an associated integral flow f in M to satisfy the following properties:

- there are no isolated vertices in M,
- all Steiner vertices in M have degree at least 3 and are contained in the smallest square containing all sources and sinks (to satisfy this property we move each Steiner vertex from outside the square to the closest point on the boundary),
- if two vertices of M are coincident, then there is no edge between them (otherwise we merge them into one vertex),
- each edge of M is used by f,
- the amount of flow entering and leaving each vertex is at most D.

For technical reasons, and without loss of generality, in this section we consider only pairs (M, f) satisfying the above properties. We get that the degree of a vertex is upper-bounded by $2D$. The Steiner vertices have degree at least 3, and the number of non-Steiner vertices is not greater than $2D$ (i.e. they contribute at least $-2D$ to $\mathfrak{val}(M)$), therefore the number of Steiner vertices in M is not greater than $\mathfrak{val}(M) + 2D$. Since $2|E| = \sum_{v \in V} \deg(v)$, we get that $|E| = \frac{1}{2}\,\mathfrak{val}(M) + |V| \leq \frac{3}{2}\,\mathfrak{val}(M) + 4D$.

For any problem instance, $v \in \mathbb{N}$ and ε, let $\mathcal{M}_v^\varepsilon$ denote the set of all multigraphs that together with some flow give a $(1+\varepsilon)$-approximate solution with a value at most v.

Lemma 1. *For any problem instance, $\varepsilon > 0$ and $v \in \mathbb{N}$ we can equip the set $\mathcal{M}_v^\varepsilon$ with a metric δ such that the metric space $(\mathcal{M}_v^\varepsilon, \delta)$ is compact, and the network cost is a continuous function in $(\mathcal{M}_v^\varepsilon, \delta)$.*

Definition 1. *A multigraph M is a* minimizer *if it is a feasible solution to the CGND problem and if there does not exist another feasible solution with a smaller cost and not greater value or with the same cost and a smaller value.*

Lemma 2. *For any $\varepsilon > 0$ and any instance of the CGND problem, there is a $(1+\varepsilon)$-approximate solution which is a minimizer.*

Proof. Consider any multigraph M which gives a $(1+\varepsilon)$-approximate solution for the given instance. It has some finite value v. Let $(\mathcal{M}_v^\varepsilon, \delta)$ be the metric space from Lemma 1. The metric space is compact and the cost is a continuous function in it, therefore there exists a multigraph with a minimum cost in $\mathcal{M}_v^\varepsilon$. The multigraph with a minimum value amongst all multigraphs with the minimum cost in $\mathcal{M}_v^\varepsilon$ is the desired minimizer. □

In the remaining part of this section we will prove the following theorem.

Theorem 1. *For any $\varepsilon > 0$ and an instance of the CGND problem there is a minimizer M which is a $(1+\varepsilon)$-approximate solution and for which $|S| + |\mathcal{Z}| \leq 32D^4 + 4D^2$, where S is the set of Steiner vertices of M and $\mathcal{Z}$ is the set of points on the plane where the edges of M cross without a Steiner vertex.*

Theorem 1, together with Lemma 1, yields the main result of this section.

Theorem 2. *Any instance of the CGND problem has an optimal solution M_{OPT} which satisfies the following properties: the number of Steiner vertices is at most $32D^4 + 4D^2$, there are no two coincident vertices, the degree of each vertex is at most $2D$ and the edges cross only in vertices.*

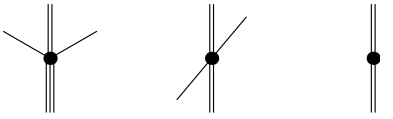


Fig. 2. A real vertex, a crossing and an optional vertex (all for $c = 2$)

3.1 Vertex Types of a Minimizer

We prove here that there can be only three types of Steiner vertices in a minimizer.

Definition 2. *Let $M = (V, E)$ be a multigraph with a flow f. A vertex $v \in V$ is called a* real vertex *if there is an integer $c \geq 0$ and a direction $\alpha \in [0, 2\pi)$ such that:*

- *the edges incident with v are exactly: c edges with the direction α, $c+1$ edges with the direction $\alpha + \pi$, and two single edges with the directions $\alpha \pm \pi/3$ (see Fig. 2),*
- *the group of $c+1$ edges is directed in one way (out of the vertex or into the vertex), and all the remaining edges are directed in the opposite way,*
- *the flow in any of the single edges is smaller than the flow in any of the $c+1$ edges.*

Definition 3. *Let $M = (V, E)$ be a multigraph with a flow f. A vertex $v \in V$ is called a* crossing *if there is integer $c \geq 1$ and directions $\alpha, \beta \in [0, 2\pi)$ s.t. $\beta \neq \alpha, \alpha + \pi$ and:*

- *the edges incident with v are exactly: c edges with the direction α, c edges with the direction $\alpha + \pi$, and single edges with the directions β and $\beta + \pi$ (see Fig. 2),*
- *all the c edges in one group are directed in one way, the directions of the two groups of edges are opposite, the directions of the two single edges are opposite,*
- *if $c \geq 2$ then the flow in any single edge is smaller than the flow in any of the non-single edges with the opposite direction.*

Definition 4. *Let $M = (V, E)$ be a multigraph with a flow f. A vertex $v \in V$ is called an* optional vertex *if there is an integer $c \geq 2$ and a direction $\alpha \in [0, 2\pi)$ such that:*

- *the edges incident with v are exactly: c edges with the direction α and c edges with the direction $\alpha + \pi$ (see Fig. 2),*
- *all the c edges in one group are directed in one way, the directions of the two groups of edges are opposite.*

We specify some properties which the Steiner vertices of a minimizer must fulfill.

Property 1. Let $M = (V, E)$ be a minimizer, $v \in V$, and $e_1, e_2 \in E$ be edges incident with v and directed in opposite ways. Then the angle between e_1 and e_2 is at least $2\pi/3$.

Property 2. Let $M = (V, E)$ be a minimizer with a flow f. Let $v \in V$ and let $e_1, e_2 \in E$ be edges incident with v and directed in the same way. If the angle between e_1 and e_2 is smaller than $2\pi/3$, then $f(e_1) + f(e_2) > k$.

Property 3. Let $M = (V, E)$ be a minimizer with a flow f. Let $v \in V$ and let $e_1, e_2, e_3 \in E$ be edges incident with v contained in an open halfplane with the borderline passing through v. If e_1 and e_2 are directed in the same way and opposite to the direction of e_3, then $f(e_1)+f(e_2)-f(e_3) > k$ and, in particular, $f(e_3) < f(e_1), f(e_2)$.

For a vertex v we call an edge e incident with v *incoming* (*outgoing*) if it is directed towards v (out of v). Using Properties 1–3 we can show the following lemma:

Lemma 3. *Let $M = (V, E)$ be a minimizer. If $v \in V$ is a Steiner vertex such that both incoming and outgoing edges have multiple directions, then v is a crossing. On the other hand, if $v \in V$ is a Steiner vertex such that all outgoing (or incoming) edges have the same direction $\alpha \in [0, 2\pi)$, then v is a real vertex or an optional vertex.*

This immediately implies the following theorem.

Theorem 3. *Each Steiner vertex in a minimizer is a real vertex, a crossing or an optional vertex.*

3.2 Graph Analysis and Cycle Argument

From Theorem 3 we know that a Steiner vertex in a minimizer is a real vertex, a crossing or an optional vertex. We can easily remove optional vertices from a minimizer.

Lemma 4. *For any minimizer M there is a corresponding minimizer M' with the same cost, value, number of real vertices and crossings and with no optional vertices.*

In this section we will prove Theorem 1. We introduce a procedure modifying minimizers to decrease the number of real vertices. We also show, that if we cannot decrease the number of real vertices any further, the total number Steiner vertices must be small.

Definition 5. *Let M be a minimizer without optional vertices. A cycle $\mathcal{C}$ in M is called a* Steiner cycle *if:*

- *$\mathcal{C}$ passes only through Steiner vertices,*
- *if $\mathcal{C}$ passes through a crossing vertex v, then $\mathcal{C}$ does not change direction in v,*
- *if $\mathcal{C}$ passes through a real vertex v, then $\mathcal{C}$ either does not change the direction, or changes it by $\pi/3$,*
- *$\mathcal{C}$ does not pass through an edge more than once.*

Lemma 5. *Let M be a minimizer without optional vertices, S the set of Steiner vertices of M and $\mathcal{Z}$ the set of points on the plane where the edges of M cross without a Steiner vertex. If M has no Steiner cycle, then $|S| + |\mathcal{Z}| \leq 32D^4 + 4D^2$.*

We introduce an operation of *shifting* a Steiner cycle $\mathcal{C}$ by a distance $\Delta > 0$ in a minimizer M with no optional vertices. We orient $\mathcal{C}$ in one of the two possible directions and shift each edge of $\mathcal{C}$ by a distance Δ to the left according to the direction chosen. The real vertices where $\mathcal{C}$ changes direction are moved to the intersection points of the shifted edges (see Fig. 3). A Steiner cycle has no repeating edges, so we never have to shift an edge in two directions at once.

Moving the real vertices as described above requires us to make some other modifications to the graph. We do not want to shift or change directions of edges not belonging to $\mathcal{C}$ — they can only become longer or shorter. To achieve this, in some cases instead of moving a real vertex we split the vertex into a real vertex of degree 3 and a crossing, and then only move the vertex of degree 3 to the required position. We also move the

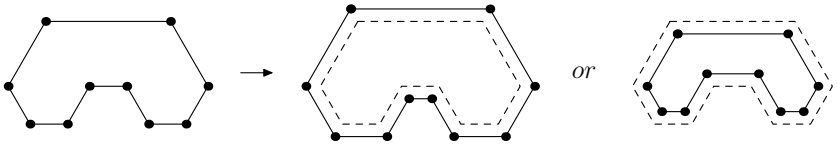


Fig. 3. Shifting a Steiner cycle. Only the real vertices on which the cycle changes direction have been pictured. The obtained cycle depends on the orientation of the original cycle.

real vertices and crossings of $\mathcal{C}$, on which $\mathcal{C}$ does not change the direction, possibly after splitting the vertices in two. The case analysis shows that shifting a Steiner cycle neither changes the cost and the value of the minimizer nor the number of the real vertices. The resulting graph has a feasible flow and is a minimizer.

When a vertex from $\mathcal{C}$ hits a vertex it is incident with, we merge the two vertices. One can show that the shifting operation can be performed until one of the following happens: a vertex from $\mathcal{C}$ gets merged with a non-Steiner vertex or two real vertices get merged (e.g. when a cycle edge gets contracted to a single point). The resulting multigraph has either a smaller number of real vertices, or the same number of real vertices and a larger sum of degrees of non-Steiner vertices.

Proof (of Theorem 1). Fix an $\varepsilon > 0$. By Lemma 2, there is a $(1+\varepsilon)$-approximate solution which is a minimizer. Let M_0 be such a minimizer that also minimizes the number of real vertices — we cannot decrease the number of real vertices of M_0 without increasing the cost or the value of M_0. Let M_1 be the minimizer obtained from M_0 by removing the optional vertices, as in Lemma 4. The number of real vertices, the cost and the value of M_1 is the same as in M_0.

Suppose that M_1 has a Steiner cycle. We shift the cycle as far as possible. As the result, we obtain a minimizer with the same cost and value, where the shifted cycle is not a Steiner cycle any more, and the resulting multigraph has the same number of real vertices and a larger sum of degrees of non-Steiner vertices (any other result of shifting a cycle would decrease the number of real vertices, and that cannot happen for M_1).

As long as M_1 has some Steiner cycle left, we shift it as far as possible. We can perform this operation only a finite number of times, as each time the sum of the degrees of the non-Steiner vertices increases, and in a minimizer it is upper bounded by $4D^2$. At some point there will be no Steiner cycle left. The graph obtained from M_1 will be a minimizer with no Steiner cycles, and from Lemma 5 we know that in such a minimizer $|S| + |\mathcal{Z}| \leq 32D^4 + D^2$. □

4 Quasi-Polynomial Time Approximation Scheme

In this section we consider the CGND problem with the total demand $D \leq n^C$ for some constant C. We will design a quasi-polynomial time approximation scheme for the above problem that works for any capacity k. We can assume that $k \leq D$.

From Theorem 2, we know that there is an optimal network $M_{OPT} = (V, E)$ that satisfies the following properties: the number of Steiner vertices is at most $32D^4+4D^2$, there are no two coincident vertices, the degree of each vertex is at most $2D$ and the

edges cross only in vertices. We have $|V| \leq 38n^{4C}$, $|E| \leq 38n^{5C}$. These bounds make it possible to use the Arora's framework [1] for the TSP problem. Due to lack of space, we give only an overview of the algorithm.

Dividing into subproblems. We can prove the following lemma.

Lemma 6. *In polynomial time, we can partition the input points such that every optimal solution consists of the disjoint solutions for the points from each partition set, and the cost of the optimal solution for the points from any partition set P_i is at least $\frac{\ell_i}{nk}$, where ℓ_i is the side length of the smallest square containing all points from P_i.*

We divide the original problem instance into a collection of at most n independent instances, as given by Lemma 6. In the rest of this section we can therefore assume that all the input points are inside some square of size $\ell \times \ell$, and $c(M_{OPT}) \geq \frac{\ell}{nk}$.

Perturbation. We divide the square $\ell \times \ell$ into *unit squares* using a uniform grid of size polynomial in n and D. We modify the problem by shifting all input points to the nearest gridpoint (intersection of grid lines), and merging coincident sinks and sources.

Let M'_{OPT} be a multigraph obtained from M_{OPT} by this operation. Notice that the edges in M'_{OPT} cross only in Steiner vertices, similarly as the edges of M_{OPT}. Using the upper bounds on the number of vertices and edges in M_{OPT} we can show:

Lemma 7. *If the size of the grid is at least $\frac{80}{\varepsilon} \cdot n^{6C+1} \times \frac{80}{\varepsilon} \cdot n^{6C+1}$, we have $c(M'_{OPT}) \leq (1+\varepsilon) \cdot c(M_{OPT})$.*

Randomized dissection and portals. We make a randomized dissection of the square — a recursive partitioning of the square into four equal squares, performed until we obtain the unit squares. The size of the grid is the smallest power of 2 which is at least $2 \cdot \frac{80}{\varepsilon} \cdot n^{6C+1}$. Also, instead of starting with the original $\ell \times \ell$ square, we start with a $2\ell \times 2\ell$ square with the original square shifted randomly inside. A more detailed description of the randomized dissection can be found in [1,2]. The number of levels of the dissection is $\mathfrak{L} = O(\log(2 \cdot \frac{80}{\varepsilon} \cdot n^{6C+1})) = O(C \cdot \log \frac{n}{\varepsilon})$.

Similarly as in [1,2], we create $m = \frac{2\mathfrak{L}}{\varepsilon} = O(\frac{C}{\varepsilon} \log \frac{n}{\varepsilon})$ equidistant portals on the boundary of each square in such a way that the portals are in all corners of the square, where m is a power of 2. We modify the solutions to the CGND problem in such a way, that the edges can cross the boundaries of the squares only in portals (i.e. we create additional Steiner points in portals). Notice that all the input points are in portals. We do not allow any Steiner points outside of portals, and we do not allow the edges to cross outside of the portals. In the remaining part of this section we shall present a quasi-polynomial time algorithm that finds an optimal solution to the modified problem. The QPTAS follows, as the following hold:

Lemma 8. *The optimal solution to the modified problem has expected cost at most $(1+\varepsilon)^2 \cdot c(M_{OPT})$.*

Lemma 9. *From the optimal solution to the modified problem we can get a solution for the original problem with an expected cost at most $(1+\varepsilon)^3 \cdot c(M_{OPT})$.*

Dynamic programming. We want to find an optimal solution to the modified problem, i.e., where all sinks and sources are in portals, the Steiner points can be created only in portals and the edges do not cross outside of the portals. We can consider only solutions (multigraphs with a specified flow) where: the flow through each portal goes in one direction only (into or out of the square) and the amount of it is at most D.

We find the optimal solution using dynamic programming. The interface of a square specifies the amount of flow and the direction of it (into or out of the square) for each portal. The number of interfaces of a square is $(1+2D)^m = n^{O(C^2\varepsilon^{-1}\cdot\log(n/\varepsilon))}$, which is quasi-polynomial in n. For each square and for each interface we want to find the solution with the minimum cost satisfying the interface (if there is a feasible solution for a given interface). Since we have doubled the size of the original square enclosing the input points, we may assume that no input points are on the perimeter of this square. Hence, the solution to the problem is the minimum cost solution for the doubled original square with an empty interface. We start with the unit squares:

Lemma 10. *The minimum cost solutions to the leaf-subproblems, i.e., to the interfaces of the unit squares can be found in time* $n^{O(C^2\varepsilon^{-1}\log(n/\varepsilon))}$.

The idea of the proof is as follows. We consider only solutions where edges do not cross inside the squares, i.e. they go along some triangulation of the square, where the vertices of the triangulated polygon are the portals. For each triangulation we consider multigraphs consistent with the triangulation, compute their cost and check whether the flow required by the interface can be transferred trough the multigraph.

Lemma 11. *The minimum cost solutions to the remaining subproblems, i.e., to the interfaces of the non-unit squares can be found in time* $n^{O(C^2\varepsilon^{-1}\log(n/\varepsilon))}$.

By applying Lemmas 9–11, we obtain our main theorem in this section.

Theorem 4. *For any* $\varepsilon > 0$*, there is a randomized* $n^{O(C^2\varepsilon^{-1}\log(n/\varepsilon))}$*-time algorithm for capacitated geometric network design with a total of* n *sources and sinks and total demand upper bounded by* n^C*, which yields a solution whose expected cost is within* $(1+\varepsilon)$ *of the optimum. The algorithm can be derandomized similarly as those in [1,3].*

Remark 1. An analogous theorem holds when instead of the requirement on the total demand of sinks balancing the demand of sources, the sinks have unlimited demand.

5 Polynomial-Time Approximation Scheme for Single Sink

In this section we consider the SCGND problem, with the input consisting of a single sink s and a set P of n sources with a total demand $D \le n^C$. The capacity of the edges is k, and throughout this section we will assume that $k \le 2^{O(\sqrt{\log n})}$. We will construct a polynomial-time approximation scheme for this problem.

The idea of the algorithm is as follows. We partition the set of sources into two subsets, depending on their distance from the sink. The "outer" set has a total demand upper bounded by a polynomial function of k. Using the algorithm from Section 4 we construct a near-optimal network which allows to transfer the flow from the outer

sources to some Steiner vertices closer to the sink. The Steiner vertices now become new sources. As the number of points and the total demand is much smaller than n, the running time is polynomial in n. We then solve the problem for the "inner" sources together with the new sources. We show that for such a set of points the cost of the minimum TSP tour is small compared to the cost of the optimal solution. That allows us to use a simple algorithm to find a near-optimal solution. Combining the two networks together gives a near-optimal solution to the SCGND problem.

Let L be maximum distance between a source and the sink, i.e. $L = \max_{p \in P} d(p, s)$. Let OPT be an optimal solution (network) for the SCGND problem, and TSP be the shortest traveling salesman tour for $P \cup \{s\}$. The following lemma can be proved similarly as Theorem 1 in [5] for the capacitated vehicle routing problem.

Lemma 12. *There is a polynomial time algorithm that outputs a solution with cost at most* $(1 + \varepsilon) \cdot c(\mathsf{TSP}) + \frac{1}{k} \sum_{p \in P} d(p, s) \cdot \mathfrak{d}(p)$. *Furthermore, the following hold:*

$$\frac{1}{k} \sum_{p \in P} d(p, s) \cdot \mathfrak{d}(p) \leq c(\mathrm{OPT}) \leq c(\mathsf{TSP}) + \frac{1}{k} \sum_{p \in P} d(p, s) \cdot \mathfrak{d}(p) \ .$$

If $c(\mathsf{TSP}) < \frac{\varepsilon}{k} \sum_{p \in P} d(p, s) \mathfrak{d}(p)$, then Lemma 12 gives a polynomial-time $(1 + \varepsilon)$-approximation. Therefore from now on we assume that $c(\mathsf{TSP}) \geq \frac{\varepsilon}{k} \sum_{p \in P} d(p, s) \mathfrak{d}(p)$.

Working with this assumption, we use the following lemma to upper bound the sum of the distances from the sources to the sink:

Lemma 13. *If* $c(\mathsf{TSP}) \geq \frac{\varepsilon}{k} \sum_{p \in P} d(p, s) \cdot \mathfrak{d}(p)$ *then there is a constant* $\mathfrak{c}$ *such that*

$$\sum_{p \in P} d(p, s) \cdot \mathfrak{d}(p) \leq \left(\frac{\mathfrak{c} \cdot k}{\varepsilon} \right)^2 \cdot L \ .$$

Consider a closed disk $\mathcal{Q}$ of radius αL with the center at the sink s, where we set $\alpha = \frac{\varepsilon^4}{\mathfrak{c}^4 \cdot k^2}$. Let P_{out} be the subset of points from P that are outside the disk $\mathcal{Q}$ (i.e., for which $d(p, s) > \alpha L$), and let $P_{in} = P \setminus P_{out}$. By Lemma 13, we have $\sum_{p \in P_{out}} d(p, s) \cdot \mathfrak{d}(p) \leq (\frac{\mathfrak{c} \cdot k}{\varepsilon})^2 \cdot L$, and hence $\sum_{p \in P_{out}} \mathfrak{d}(p) \leq (\frac{\mathfrak{c} \cdot k}{\varepsilon})^2 \cdot \frac{1}{\alpha}$.

We partition the optimal network into two parts: OPT_{in} and OPT_{out}, which consist of the edges lying respectively inside and outside $\mathcal{Q}$ (we add artificial Steiner points on the boundary of $\mathcal{Q}$ to divide the edges). Then $c(\mathrm{OPT}) = c(\mathrm{OPT}_{in}) + c(\mathrm{OPT}_{out})$.

Lemma 14. *In time* $k^{O(\varepsilon^{-1} \log k)}$ *we can find a set* P'_{out} *of* $\sum_{p \in P_{out}} \mathfrak{d}(p)$ *points on the boundary of the disk* $\mathcal{Q}$ *and a network of cost at most* $c(\mathrm{OPT}_{out}) + 3 \cdot \varepsilon \cdot c(\mathrm{OPT})$ *that allows sending all the flow from* P_{out} *to the unit sinks* P'_{out}.

The idea of the proof is as follows. We create equidistant points (portals) on the boundary of the disk $\mathcal{Q}$ and treat them as sinks with infinite capacity. If the number of the portals is large enough (but still smaller than the total demand of the points from P_{out}), the cost of an optimal network which allows to send the flow from the sources P_{out} to the created sinks is not greater than $c(\mathrm{OPT}_{out}) + \varepsilon \cdot c(\mathrm{OPT})$. A good approximation of such a network can be found by the algorithm from Theorem 4 adapted to Remark 1.

Lemma 15. *In polynomial time we can find a network which sends the flow from the points $P_{in} \cup P'_{out}$ to the sink s with cost at most $c(\mathrm{OPT}_{in}) + 2 \cdot \varepsilon \cdot c(\mathrm{OPT})$.*

One can prove it by showing that the algorithm from the Lemma 12 for the points $P_{in} \cup P'_{out}$ gives the desired approximation ratio. Combining the algorithms from Lemmas 12, 14 and 15 gives us a solution to the original SCGND problem with cost at most $(1 + 5\,\varepsilon) \cdot c(\mathrm{OPT})$ and running in time $k^{O(\varepsilon^{-1} \log k)} + n^{O(1)}$. This gives our main result.

Theorem 5. *For any $\varepsilon > 0$, there is a deterministic algorithm for the SCGND problem with total demand polynomial in n and edge capacity k, which runs in time $k^{O(\varepsilon^{-1} \log k)} + n^{O(1)}$ and yields a solution whose cost is within $(1 + \varepsilon)$ of the optimum. For $k = 2^{O(\sqrt{\log n})}$, it runs in polynomial time.*

6 Final Remarks

It is an intriguing question of whether or not our upper bound on the number of Steiner points in an optimal solution to CGND could be extended to include several types of links. Such an extension would lead to corresponding extensions of our QPTAS for CGND and our PTAS for SCGND. Also, the questions if one can get a PTAS for SCGND for larger k and a PTAS for CGND for some values of k are interesting.

References

1. Arora, S.: Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. Journal of the ACM 45(5), 753–782 (1998)
2. Czumaj, A., Czyzowicz, J., Gąsieniec, L., Jansson, J., Lingas, A., Zylinski, P.: Approximation algorithms for buy-at-bulk geometric network design. In: Dehne, F., Gavrilova, M., Sack, J.-R., Tóth, C.D. (eds.) WADS 2009. LNCS, vol. 5664, pp. 168–180. Springer, Heidelberg (2009)
3. Czumaj, A., Lingas, A.: On approximability of the minimum-cost k-connected spanning subgraph problem. In: Proc. 10th SODA, pp. 281–290 (1999)
4. Garey, M.R., Johnson, D.S.: Computers and Intractability. A Guide to the Theory of NP-completeness. W.H. Freeman and Company, New York (1979)
5. Haimovich, M., Rinnooy Kan, A.H.G.: Bounds and heuristics for capacitated routing problems. Mathematics of Operation Research 10(4), 527–542 (1985)
6. Hassin, R., Ravi, R., Salman, F.S.: Approximation algorithms for a capacitated network design problem. Algorithmica 38, 417–431 (2004)
7. Hwang, F.K., Richards, D.S., Winter, P.: The Steiner Tree Problem. Annals of Discrete Mathematics, vol. 53. North-Holland, Amsterdam (1992)
8. Mitchell, J.S.B.: Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric TSP, k-MST, and related problems. SIAM Journal on Computing 28(4), 1298–1309 (1999)
9. Morsy, E., Nagamochi, H.: Approximation to the minimum cost edge installation problem. In: Tokuyama, T. (ed.) ISAAC 2007. LNCS, vol. 4835, pp. 292–303. Springer, Heidelberg (2007)
10. Salman, F.S., Cheriyan, J., Ravi, R., Subramanian, S.: Approximating the single-sink link-installation problem in network design. SIAM J. Optimization 11(3), 595–610 (2000)
11. Zachariasen, M.: A catalog of Hanan grid problems. Networks 38(2), 76–83 (2001)

An $O(\log n)$-Competitive Algorithm for Online Constrained Forest Problems

Jiawei Qian* and David P. Williamson**

School of Operations Research and Information Engineering,
Cornell University, Ithaca, NY 14853, USA
jq35@cornell.edu, dpw@cs.cornell.edu

Abstract. In the generalized Steiner tree problem, we find a minimum-cost set of edges to connect a given set of source-sink pairs. In the online version of this problem, the source-sink pairs arrive over time. Agrawal, Klein, and Ravi [1] give a 2-approximation algorithm for the offline problem; Berman and Coulston [3] give an $O(\log n)$-competitive algorithm for the online problem. Goemans and Williamson [4] subsequently generalized the offline algorithm of Agrawal et al. to handle a large class of problems they called *constrained forest problems*, and other problems, such as the prize-collecting Steiner tree problem. In this paper, we show how to combine the ideas of Goemans and Williamson and those of Berman and Coulston to give an $O(\log n)$-competitive algorithm for online constrained forest problems, including an online version of the prize-collecting Steiner tree problem.

1 Introduction

Given an undirected graph $G = (V, E)$, edge costs $c_e \geq 0$ for all $e \in E$, and a set of l source-sink pairs s_i-t_i, the goal of the *generalized Steiner tree problem* (also known as the *Steiner forest problem*) is to find a minimum-cost set of edges $F \subseteq E$ such that for each i, s_i and t_i are connected in (V, F). This problem is (as its name implies) a generalization of the *Steiner tree problem*: in Steiner tree problem, we are given an undirected graph with edge costs as above, and also a set $R \subseteq V$ of *terminals*. The goal of the Steiner tree problem is to find a minimum-cost tree T that spans all the terminals R. If we choose one of the terminals $r \in R$ arbitrarily, and set $s_i = r$ for all i and the sink vertices t_i are the remaining vertices in R, then clearly a Steiner tree instance can be expressed as a generalized Steiner tree problem instance. In the 1990s, Agrawal, Klein, and Ravi [1] gave a 2-approximation algorithm for the generalized Steiner tree problem.

At about the same time, online algorithms were being proposed for online versions of the Steiner tree problem, and later, the generalized Steiner tree problem.

* Supported in part by NSF Grant CCF-0830519.

** Part of this work was performed while the author was on sabbatical at TU Berlin, sponsored by the Berlin Mathematical School and a von Humboldt Research Award.

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 37–48, 2011.

In the online version of the Steiner tree problem, terminals arrive over time. At each time step we must give a set of edges F that connects all of the terminals that have arrived thus far; we are not allowed to remove any edges from F in future iterations. The quality of an online algorithm for this problem is measured in terms of its *competitive ratio*: an α-competitive algorithm is one such that at any time step, the set of edges constructed by the algorithm is within a factor of α of the cost of an optimal Steiner tree on the set of terminals that have arrived thus far. Similarly, in the online generalized Steiner tree problem, source-sink pairs arrive in each time step, and we must find a set of edges F such that each s_i-t_i pair that has arrived thus far is connected in (V, F). Imase and Waxman [7] gave a greedy $O(\log n)$-competitive algorithm for the online Steiner tree problem, where $n = |V|$; when a terminal arrives, it finds the shortest path from the terminal to the tree already constructed, and adds that set of edges to its solution. Imase and Waxman also show that the competitive ratio of any online algorithm must be $\Omega(\log n)$. Awerbuch, Azar, and Bartal [2] then showed that a similar greedy algorithm for the online generalized Steiner tree problem has competitive ratio $O(\log^2 n)$. In 1997, Berman and Coulston [3] devised a more complicated algorithm that is an $O(\log n)$-competitive algorithm for the online generalized Steiner tree problem, matching the lower bound of Imase and Waxman to within constant factors.

Also in the 1990s, Goemans and Williamson [4] extended the offline algorithm of Agrawal, Klein, and Ravi to a large class of problems they called constrained forest problems; in doing so, they cast the algorithm of Agrawal et al. as a primal-dual algorithm. A constrained forest problem is defined by a function $f : 2^V \to \{0, 1\}$; for any set $S \subseteq V$ such that $f(S) = 1$, a feasible solution must have selected at least one edge in $\delta(S)$, the set of edges with exactly one endpoint in S. The Goemans-Williamson algorithm works when the function f is *proper*: that is, when $f(S) = f(V - S)$ for all $S \subseteq V$, and for all disjoint sets $A, B \subseteq V$, $f(A \cup B) \leq \max(f(A), f(B))$. For instance, for the case of the generalized Steiner tree problem $f(S) = 1$ if and only if there exists some i such that $|S \cap \{s_i, t_i\}| = 1$, and this function is proper. Another example of constrained forest problems given in [4] is the nonfixed point-to-point connection problem, in which a subset C of the vertices are sources, a disjoint subset D of vertices are destinations, and we must find a minimum-cost set of edges such that each source is connected to a destination so that each connected component has the same number of sources and destinations; this is modelled by having $f(S) = 1$ if $|S \cap C| \neq |S \cap D|$. Yet another example given in [4] is that of partitioning specified vertices D into connected components such that the number of vertices of D in each connected component C is divisible by some parameter k. This problem is given the proper function f such that $f(S) = 1$ if $|S \cap D| \not\equiv 0 (\mathrm{mod}\ k)$.

In this paper, we show that by melding the ideas of Goemans and Williamson with those of Berman and Coulston, we can obtain an $O(\log n)$-competitive algorithm for any *online* constrained forest problem. In an online constrained forest problem, in each time step i we are given a proper function f_i. We must choose a set of edges F such that for all $S \subseteq V$, if $\max_{j=1,\ldots,i} f_j(S) = 1$, then

$|\delta(S) \cap F| \geq 1$. This yields, for example, online algorithms for online variants of the nonfixed point-to-point connection problem and the partitioning problems given above.

Our techniques also extend to give an $O(\log n)$-competitive algorithm for an online version of the prize-collecting Steiner tree problem. In the offline version of the problem, we are given an undirected graph $G = (V, E)$, edge costs $c_e \geq 0$ for all $e \in E$, a root vertex $r \in V$, and penalties $\pi_v \geq 0$ for all $v \in V$. The goal is to find a tree T spanning the root vertex that minimizes the cost of the edges in the tree plus the penalties of the vertices not spanned by the tree; that is, we want to minimize $\sum_{e \in T} c_e + \sum_{v \in V - V(T)} \pi_v$, where $V(T)$ is the set of vertices spanned by T. In the online version of the problem, initially every vertex v has penalty $\pi_v = 0$. At each step in time, for one vertex v its penalty π_v is increased from 0 to some positive value. We then must either connect the vertex to the root by adding edges to our current solution or pay the penalty π_v for each remaining time step of the algorithm even if it is connected to the root later on. The competitive ratio of the algorithm compares the cost of our solution in each step with the cost of the optimal solution of the instance at that point in time.

The basic idea of the Berman-Coulston algorithm (BC) is that it constructs many different families of nonoverlapping balls around terminals as they arrive; in the jth family, balls are limited to have radius at most 2^j. Each family of balls is a lower bound on the cost of an optimal solution to the generalized Steiner tree problem. When balls from two different terminals touch, the algorithm buys the set of edges connecting the two terminals, and balls from one of the two terminals (in some sense the 'smaller' one) can be charged for the cost of the edges, leaving the balls from the other terminal (the 'larger' one) uncharged and able to pay for future connections. One can show that the $O(\log n)$ largest families are essentially all that are relevant for the charging scheme, so that the largest of these $O(\log n)$ families is within an $O(\log n)$ factor of the cost of the constructed solution, thereby giving the competitive ratio. Our algorithm replaces each family of balls with an analogous solution to the dual of the linear programming relaxation of the constrained forest problem, as used by Goemans and Williamson. We need somewhat more complicated dual solutions than the balls used by BC. However, we can then largely follow the outline of the BC analysis to obtain our $O(\log n)$ competitive ratio.

The rest of the paper is structured as follows. In Section 2, we introduce the online constrained forest problem more precisely and define some concepts we will need for our algorithm. In Section 3, we give the algorithm and its analysis. In Section 4, we extend the algorithm and analysis to the prize-collecting Steiner tree problem. We mention some open problems in our conclusion in Section 5.

2 Preliminaries

Given an undirected graph $G = (V, E)$, edges costs $c_e \geq 0$ and a $\{0, 1\}$-proper function $f : 2^V \to \{0, 1\}$, the offine constrained forest problem studied in Goemans and Williamson [4] is to find a set of edges F of minimum cost that satisfies

a connectivity requirement function $f : 2^V \rightarrow \{0,1\}$; the function is *satisfied* if for each set $S \subseteq V$ with $f(S) = 1$, we have $|\delta_F(S)| \geq 1$, where $\delta(S)$ is the set of edges with exactly one endpoint in S, and $\delta_F(S) = \delta(S) \cap F$. In the online version of this problem, we have a sequence of connectivity functions $f_1, f_2, ..., f_i$, arriving one by one. Starting with $F = \emptyset$, for each *time step* $i \geq 1$, function f_i arrives and we need to add edges to F to satisfy function f_i. Let $g_i(S) = \max\{f_1(S), ..., f_i(S)\}$ for all $S \subseteq V$ and $i \geq 1$. Then our goal is to a find a minimum-cost set of edges F that satisfies function g_i, that is, all connectivity requirements given by $f_1, ..., f_i$ that have arrived thus far. We require that each function f_i be a proper function, as defined above. It is easy to see that function g_i is also proper.

Call a vertex v a *terminal* if $f_l(\{v\}) = 1$ for some $l \leq i$. Let $R_i = \{s \in V \mid g_i(\{s\}) = 1\}$ be the set of terminals defined by function g_i; that is, R_i is the set of all terminals that have arrived by time i. A special case of this problem is the online generalized Steiner tree problem where terminal pairs $(s_1, t_1), ..., (s_i, t_i)$ arrive one at a time. In this case, $f_i(S) = 1$ if $|S \cap \{s_i, t_i\}| = 1$ and (s_i, t_i) is the pair of terminals arrive in time step i; then $R_i = \{s_j, t_j : j \leq i\}$. Berman and Coulston [3] give an $O(\log |R_i|)$-competitive algorithm for the online generalized Steiner tree problem.

Let (IP_i) be the integer program corresponding to the online proper constrained forest problem with set of functions $f_1, ..., f_i$ that have arrived thus far and the corresponding function g_i. The integer programming formulation of (IP_i) is

$$
\begin{aligned}
& \text{Min} \sum_{e \in E} c_e x_e & & \\
(IP_i) \qquad & \sum_{e \in \delta(S)} x_e \geq g_i(S), & & \forall S \subseteq V, \\
& x_e \in \{0,1\}, & & \forall e \in E.
\end{aligned}
$$

We let (LP_i) denote the corresponding linear programming relaxation in which the constraints $x_e \in \{0,1\}$ are replaced with $x_e \geq 0$. The dual of this linear program, (D_i), can be described as

$$
\begin{aligned}
& \text{Max} \sum_{S \subseteq V} g_i(S) y_S & & \\
(D_i) \qquad & \sum_{S : e \in \delta(S)} y_S \leq c_e, & & \forall e \in E, \\
& y_S \geq 0, & & \forall S \subseteq V.
\end{aligned}
$$

We now define a number of terms that we will need to describe our algorithm. We will keep an infinite number of feasible dual solutions y^j to bound the cost of edges in F over all time steps; we call this the dual solution for *level* j. For each level j, we will maintain that for any terminal s that has arrived thus far, $\sum_{S \subseteq V : s \in S} y_S^j \leq 2^j$. So we say that the *limit* of the dual in level j is 2^j, and we say that a dual variable y_S^j *reaches its limit* if the inequality for level j is tight for any terminal $s \in S$. As a matter of algorithmic implementation, we don't

need to maintain levels $j < -1$, or $j > \lceil \log(\max_{u,v \in V} d(u,v)) \rceil$, where $d(u,v)$ is the distance in G between u and v using edge costs c_e.

An edge $e \in E$ is *tight* in level j for dual vector y^j if the corresponding constraint in dual problem (D_i), $\sum_{S:e \in \delta(S)} y_S^j \leq c_e$, holds with equality. A path $p \subseteq E$ is *tight* in level j if every edge in the path is tight in level j.

Let $\bar{F}^j$ denote the set of edges that are tight in level j. To avoid confusion with connected components in F, we will use the term *moat* to refer to a connected component S^j in $\bar{F}^j$ and use y_S^j to refer the dual variable associated with S^j.

A set $S \subseteq V$ is a *violated* set for function g_i by edges B if $|\delta_B(S)| < g_i(S)$; that is, if $g_i(S) = 1$ but $\delta_B(S) = \emptyset$. A *minimal violated set* is a violated set with every strict subset not violated. The connectivity requirement function g_i is satisfied by edges B if every set $S \subseteq V$ is not a violated set for g_i by B.

During time step i, a terminal $s \in R_i$ is *active* if for some set S, we have $s \in S$ and S is a violated set for function g_i by current solution F. Let A be the set of active terminals at any time of the algorithm. We define the set of *active* moats as the moats S^j of the lowest level j that satisfy the following three conditions: (i) S^j contains some active terminal $s \in A$; (ii) S^j is a minimum violated set for g_i by edges $\bar{F}^j$; (iii) y_S^j has not yet reached its limit in level j. We denote the current set of active moats by $\mathcal{M}$. Last, we say a dual variable y_S^j is *active* if its corresponding moat S^j is active.

3 The Algorithm and Its Analysis

3.1 The Primal-Dual Online Algorithm

Our algorithm (see Fig. 1) is a dual ascent algorithm in which we grow active dual variables. We say two disjoint moats S_1^j and S_2^j *collide* in level j during our dual growing process if both of them have been active at some point and a path connecting two terminals $s_1 \in S_1^j$ and $s_2 \in S_2^j$ becomes tight in level j. In order for this to happen, at least one of S_1^j and S_2^j must currently be active.

Our algorithm starts with $F = \emptyset$ and $y_S^j = 0$ for all j and all $S \subseteq V$. At the beginning of each time step i, the function f_i arrives and some non-terminal nodes in V may become terminals. We will update active terminal set A and active moat set $\mathcal{M}$. In each time step i, while there are still some active terminals in A, our algorithm will grow uniformly all active dual variables until: (1) an active y_S^j with reaches its limit in level j; (2) an edge $e \in E$ becomes tight in level j; we then add e to $\bar{F}^j$; (3) two disjoint moats S_1^j and S_2^j collide in level j; we then let p be the path connecting two terminals $s_1 \in S_1^j$ and $s_2 \in S_2^j$ in level j, and we build path p in F if s_1 and s_2 are not yet connected in F, and update the set A of active terminals. At the end of each iteration, we update the set $\mathcal{M}$ of active moats. We output F as the solution for (IP_i).

3.2 The Analysis

Now, we will now state our main theorem and a few lemmas that we need to prove it.

Algorithm

$F = \emptyset$, $\bar{F}^j = \emptyset$ for all j, and $y_S^j = 0$ for all j and $S \subseteq V$
For each $\{0,1\}$-proper function f_i that arrives
    Update active terminals A, and active moats $\mathcal{M}$
    While $|A| > 0$
        Grow uniformly all active dual variables y_S^j until
        1) An active y_S^j with reaches limit in level j
        2) An edge $e \in E$ becomes tight in level j, then
            $\bar{F}^j = \bar{F}^j \cup \{e\}$
        3) Two disjoint moats S_1^j and S_2^j collide in level j, then
            Let $p \subseteq E$ be the corresponding path that becomes tight in level j
            Let s_1 and s_2 be the two terminals connected by p in $\bar{F}^j$
            If s_1 and s_2 are yet connected in F
                $F = F \cup \{p\}$, i.e. build edges $p - F$
                Update A
        Update $\mathcal{M}$

Fig. 1. Primal-Dual Algorithm for Online Proper Constrained Forest Problem

Theorem 1. *Our algorithm gives an $O(\log |R_i|)$ competitive ratio for the online proper constrained forest problem (IP_i).*

Lemma 1. *At the end of time step i of our algorithm, F is a feasible solution to (IP_i) and each dual vector y^j is a feasible solution to (D_i).*

Proof. Our algorithm terminates each time step i when there are no active terminals in A. By definition of active terminals, this implies that there is no violated set for g_i for the solutions F; that is, F is a feasible solution to (IP_i).

We need to show our algorithm always terminates in each time step. Notice that if there are no active moats in $\mathcal{M}$, then there must be no active terminals in A, since there is always a level j for sufficiently large j that conditions (ii) and (iii) are satisfied in the definition of $\mathcal{M}$. Similarly, if there is still an active terminal, there must be an active moat that contains it. Then our algorithm will continue to grow duals at progressively higher levels; eventually all pairs of active terminals must be connected.

By construction of the algorithm each dual solution y^j is feasible for (D_i) since we stop growing a dual y_S^j if it would violate a dual constraint. □

In order to give a bound to the total cost of edges in F, we create an account for each connected component X in F, denoted Account(X). We will define a shadow algorithm to credit potential to accounts as dual grows and remove potential from accounts to pay for building edges. We will show that the total cost of edges in F plus the total unused potential remaining in all accounts is always equal to the sum of all dual variables over all levels, i.e. $\sum_j \sum_S y_S^j$.

We need the following lemma before we describe the shadow algorithm.

Lemma 2. *Any active dual variable y_S^j has a unique connected component X in F that contains all terminals in its corresponding moat S^j.*

Proof. It is sufficient to show that all terminals in an active moat S^j are connected in F. Suppose not; then we have terminals s_1 and s_2 both in S^j and not connected in F. Then either s_1 and s_2 have no path in $\bar{F}^j$ connecting them or at least one of s_1 and s_2 was not a terminal when the path in $\bar{F}^j$ connecting s_1 and s_2 became tight. The first case contradicts the fact that y_S^j is active so that S^j must be a moat, i.e. a connected component in $\bar{F}^j$. The second case cannot happen by the construction of our algorithm since we grow duals from lowest levels possible. When s_1 and s_2 became terminals, some level j' with $j' < j$ small enough will have s_1 and s_2 in different moats, and a path p between them becomes tight in some level between j' and $j-1$ by our structure of limits in each level (i.e. dual growth in one level can be no larger than two times of dual growth in one level below). Then path p is built in F and s_1 and s_2 will be connected in F before the algorithm grows dual in level j again. □

Now the shadow algorithm is as follows. First, whenever we grow an active dual variable y_S^j, we will credit the same amount of potential to Account(X), where X is the unique connected component in F that contains all terminals in S^j. Second, whenever the algorithm builds a path p in F connecting two terminal s_1 and s_2, it must be the case that two disjoint moats S_1^j and S_2^j collide in some level j with $s_1 \in S_1^j$ and $s_2 \in S_2^j$ not yet connected in F. Let X_k be connected component in F that contains s_k for $k = 1, 2$. As a result of building edges $p - F$, $X_3 = X_1 \cup X_2 \cup \{p - F\}$ will become a connected component in F. We will merge unused potential remaining in Account(X_1) and Account(X_2) into Account(X_3) and remove potential from Account(X_3) to pay for the cost of building edges in $p - F$.

At any time of the algorithm, for each connected component X, define the class of X to be the highest level j with a dual variable already grown that credits X; we denote this as as Class(X) and sometimes refer to it as the *top level* of X. Define TopGrowth(X) to be the maximum total dual growth of a terminal in X in level Class(X), i.e.

$$\text{TopGrowth}(X) = \max_{s \in X} \{ \sum_{S \subseteq V : s \in S} y_S^{\text{Class}(X)} \text{ and } s \text{ is a terminal} \}.$$

We know that TopGrowth$(X) \leq 2^j$ by dual limit on level j.

Lemma 3. *At any time of the algorithm, the following two invariants hold:*

1. *Every connected component X of F has*

$$\text{Account}(X) \geq 2^{\text{Class}(X)} + \text{TopGrowth}(X);$$

2. $\sum_{e \in F} c_e + \sum_{X \in F} \text{Account}(X) = \sum_j \sum_S y_S^j$.

Proof. Invariant 1 ensures that for a component X, Account(X) stores at least 2^j total potential for each level $j <$ Class(X) plus the maximum total dual growth of a terminal in X at the top level, which gives $2^{\text{Class}(X)-1} + 2^{\text{Class}(X)-2} + ... = 2^{\text{Class}(X)}$ plus TopGrowth(X).

We prove the first invariant by induction on the algorithm. It is easy to see that this invariant holds when no edges are added to F since the algorithm grows dual variables in level j until some active dual variable reaches limit 2^j; it then grows duals in next higher level. Account(X) is credited 2^j for each level below the level Class(X) while getting TopGrowth(X) for current level.

Now, assume invariant 1 holds just before we add edges to F. The algorithm builds a path p in F connecting two terminals s_1 and s_2 only if there are two disjoint moats S_1^j and S_2^j that collide in some level j with $s_1 \in S_1^j$ and $s_2 \in S_2^j$ not yet connected in F. Let X_k be connected component in F that contains s_k for $k = 1, 2$. We know at least one of S_1^j and S_2^j must be active. Without loss of generality, let it be S_1^j. Then we know Class(X_1) $= j$ and Class(X_2) $= j' \geq j$ since we only grow active dual variables in the top level of each component in F. Then by assumption, Account(X_1) $\geq 2^j +$ TopGrowth(X_1) and Account(X_2) $\geq 2^{j'} +$ TopGrowth(X_2). After building edges $p - F$, $X_3 = X_1 \cup X_2 \cup \{p - F\}$ will be a connected component in F. Our shadow algorithm merges the unused potential remaining in Account(X_1) and Account(X_2) into Account(X_3), and removes potential from Account(X_3) to pay for the cost of building edges $p - F$. Since Class(X_3) $= \max\{j, j'\} = j'$, we need to show Account(X_3) $\geq 2^{j'} +$ TopGrowth(X_3).

If $j = j'$, we know TopGrowth(X_1)+TopGrowth(X_2) $\geq \sum_{e \in p} c_e \geq \sum_{e \in p-F} c_e$ since S_1^j and S_2^j collide in level j, and the cost of the path from s_1 to s_2 cannot be more than the total dual containing s_1 and s_2 in level j. Also, TopGrowth(X_3) $\leq 2^j$ by the dual limit on level j. So we have

$$\begin{aligned}\text{Account}(X_3) &= \text{Account}(X_1) + \text{Account}(X_2) - \textstyle\sum_{e \in p-F} c_e \\ &\geq 2^j + \text{TopGrowth}(X_1) + 2^j + \text{TopGrowth}(X_2) - \textstyle\sum_{e \in p-F} c_e \\ &\geq 2^j + 2^j \geq 2^{j'} + \text{TopGrowth}(X_3).\end{aligned}$$

If $j < j'$, we know TopGrowth(X_1) $+ 2^j \geq \sum_{e \in p} c_e \geq \sum_{e \in p-F} c_e$ since S_1^j and S_2^j collide in level j and the cost of the path from s_1 to s_2 cannot be more than the total dual containing s_1 and s_2 in level j. Also, TopGrowth(X_3) $=$ TopGrowth(X_2) since $j < j'$. So, we have

$$\begin{aligned}\text{Account}(X_3) &= \text{Account}(X_1) + \text{Account}(X_2) - \textstyle\sum_{e \in p-F} c_e \\ &\geq 2^j + \text{TopGrowth}(X_1) + 2^{j'} + \text{TopGrowth}(X_2) - \textstyle\sum_{e \in p-F} c_e \\ &\geq 2^{j'} + \text{TopGrowth}(X_3).\end{aligned}$$

Therefore, the invariant 1 holds at any time of the algorithm. Furthermore, since accounts get credited for dual growth and debited exactly the cost of edges in F, we also have that invariant 2 holds at any time of the algorithm. □

Lemma 4. *Let the dual vector y^j with the maximum total dual $\sum_S y_S^j$ be $y^{\max}$. At the end of time step i, we have $\sum_{e \in F} c_e \leq (\log |R_i| + 2) \sum_S y_S^{max}$.*

Proof. By invariant 2 of Lemma 3, $\sum_{e\in F} c_e = \sum_j \sum_S y_S^j - \sum_{X\in F} \text{Account}(X)$. So it suffices to show $\sum_j \sum_S y_S^j - \sum_{X\in F} \text{Account}(X) \le (\log |R_i| + 2) \sum_S y_S^{max}$.

At the end of time step i, let X^* be a connected component in F of highest class and let $m = \text{Class}(X^*)$. We know $\text{Account}(X^*) \ge 2^m$ by invariant 1 of Lemma 3. So the total unused potential remaining in all accounts (that is, $\sum_{X\in F} \text{Account}(X)$) is at least 2^m.

Let $\lambda = \lceil \log_2 |R_i| \rceil$. Each terminal s has total dual $\sum_{S\subseteq V: s\in S} y_S^{-\lambda} \le 2^{-\lambda} \le 1/|R_i|$ in level $-\lambda$ by the dual limit, so that $\sum_S y_S^{-\lambda} \le 1$. Similarly, we have $\sum_S y_S^{m-\lambda-j-1} \le 2^{m-j-1}$. Consider all dual vectors of the form $y^{m-\lambda-j-1}$ with $j \ge 0$; then we have $\sum_{j\ge 0} \sum_S y_S^{m-\lambda-j-1} \le 2^{m-1} + 2^{m-2} + 2^{m-3} + \ldots = 2^m \le \sum_{X\in F} \text{Account}(X)$.

Then, if we consider the dual solutions $y^{m-\lambda}, \ldots, y^m$, we have

$$\sum_j \sum_S y_S^j - \sum_{X\in F} \text{Account}(X) \le \sum_{k=0}^{\lambda} \sum_S y_S^{m-\lambda+k} \le (\log |R_i| + 2) \sum_S y_S^{max}.$$

The lemma follows. □

Now, we are ready to prove Theorem 1.

Proof of Theorem 1: By Lemma 1, at the end of time step i of the algorithm, F is a feasible solution to (IP_i). We have

$$\begin{aligned} \sum_{e\in F} c_e &= \sum_j \sum_S y_S^j - \sum_{X\in F} \text{Account}(X) && \text{by Lemma 3} \\ &\le (\log |R_i| + 2) \sum_S y_S^{\max} && \text{by Lemma 4} \\ &\le (\log |R_i| + 2) OPT_i && \text{by Lemma 1} \end{aligned}$$

where OPT_i is the optimal value of (IP_i) and the last inequality follows by the fact that the cost of a feasible dual solution to (D_i) is a lower bound on OPT_i. Therefore, our algorithms is an $O(\log |R_i|)$-competitive algorithm for the online proper constrained forest problem. Note that we have $|R_i| = O(n)$, where is n is the number of nodes in G. □

4 The Online Prize-Collecting Steiner Tree Problem

Define the online prize-collecting Steiner tree problem as follows: we are given a root node r in G, and a penalty of zero for each non-root node. In each time step i, a terminal $s_i \ne r$ arrives with a new penalty $\pi_i > 0$. We have a choice to either connect s_i to root r or pay a penalty π_i for not connecting it (for time step i and each future time step); in the latter case, we mark the terminal. Our goal is to find a set of edges F that minimizes the sum of edge costs in F plus sum of penalties for all marked terminals.

The integer programming formulation of (IP_i) is

$$
\begin{aligned}
& \textsf{Min} \sum_{e \in E} c_e x_e + \sum_{s_l \in R_i} \pi_l z_l & \\
(IP_i) \qquad & \sum_{e \in \delta(S)} x_e + z_l \geq 1, & \forall S \subseteq V - \{r\}, s_l \in S \cap R_i, \\
& x_e \in \{0,1\}, & \forall e \in E, \\
& z_l \in \{0,1\}, & \forall s_l \in R_i.
\end{aligned}
$$

Let (LP_i) denote the corresponding linear programming relaxation in which the constraints $x_e \in \{0,1\}$ and $z_l \in \{0,1\}$ are replaced with $x_e \geq 0$ and $z_l \geq 0$. The dual of this linear program, (D_i), is

$$
\begin{aligned}
& \textsf{Max} \sum_{S \subseteq V - \{r\}} y_S & \\
(D_i) \qquad & \sum_{S : e \in \delta(S)} y_S \leq c_e, & \forall e \in E, \\
& \sum_{S \subseteq U} y_S \leq \sum_{s_l \in U \cap R_i} \pi_l, & \forall U \subset V, r \notin U, \\
& y_S \geq 0, & \forall S \subseteq V - \{r\}.
\end{aligned}
$$

For the each dual problem (D_i), call the constraints of type $\sum_{S : e \in \delta(S)} y_S \leq c_e$ the edge cost constraints and call the constraints of type $\sum_{S \subseteq U} y_S \leq \sum_{s_l \in U \cap R_i} \pi_l$ the penalty constraints. A penalty constraint corresponding to a set U^j is *tight* in level j if the left-hand side of the inequality is equal to the right-hand side.

A terminal is *active* during time step i if it is *unmarked* and it is not yet connected to root r in current solution F. A terminal is *marked* by our algorithm if we decide to pay its penalty. A moat S^j is *active* during time step i, if it is on the lowest level j that satisfies the following three conditions: (i) S^j contains some active terminal $s \in A$; (ii) y_S^j has not yet reached its limit in level j; (iii) S^j does not contain root r. We denote the current set of active moats by $\mathcal{M}$.

The rest of the definitions follow as in the main algorithm.

We extend our main algorithm to give an $O(\log |R_i|)$-competitive algorithm for the prize-collecting Steiner tree problem. For each time step i, a new terminal s_i with penalty π_i arrives. With the modified definitions of active terminals and actives moats, we follow the same lines of main algorithm to grow dual variables, with the same conditions (1)-(3) in that algorithm, but additionally: (4) When a path p connecting a terminal s to root r becomes tight in level j, we buy the path if s and r are not yet connected in F and update active terminal set A; (5) when a penalty constraint corresponding to a set U^j becomes tight in level j, in which case we mark all terminals in U^j to pay their penalties and update active terminal set A. We let a set Q be all the vertices marked by our algorithm in current time step and in previous time steps. At the end of time step i, our algorithm outputs F and the set of terminals Q marked to pay penalties.

Theorem 2. *Our extended algorithm gives an $O(\log |R_i|)$-competitive algorithm for the online prize-collecting Steiner tree problem (IP_i).*

Proof. To bound total edge costs and penalties, we need to bound the cost of edges built by conditions (3), (4) and penalties paid by condition (5).

Consider edges in F. Let P be the set of paths we built in F by condition (4) of the extended algorithm, i.e. paths that connect a component in $F - P$ to root r. Then $F - P$ is the set of edges built by condition (3) of the extended algorithm. By invariant 2 of Lemma 3, we have $\sum_{e \in F-P} c_e = \sum_j \sum_S y_S^j - \sum_{X \in F-P} \text{Account}(X)$. By condition (4) of the algorithm, for each path p that connects a terminal s to root r, there must be a moat S^j such that $\sum_{e \in p} c_e \leq \sum_{S' \subseteq S^j : s \in S'} y_{S'}^j$. Let X_p be the component in $F - P$ that connected to root by p. All $S' \subseteq S^j$ with $s \in S'$ must credit $\text{Account}(X_p)$. Also, j must be equal to $\text{Class}(X_p)$ since every terminal in X_p is connected to root r after building p in F; after X_p connects to r, our algorithm does not grow any dual that contains a terminal in X_p. By definition of TopGrowth, we know $\sum_{e \in p} c_e \leq \text{TopGrowth}(X_p)$. Therefore, we have $\sum_{e \in P} c_e \leq \sum_{X_p : p \in P} \text{TopGrowth}(X_p)$ so that

$$\begin{aligned}
\textstyle\sum_{e \in F} c_e &= \textstyle\sum_{e \in F-P} c_e + \sum_{e \in P} c_e \\
&\leq \textstyle\sum_j \sum_S y_S^j - \sum_{X \in F-P} \text{Account}(X) + \sum_{X_p : p \in P} \text{TopGrowth}(X_p) \\
&\leq \textstyle\sum_j \sum_S y_S^j - \sum_{X \in F-P : X = X_p, p \in P} (\text{Account}(X_p) - \text{TopGrowth}(X_p)) \\
&\quad \textstyle- \sum_{X \in F-P : X \neq X_p \forall p \in P} \text{Account}(X) \\
&\leq (\log |R_i| + 2) \textstyle\sum_S y_S^{max}.
\end{aligned}$$

The last inequality follows by same argument as Lemma 4 since for the component X^* in $F - P$ with highest class, whether it has a path that connects it to root or not, $\text{Account}(X^*) - \text{TopGrowth}(X^*) \geq 2^{\text{Class}(X^*)}$ by invariant 1 of Lemma 3.

Next, we use a new copy of dual variables to bound penalties of terminals in Q, i.e. $\sum_{s_l \in Q} \pi_l$. For each dual solution y^j, let $\mathcal{S}^j$ be the set of moats in $\bar{F}^j$ that correspond to a tight penalty constraint. It must the case that $\sum_{s_l \in S^j \cap Q} \pi_l = \sum_{S' \subseteq S^j} y_{S'}^j$ for any $S^j \in \mathcal{S}^j$ by condition (5) of the algorithm. By construction, moats in $\mathcal{S}^j$ cannot overlap, so each dual variable is charged to pay a penalty at most once. Also, since we keep growing same set of dual variables in all time steps, our algorithm continues to pay penalties corresponding to tight penalty constraints over all time steps. To bound the total penalty, we know that a terminal can be marked to pay a penalty only by condition (5), so that

$$\sum_{s_l \in Q} \pi_l \leq \sum_j \sum_{S^j \in \mathcal{S}^j} \sum_{S' \subseteq S^j} y_{S'}^j \leq \sum_j \sum_S y_S^j.$$

Let X^* be a component in F of highest class and let $m = \text{Class}(X^*)$. Consider $\sum_j \sum_S y_S^j \leq 2 \sum_j \sum_S y_S^j - \sum_j \sum_S y_S^j$, by invariant 1 of Lemma 3, we know $\sum_j \sum_S y_S^j \geq \text{Account}(X^*) \geq 2^m$. By similar technique in Lemma 4, let $\lambda = \lceil \log_2 |R_i| \rceil$, we know $\sum_S y_S^{m-\lambda-k-2} \leq 2^{m-k-2}$. Consider all dual vectors of the form $y^{m-\lambda-k-2}$ with $k \geq 0$, we have $2 \sum_{k \geq 0} \sum_S y_S^{m-\lambda-k-2} \leq 2^m$. Then, for dual solutions $y^{m-\lambda-1}, \ldots, y^m$, we have

$$2 \sum_{k=0}^{k=\lambda+1} \sum_{S} y_S^{m-\lambda-1+k} \leq 2(\log |R_i| + 3) \sum_{S} y_S^{max}.$$

Therefore,

$$\begin{aligned} \textstyle\sum_{e \in F} c_e + \sum_{s_l \in Q} \pi_l &\leq [(\log |R_i| + 2) + 2(\log |R_i| + 3)] \textstyle\sum_S y_S^{max} \\ &\leq O(\log |R_i|) OPT_i. \end{aligned}$$

□

5 Conclusion

In the online survivable network design problem, we are given as input an undirected graph and nonnegative edge costs, and in the ith time step, a pair of terminals (s_i,t_i) arrives with a connectivity requirement r_i. One must then augment the current solution so that there are at least r_i edge-disjoint paths between s_i and t_i. It is an interesting open question whether primal-dual algorithms for the offline survivable network design problem (such as those in [8,5]) can be adapted to the online case as we did here. If $r_{\max} = \max_i r_i$, Gupta, Krishnaswamy, and Ravi [6] have recently given an $O(r_{\max} \log^3 n)$-competitive algorithm for the online survivable network design problem, so such an adaptation might be possible.

References

1. Agrawal, A., Klein, P., Ravi, R.: When trees collide: An approximation algorithm for the generalized Steiner problem on networks. SIAM Journal on Computing 24, 440–456 (1995)
2. Awerbuch, B., Azar, Y., Bartal, Y.: On-line generalized Steiner problem. Theoretical Computer Science 324, 313–324 (2004)
3. Berman, P., Coulston, C.: On-line algorithms for Steiner tree problems. In: Proceedings of the 29th Annual ACM Symposium on the Theory of Computing, pp. 344–353 (1997)
4. Goemans, M.X., Williamson, D.P.: A general approximation technique for constrained forest problems. SIAM Journal on Computing 24, 296–317 (1995)
5. Goemans, M., Goldberg, A., Plotkin, S., Shmoys, D., Tardos, E., Williamson, D.: Improved approximation algorithms for network design problems. In: Proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 223–232 (1994)
6. Gupta, A., Krishnaswamy, R., Ravi, R.: Online and stochastic survivable network design. In: Proceedings of the 41st Annual ACM Symposium on the Theory of Computing, pp. 685–694 (2009)
7. Imase, M., Waxman, B.M.: Dynamic Steiner tree problem. SIAM Journal on Discrete Mathematics 4, 369–384 (1991)
8. Williamson, D.P., Goemans, M.X., Mihail, M., Vazirani, V.V.: A primal-dual approximation algorithm for generalized Steiner network problems. Combinatorica 15, 435–454 (1995)

On the Power of Lower Bound Methods for One-Way Quantum Communication Complexity

Shengyu Zhang

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

Abstract. One of the most fundamental questions in communication complexity is the largest gap between classical and quantum one-way communication complexities, and it is conjectured that they are polynomially related for all total Boolean functions f. One approach to proving the conjecture is to first show a quantum lower bound $L(f)$, and then a classical upper bound $U(f) = poly(L(f))$. Note that for this approach to be possibly successful, the quantum lower bound $L(f)$ has to be polynomially tight for all total Boolean functions f.

This paper studies all the three known lower bound methods for one-way quantum communication complexity, namely the Partition Tree method by Nayak, the Trace Distance method by Aaronson, and the two-way quantum communication complexity. We deny the possibility of using the aforementioned approach by any of these known quantum lower bounds, by showing that each of them can be at least exponentially weak for some total Boolean functions. In particular, for a large class of functions generated from Erdös-Rényi random graphs $G(N, p)$, with p in some range of $1/poly(N)$, though the two-way quantum communication complexity is linear in the size of input, the other two methods (particularly for the one-way model) give only constant lower bounds. En route of the exploration, we also discovered that though Nayak's original argument gives a lower bound by the VC-dimension, the power of its natural extension, the Partition Tree method, turns out to be exactly equal to another measure in learning theory called the *extended equivalence query complexity*.

1 Introduction

Communication complexity studies the minimum amount of communication needed to compute a function of inputs distributed over two (or more) parties. Through more than three decades of studies since its invention by Yao [30], it has flourished into a research field with connections to numerous other computational settings such as circuit complexity, streaming algorithms, data structures, decision tree complexity, VLSI design, algorithmic game theory, optimization, pseudo-randomness and so on [19,28].

In the basic two-party setting, Alice and Bob are given inputs $x \in \{0,1\}^n$ and $y \in \{0,1\}^m$, respectively, and they need to compute $f(x, y)$. In the *two-way* model, Alice and Bob are allowed to send messages back and forth; in the one-way

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 49–60, 2011.

model, only Alice sends a message to Bob. The least amount of communication needed for the worst-case input in a deterministic, randomized and quantum protocol, with the latter two allowing a bounded error, is the communication complexity in the corresponding models. We denote by $\mathsf{D}(f)$, $\mathsf{R}(f)$ and $\mathsf{Q}(f)$ the deterministic, randomized and quantum communication complexities of function f in the two way model, and $\mathsf{D}^1(f)$, $\mathsf{R}^1(f)$ and $\mathsf{Q}^1(f)$ the corresponding complexities in the one-way model.

Though much weaker than the two-way model, the one-way model has also caused much attention for various reasons: First, the model is powerful enough to admit many efficient protocols, including both cases for specific functions (such as Equality) and cases for general functions (such as the one with cost in terms of the γ_2^∞-norm [20]). Second, the one-way communication complexity has close connections to areas such as streaming algorithms [22]. Third, proving lower bounds of the one-way communication complexity for general functions turns out to be mathematically quite challenging.

Lower bound methods for communication complexity are of particular interest since in most, if not all, connections to other theoretical areas, communication complexity serves as a lower bound. Quantum lower bounds are interesting for another important reason: One of the most fundamental questions in the field is to pin down the largest gap between classical and quantum communication complexities for total Boolean functions. However, the problem is notoriously hard and our knowledge is very limited: the largest known gap between $Q(f)$ and $R(f)$ for a total function is quadratic (achieved by, for example, Disjointness [15,25,8,3,12,26,27]), while the best upper bound of $R(f)$ in terms of $Q(f)$ is still exponential. The situation in the one-way model is more embarrassing: Despite a lot of efforts [1,2,14], the best separation between classical and quantum one-way communication complexities is a factor of 2 (for Equality function as observed by Winter[29]), while the best upper bound of gap is exponential. Actually, it was highly nontrivial to find even relations [6] or partial functions [10,16] with exponential gaps. Based on this and various other facts such as Holevo's bound, it is reasonable to conjecture that $R^1(f)$ and $\mathsf{Q}^1(f)$ are polynomially related for all total Boolean functions f.

Two approaches were previously taken to understand the relation. One is trying to simulate a quantum protocol directly by a randomized one. Unfortunately, the cost of all classical simulations so far have parameters other than $\mathsf{Q}^1(f)$, and those parameters can be easily as large as n for some total functions. The other natural approach is to firstly derive a general lower bound $\mathsf{Q}^1(f) = \Omega(L(f))$, and then prove a matching upper bound $R^1(f) = poly(L(f))$. Recently Jain, Klauck and Nayak proved that the one-way rectangle bound tightly characterizes $R^1(f)$ [13], which raised the hope of proving the polynomial relation by establishing a matching quantum lower bound. Jain and Zhang tried along this way [14], but were only able to prove that the distributional quantum complexity is at least the distributional rectangle bound for all product distributions.

Note that for this approach to succeed for a general total function f, the tightness of the quantum lower bound $L(f)$ is crucial: If it is not always

polynomially tight, then any attempt on establishing a matching classical upper bound is doomed to fail. There are two methods particularly for the quantum one-way model. One is the *trace distance method* for general functions by Aaronson [1]. The other originates in some sense in the paper [4] by Ambainis, Nayak, Ta-Shma, and Vazirani, and is more explicit in [23] by Nayak for the Random Access Code (RAC) problem; we will refer to this technique as the *partition tree* method (for the reason that will be clear from later discussions). Besides these two methods, the two-way quantum communication complexity $\mathsf{Q}(f)$ can also serve as a lower bound for $\mathsf{Q}^1(f)$ by definition. In this paper, we show that

Theorem 1. *None of the above three known lower bound methods for* $\mathsf{Q}^1(f)$ *is polynomially tight. Actually, they can all be at least exponentially weak.*

This theorem implies that any one of the known methods does not suffice to prove the conjecture that $\mathsf{R}^1(f) = poly(\mathsf{Q}^1(f))$. It can also be viewed as an partial explanation on why the conjecture, if true, is so hard to prove. These negative results on the tightness call for new lower bound methods for $\mathsf{Q}^1(f)$, and we hope that the exhibited weakness of these methods can guide us to search for new and more powerful ones.

Next we discuss in more details about our studies of the various lower bound methods. First, unlike the trace distance method, the partition tree method does not have a well-defined formula for general functions. This paper starts from cleaning up the picture of the method, leading to a robust generalization. As an unexpected connection, its power turns out to be exactly the *extended equivalence query complexity* in computational learning theory. This is interesting compared to that Nayak's original argument actually proves $\mathsf{Q}^1(f) \geq VC(f)$, the VC-dimension of f [18].

We then analyze the power of the three lower bound techniques. Various relations between these techniques are studied, among which the advantage of $\mathsf{Q}(f)$ over the other two methods is particularly interesting. Presumably $\mathsf{Q}(f)$ should not be a good lower bound for $\mathsf{Q}^1(f)$ which in general can be much larger; for example, for Index function $\mathsf{Q}(f) \leq \log_2 n$ but $\mathsf{Q}^1(f) = 1$. However, it turns out that for most functions induced by a random graph $G(N, p)$ for a large range of $p = 1/poly(N)$, both the partition tree method and the trace distance methods can only give a constant lower bound for $\mathsf{Q}^1(f)$, while we can show that $\mathsf{Q}(f) = \Omega(n)$ by the generalized discrepancy method [20].

More related work. On the relation of $\mathsf{R}^1(f)$ and $\mathsf{Q}^1(f)$, if parameters other than $\mathsf{Q}^1(f)$ are allowed, then nontrivial classical upper bounds are known: The aforementioned bound in VC-dimension and Sauer's lemma that $D^1(f) = O(mVC(f))$ imply the upper bound $D^1(f) = O(m\mathsf{Q}^1(f))$. Aaronson later generalized this to partial functions $D^1(f) = O(m\mathsf{Q}^1(f)\log \mathsf{Q}^1(f))$ [1] and $\mathsf{R}^1(f) = O(m\mathsf{Q}^1(f))$ [2]. Jain and Zhang [14] improved the last bound to $\mathsf{R}^1(f) = O((I_\mu(X;Y)+1)VC(f))$ for total functions where $I_\mu(X;Y)$ is the mutual information of the correlated inputs (X, Y) under a hard distribution μ. Klauck gave a variant of Nayak's argument in [17], which unfortunately may be weaker than the VC-dimension (up to an logarithm), while our partition tree bound is always at least $VC(f)$.

There are quite a few results on separations of classical and quantum communication complexities for total functions in the so-called SMP model [7] and for partial functions or relations in various other models [8,24,11,9,16].

2 Preliminaries

Suppose Alice's input set is $\mathcal{X}$ with size $N = 2^n$ and Bob's input set is $\mathcal{Y}$ with size $M = 2^m$. The set of inputs $\{(x, y) : f(x, y) = b\}$ is called b-inputs. For a graph $G = (V, E)$, the function $f_G : V \times V \to \{0, 1\}$ is defined as $f_G(x, y) = 1$ iff $(x, y) \in E$. We assume that $f(x, x) = 0$. For a vertex $v \in V$, its neighbor set is denoted by $N(v)$. An N-node random graph in the Erdös-Rényi model $G(N, p)$ is obtained by connecting each pair of vertices independently with probability p. For a graph G, its adjacency matrix is A_G. For a matrix A, let $\sigma_1(A)$, ..., $\sigma_{rank(A)}(A)$ be the singular values of A in the decreasing order.

The trace distance method was introduced by Aaronson.

Theorem 2 (Aaronson, [1]). *Let $f : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$ be a total Boolean function, and μ is a probability distribution on the 1-input set $\{(x, y) : f(x, y) = 1\}$. Let D_k be the distribution over $(\{0,1\}^n)^k$ formed by first choosing $y \in \mu$ and then choosing k samples independently from the conditional distribution μ_y. Suppose that $\mathbf{Pr}_{x \leftarrow \mu,\ y \leftarrow \mu}[f(x, y) = 0] = \Omega(1)$, where "$x \leftarrow \mu,\ y \leftarrow \mu$" is to draw x and y independently from the two marginal distributions of μ, then $\mathsf{Q}^1(f) = \Omega(\log(1/\|D_2 - D_1^2\|))$.*

Definition 1. *The* trace distance bound *for $\mathsf{Q}^1(f)$ is $\mathsf{TD}(f) = \max_\mu \log_2 \frac{1}{\|D_2 - D_1^2\|}$ where the maximum is taken over all probability distributions μ on the 1-inputs.*

Linial and Shraibman introduced the following lower bound for $\mathsf{Q}(f)$ based on the factorization norm. For a matrix A, define $\gamma_2(A) = \min_{A=BC} \|B\|_{2\to\infty}\|C\|_{1\to 2}$ where for vector norms $\|\cdot\|_X$ and $\|\cdot\|_Y$, the operator norm $\|A\|_{X\to Y} \stackrel{\text{def}}{=} \max_{\|x\|_X=1} \|Ax\|_Y$. For a sign matrix A and $\alpha \geq 1$, let $\gamma_2^\alpha(A) = \min_{B:1\leq a_{ij}b_{ij}\leq\alpha} \gamma_2(B)$.

Theorem 3 (Linial and Shraibman, [20]). $Q_\epsilon(f) \geq \log_2 \gamma_2^{1/(1-2\epsilon)}(f) - O_\epsilon(1)$.

The bound is also known as the *generalized discrepancy method*. The bound actually holds even for $Q^*(f)$, the quantum communication complexity with entanglement shared by Alice and Bob, is lower bounded by the above quantity. Here we are mainly concerned with the case without entanglement because the no-separation conjecture becomes trivial (due to quantum teleportation) if we compare $Q^{1,*}(f)$ and $R^{1,*}(f)$.

Definition 2. *The* ϵ-factorization norm bound *for $Q_\epsilon(f)$ is $\mathsf{FN}_\epsilon(f) = \log_2 \gamma_2^{1/(1-2\epsilon)}(f)$, and the* factorization norm bound *for $Q^*(f)$ is $\mathsf{FN}(f) = \mathsf{FN}_{1/3}(f)$.*

3 The Partition Tree Method

The partition tree bound is defined as follows. Consider a binary *partition tree* $\mathcal{T}$ *of* $\mathcal{X}$, where each node $v = v_1...v_i$ (i is the depth of v) is associated with an input y_v of Bob. Let X be a random variable according to the distribution p over $\mathcal{X}$. This tree induces a subset $\mathcal{X}_v \subseteq \mathcal{X}$ for each node v in the following inductive way: the root corresponds to $\mathcal{X}$, and suppose the set $\mathcal{X}_v$ is defined then the two subsets $\mathcal{X}_{v0}$ and $\mathcal{X}_{v1}$ for its two children $v0$ and $v1$ is defined by $\mathcal{X}_{vb} = \{x \in \mathcal{X}_v : f(x, y_v) = b\}$. A note v is a leaf iff $f(x, y_v)$, for all $x \in \mathcal{X}_v$, have the same value. Define a sequence of random variables $V_1, ..., V_{depth(\mathcal{T})}$ by $\mathbf{Pr}[V_{i+1} = b|V_1...V_i] = p(\mathcal{X}_{V_1...V_ib})/p(\mathcal{X}_{V_1...V_i})$. For a node $v = v_1...v_i$, define $p(v) \stackrel{\text{def}}{=} p(\mathcal{X}_v)$ and $p_v(b) \stackrel{\text{def}}{=} p(\mathcal{X}_{vb}|\mathcal{X}_v)$ for $b \in \{0, 1\}$ and $p_v(\min) \stackrel{\text{def}}{=} \min\{p_v(0), p_v(1)\}$.

Definition 3. *The* partition tree bound *for* $\mathsf{Q}^1(f)$ *is* $\mathsf{PT}(f) = \max_{\mathcal{T},p} \sum_{v\in\mathcal{T}} p(v)p_v(\min)$.

It is not quite immediate to generalize Nayak's argument (for RAC) to this formulation as a lower bound of $\mathsf{Q}^1(f)$. Please see the full version for detailed explanations.

To study $\mathsf{PT}(f)$, first observe that if one can find a balanced binary subtree of height h, then $\mathsf{PT}(f) \geq h(1 - H(\epsilon))$ since one can put half-half probabilities on both branches of each node of the subtree. (Note that this is at least $VC(f)$ but could be much larger than it, as shown in the Greater Than function in the full version.) The following theorem says that this is actually also the best lower bound the partition tree method can provide.

Theorem 4. *There exists a subset* $S \subseteq \mathcal{X}$ *and a partition tree* $\mathcal{T}^*$ *for* f *on* $(S, \mathcal{Y})$ *s.t.* $\mathsf{PT}(f) =$ *the length of the shortest path of* $\mathcal{T}^*$.

See the full version for the proof.

Note that the standard decision tree complexity is to minimize the the length of the longest path, but here the best PT bound is to maximize the length of the shortest path.

It turns out to have an interesting connection to the *extended equivalence query complexity* in learning theory, which we will define using the language of communication complexity as follows. Alice has an input x and Bob wants to exactly learn x by making queries to Alice, who then responses with an answer. Different query models were studied in learning theory.

1. *membership query*: Bob's query is a column y, and Alice's response is $f(x, y)$;
2. *equivalence query*: Bob's query is a string $a \in \{0, 1\}^M$ as a guess of x. If $a = x$, then Alice tells Bob so and the game is over. Otherwise, Alice not only tells Bob that his guess is wrong, but also provides a column y which $f(x, y) \neq a_y$.

   If Bob is restricted to use strings $a \in \{0, 1\}^M$ appearing as rows in the matrix f as queries, then this is called the *equivalence query*; if Bob is allowed to use any string $a \in \{0, 1\}^M$, it is called the *extended equivalence query*.

The minimal number of a particular type of queries Bob needs to make for the worst-case input x is called the query complexity of that type. Denote by $MQ(f)$, $EQ(f)$ and $XEQ(f)$ the membership query complexity, the equivalence query complexity and the extended equivalence query complexity of the function f, respectively. The following theorem gives a characterization of $XEQ(f)$ by relating it to membership query computation.

Theorem 5 (Littlestone, [21]). *$XEQ(f) = \max_{\mathcal{T}} \min_x d(x, \mathcal{T})$, where $\mathcal{T}$ is a membership query computation tree and $d(x, \mathcal{T})$ is the depth of x in $\mathcal{T}$.*

A membership query computation tree is a decision tree with membership queries in the natural way; see the survey [5] for a formal definition (as well as an extensive review of different types of queries in learning theory). Its important relation to our work is that the membership query computation tree is exactly our partition tree, and thus the above theorem and Theorem 4 combined give the following full characterization of PT.

Theorem 6. *$\mathsf{PT}(f) = XEQ(f)$.*

4 Comparisons between the Powers

In this section we will study the power of the lower bound methods, the PT bound part of which uses the limitation result established in the previous section. We will prove Theorem 1 by a circle of comparison results in the order of PT >> Q >> TD >> PT. The first separation is easily exhibited by Index function. Next we will show that though as a lower bound method merely for the two-way complexity, the factorization norm method can be much stronger than the other two methods for the one-way complexity. In fact, for almost all functions f in some range (the precise meaning of which will be clear shortly) the factorization norm gives a linear lower bound for $\mathsf{Q}(f)$ while the other two cannot even prove a super constant lower bound for $\mathsf{Q}^1(f)$. The advantage of FN over TD is given next, and that of FN over PT is given in Section 4.3.

4.1 On the Advantage of the Factorization Norm Method over the Trace Distance Method

In this section we will show that for a random Erdös-Rényi graph $G(N, p)$ for some range of p, we have $\mathsf{FN}(f_G) = \Omega(n)$ but $\mathsf{TD}(f_G) = O(1)$.

Here we consider a random graph $G(N, p)$ since the corresponding limitations for TD and PT are easier to show. By studying the normalized Laplacian operator on the graph, one can show that the factorization norm method gives a good lower bound for most such random graphs.

Theorem 7. *If $\omega(\log^4 N/N) \le p \le 1 - \Omega(1)$, then with probability $1 - o(1)$, an N-node random graph $G(N, p)$ has $\mathsf{FN}(f_G) - O(1) \ge \frac{1}{2}\log_2(pN) - O(1)$.*

The proof is in the full version, from which one can see that actually even the discrepancy bound, a bound weaker than FN, is still at least $\Omega(\log_2(pN)))$.

Next we show that TD can only give a constant lower bound for random graph functions.

Theorem 8. *For $p = o(N^{-6/7})$, a random graph $G(N,p)$ has $\mathsf{TD}(f_G) = O(1)$ with probability $1 - o(1)$.*

Here we are not aiming to maximize the range of p, though we believe that the result still holds for larger p. The main goal is to show the existence of a range $p = 1/poly(N)$. We will first show in the following Lemma 2 that with probability $1 - o(1)$, a random graph $G(N,p)$ has some good properties. The proof uses Lemma 1 which is on the relation of the number of edges and that of vertices with some connection requirement. After these, we will show that for graphs with those properties, the TD bound is very low.

Lemma 1. *For any constant $\delta > 0$, there are constants c and d s.t. for all distinct vertices $V_x = \{x_1, ..., x_c\}$ and $V_z = \{z_1, ..., z_d\}$, if any x_i and z_k share at least one common neighbor, and there is no vertex y connecting to all x_i's and z_k's, then there exists $V_y = \{y_1, ..., y_e\}$, s.t. any pair (x_i, z_k) of vertices are connected via y_j for some $j \in [e]$, and $g/(c+d+e) \geq 4/3 - \delta$, where g is the number of edges between $V_x \cup V_z$ and V_y.*

Proof. For each (x_i, z_k), there is at least one y connecting them. Collect all these y's to form the set V_y. (For pairs (x_i, z_k) with more than one connectors y, we pick an arbitrary one.) Thus each y has degree at least 2, and therefore

$$g/(c+d+e) \geq 2e/(c+d+e). \tag{1}$$

Now we will give another lower bound

$$g/(c+d+e) \geq 1 + (d-2)/(e+6). \tag{2}$$

Combining the two inequalities gives the desired result.

To show the second bound, fix a setting with $g/(e+6)$ minimized. See Figure 1 (where every $N(z_k) \cap V_y$ is the same set simply for convenience of illustration). The way we chose V_y guarantees that $N(V_y)$ contains the whole V_x. Pick a subset $S \subseteq V_y$ with the minimum size *s.t.* the $N(S) \supseteq V_x$. By definition, the number of edges from S to V_x is at least $|V_x| = c$. Define $e_1 = |S|$ and $e_2 = e - |S|$; Condition 2 implies $e_1 \geq 2$. Note that for each z_k, its neighbor set in V_y, *i.e.*

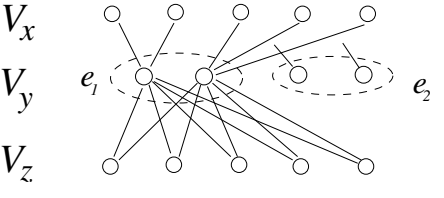


Fig. 1. Illustration for the proof of Lemma 1

$N(z_k) \cap V_y$, also connects to all V_x, thus the number of edges from z_k to V_y is $|N(x_i) \cap V_y| \geq e_1$. Also note that each node in $V_y - S$ has at least one edge to V_x. Thus the total number of edges in this small graph $V_x \cup V_y \cup V_z$ is at least

$$de_1 + c + e_2 = (d-1)e_1 + c + e \geq 2(d-1) + c + e = 1 + (d-2)/(e+6). \quad (3)$$

Lemma 2. *For $p = o(N^{-6/7})$, a random graph $G = G(N,p)$ has all the following properties with probability $1 - o(1)$.*

1. *For any vertex x with (at least) three neighbors y_1, y_2, y_3, at least one of the two pairs (y_1, y_2) and (y_2, y_3) only has x as their common neighbor.*
2. *There are universal constants c and d s.t. for any c vertices $x_1, ..., x_c$ that do not share a common neighbor, there are at most $d-1$ vertices $z_1, ..., z_{d-1}$ which have distance exactly 2 to all x_i's.*
3. *The graph does not contain a $K_{3,2}$, the $(3,2)$-complete bipartite graph, as a subgraph.*

Lemma 3. *Suppose there is a distribution μ on 1-inputs with $\mathbf{Pr}_{x \leftarrow \mu, y \leftarrow \mu}[f(x,y) = 0] = \Omega(1)$ satisfied. If there is a submatrix A s.t. $\mu(A) = 1 - o(1)$, and A as a function has $Q^{1,pub}(A) = q$, then $\|D_2 - D_1^2\|_1 = 2^{\Omega(-q)}$. In particular, $\|D_2 - D_1^2\|_1 = \Omega(1)$ for the following two special cases*

1. *there is a subset $S \subseteq \mathcal{X}$ s.t. $|S| = O(1)$ and $\mu(S) = 1 - o(1)$,*
2. *there is a submatrix A s.t. each column is monochromatic except for at most $O(1)$ entries, and $\mu(A) = 1 - o(1)$.*

See the full version for proofs of the above two lemmas.

Proof. (of Theorem 8) We take the graphs with all good properties in Lemma 2. It is enough to show that any distribution μ on the edge set E with the following condition satisfied

$$\mathbf{Pr}_{x \leftarrow \mu,\ y \leftarrow \mu}[f(x,y) = 0] = \Omega(1), \quad (4)$$

has that $\|D_2 - D_1^2\|_1 = \Omega(1)$. Assuming that it is not true, *i.e.* $\|D_2 - D_1^2\|_1 = o(1)$, we will first show that this assumption forces μ to put most mass on just one star-shape cluster of vertices, then show that in this case, it is also unavoidable to have $\|D_2 - D_1^2\|_1 = \Omega(1)$ finally.

For two vertices x and x', we say x *covers* x', denoted by $x \sim x'$, if they share a common neighbor y. Otherwise we write $x \nsim x'$. For a vertex itself, we assume $x \sim x$ as long as x has a non-zero degree. Define the set $Cover(x) = \{x' : x \sim x'\}$. By definition,

$$\begin{aligned}\|D_2 - D_1^2\|_1 &= \sum_{x,x'} \left| \sum_y \mu(y)\mu(x|y)\mu(x'|y) - \mu(x)\mu(x') \right| \\ &= \sum_{x,x':\ x \sim x'} \left| \sum_y \mu(y)\mu(x|y)\mu(x'|y) - \mu(x)\mu(x') \right| + \sum_{x,x':\ x \nsim x'} \mu(x)\mu(x')\end{aligned}$$

Thus

$$\sum_x \mu(x)\mathbf{Pr}_{x'\leftarrow\mu}[x \nsim x'] = \sum_{x,x':\ x\nsim x'} \mu(x)\mu(x') \leq \|D_2 - D_1^2\|_1 = o(1) \quad (5)$$

This means that an average x covers most other vertices x' (weighted under μ). In particular, there exists one x_0 *s.t.* $\mathbf{Pr}_{x'\leftarrow\mu}[x' \nsim x_0] = o(1)$. Suppose its neighbor set is $N(x_0) = \{y_1, ..., y_t\}$, and define *clusters* $S_i = N(y_i) - \{x_0\}$, and put $S = \cup_i S_i$. Also note that $Cover(x_0)$ is nothing but $S \cup \{x_0\}$, thus $\mu(S \cup \{x_0\}) = 1 - o(1)$. Note that by Lemma 3, we can assume that $\mu(x_0) = 1 - \Omega(1)$ (otherwise the desired conclusion has already been proved). Denote by $\mu_{\neg x_0}$ the distribution $\mu(x)$ conditioned on $x \neq x_0$.

Lemma 4. *At least one of the following two statements is true:*

1. *There are two disjoint subsets G_1 and G_2 of S s.t. $\mu_{\neg x_0}(G_b) = \Omega(1)$ for both $b = 1, 2$, and any two vertices $x_1 \in G_1$ and $x_2 \in G_2$ belong to two different clusters.*
2. *There is a single cluster S_i with $\mu_{\neg x_0}(S_i) = 1 - o(1)$.*

See the full version for a proof. We continue the proof of Theorem 8. If the second statement of the above claim is true, it means that there is a single $y_i \in N(x_0)$ *s.t.* $\mu(N(y_i)) = 1 - o(1)$. (Note that $N(y_i)$ includes a cluster and x_0 itself.) By the third property of Lemma 2, each vertex y other than y_i only connects to at most two vertices in $N(y_i)$ (to avoid a $(3, 2)$-complete bipartite graph). Thus the submatrix on $N(y_i) \times \mathcal{Y}$ has $1 - o(1)$ μ-mass but has all 1's in column y_i and at most two 1's in all other columns. By Lemma 3, we see that $\|D_2 - D_1^2\|_1 = \Omega(1)$.

Therefore, we can assume that the first statement of the claim is the case, so

$$\mathbf{E}_{x\leftarrow\mu_{\neg x_0}}\left[\mathbf{Pr}_{x'\leftarrow\mu}[x' \nsim x]\right] \geq \sum_{b=1,2} \mu_{\neg x_0}(G_b)\mathbf{E}_{x\leftarrow\mu_{\neg x_0}}\left[\mathbf{Pr}_{x'\leftarrow\mu}[x' \nsim x] \;\middle|\; x \in G_b\right].$$

On the other hand, we have

$$\mathbf{E}_{x\leftarrow\mu_{\neg x_0}}\left[\mathbf{Pr}_{x'\leftarrow\mu}[x' \nsim x]\right] = \frac{\sum_{x\neq x_0}\mu(x)\mathbf{Pr}_{x'\leftarrow\mu}[x' \nsim x]}{1 - \mu(x_0)} = \frac{o(1)}{\Omega(1)} = o(1). \quad (6)$$

Since both $\mu_{\neg x_0}(G_b) = \Omega(1)$, we have $\mathbf{E}_{x\leftarrow\mu_{\neg x_0}}\left[\mathbf{Pr}_{x'\leftarrow\mu}[x' \nsim x] \;\middle|\; x \in G_b\right] = o(1)$ for both $b = 1, 2$. Therefore, we can find two points $x_b \in G_b$ both with $\mathbf{Pr}_{x'\leftarrow\mu}[x' \nsim x_b] = o(1)$. This means that for both $b = 1, 2$, most of mass of μ is put on $Cover(x_b)$. Combined with the same fact for x_0, we see that actually $\mu(\cap_{i=0,1,2} Cover(x_i)) = 1 - o(1)$. But note that both x_1 and x_2 are covered by x_0 since they are chosen from S, and they are not in the same cluster as guaranteed by the first statement of the above claim. Consequently, x_0, x_1, x_2 do not share a common neighbor.

Now define set $T = \{x_0, x_1, x_2\}$. As long as $|T|$ is constant, we can assume by Lemma 3 that $\mu(T) = 1 - \Omega(1)$. Then similar to Eq (6), it follows that $\mathbf{E}_{x\leftarrow\mu}[\mathbf{Pr}_{x'\leftarrow\mu}[x' \nsim x] | x \notin T] = o(1)$. Thus there exists another point x in $S - T$

s.t. $\mathbf{Pr}_{x' \leftarrow \mu}[x' \nsim x] = o(1)$. Add this point to T and continue this process until $|T| = c$. Each point $x \in T$ has the property that $\mu(Cover(x)) = 1 - o(1)$, and consequently $\mu(\cap_{x \in T} Cover(x)) = 1 - o(1)$ by noting that $|T| = c$ is a constant. Also recall that the vertices in T do not share a common neighbor since actually even x_0, x_1, x_2 do not. By the second property of Lemma 2, the intersection of their cover sets has only constant size, and thus using Lemma 3 we get $\|D_2 - D_1^2\|_1 = \Omega(1)$. This completes the proof.

4.2 On the Advantage of the Trace Distance Method over the Partition Tree Method

We observed that the partition tree method can be much better than the factorization norm method, and have shown that the factorization norm method can be much better than the trace distance method. To finish the circle, we now show that the trace distance method can be much better than the partition tree method. Different than Theorem 9, this time we can give an explicit function to show the separation.

The Coset function $Coset(G)$ is defined as follows. For a fixed group G, Alice is given a coset x as her input and Bob is given an element $y \in G$ as his input; the question is whether $y \in x$. Aaronson [1] studied the function for the group $\mathbb{Z}_p^2$ (where p is a prime number) and proved that $\mathsf{Q}^1(Coset(\mathbb{Z}_p^2)) = \Theta(\log p)$; that is, Alice asymptotically needs to send the whole input to Bob. Here we show that the partition tree method can only give a very small constant lower bound for this function. The proof is in the full version.

Proposition 1. $\mathsf{PT}(Coset(\mathbb{Z}_p^2)) = 2$.

4.3 Other Discussions of the Power Comparisons

The main goal of this paper is to study the ultimate power of the known lower bound methods for $\mathsf{Q}^1(f)$, and in particular their tightness because of the no-separation conjecture reason mentioned in Section 1. Though it is not our goal to thoroughly study all the six relations between the three methods, it is good to know for more insights. This section so far showed three of them as a circle, leaving the three other relations to discuss. First, it turns out that PT is also weak for random graph functions.

Theorem 9. *For any $\alpha = \Omega(1)$, if $p = N^{-\alpha}$, then an N-node random graph $G(N,p)$ has $\mathsf{PT}(f_G) = O(1)$ with probability $1 - o(1)$.*

For PT over TD, we believe that actually $\mathsf{TD}(\mathsf{Index}) = O(\log n)$, though we can only show it for the *symmetric* distribution μ, *i.e.* $\mu(x,y) = \mu(x',y')$ if $|x| = |x'|$.

Theorem 10. *For any distribution p on $\{0,1,...,n\}$, let $\mu_p(x,y) = p(|x|)$ for all (x,y) with $x_y = 1$. Then the trace distance bound under μ_p for the* Index *function is only $O(\log n)$.*

See the full version for both proofs.

5 Concluding Remarks and Open Questions

The tightness results in this paper call for new lower bound methods for $\mathsf{Q}^1(f)$. With the light shed by comparisons in Section 4, one (vague) approach is trying to somehow combine the advantages of the methods to get a more powerful one.

The factorization norm method appears pretty strong for lower bounding $\mathsf{Q}(f)$. Can we modify it to obtain a good lower bound for $\mathsf{Q}^1(f)$?

Acknowledgment. We would like to thank Rahul Jain for many valuable discussions during the collaboration of paper [14], and Yi-Kai Liu for pointing out the reference [5].

The work was partially supported by China Basic Research Grant 2011CBA00300 (sub-project 2011CBA00301) and Hong Kong General Research Fund 419309 and 418710. The author also benefited from visiting Centre of Quantum Technologies and Tsinghua University, the latter under the support of China Basic Research Grant 2007CB807900 (sub-project 2007CB807901).

References

1. Aaronson, S.: Limitations of quantum advice and one-way communication. Theory of Computing 1, 1–28 (2005)
2. Aaronson, S.: The learnability of quantum states. Proceedings of the Royal Society A 463, 2088 (2007)
3. Aaronson, S., Ambainis, A.: Quantum search of spatial regions. Theory of Computing 1, 47–79 (2005)
4. Ambainis, A., Nayak, A., Ta-Shma, A., Vazirani, U.: Dense quantum coding and quantum finite automata. Journal of the ACM 49(4), 1–16 (2002)
5. Angluin, D.: Queries revisited. Theoretical Computer Science 313(2), 175–194 (2004)
6. Bar-Yossef, Z., Jayram, T.S., Kerenidis, I.: Exponential separation of quantum and classical one-way communication complexity. In: Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC), pp. 128–137 (2004)
7. Buhrman, H., Cleve, R., Watrous, J., de Wolf, R.: Quantum fingerprinting. Physical Review Letters 87(16) (2001)
8. Buhrman, H., Cleve, R., Wigderson, A.: Quantum vs. classical communication and computation. In: Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing (STOC), pp. 63–68 (1998)
9. Gavinsky, D.: Classical interaction cannot replace a quantum message. In: Proceedings of the Fortieth Annual ACM Symposium on the Theory of Computing (STOC), pp. 95–102 (2008)
10. Gavinsky, D., Kempe, J., Kerenidis, I., Raz, R., de Wolf, R.: Exponential separation of quantum and classical one-way communication complexity. In: Proceedings of the 39th Annual ACM Symposium on Theory of Computing (STOC), pp. 516–525 (2007)
11. Gavinsky, D., Pudlák, P.: Exponential separation of quantum and classical non-interactive multi-party communication complexity. In: Proceedings of the 23rd Annual IEEE Conference on Computational Complexity, pp. 332–339 (2008)

12. Høyer, P., Mosca, M., de Wolf, R.: Quantum search on bounded-error inputs. In: Baeten, J.C.M., Lenstra, J.K., Parrow, J., Woeginger, G.J. (eds.) ICALP 2003. LNCS, vol. 2719, pp. 291–299. Springer, Heidelberg (2003)
13. Jain, R., Klauck, H., Nayak, A.: Direct product theorems for classical communication complexity via subdistribution bounds. In: Proceedings of the Fortieth Annual ACM Symposium on the Theory of Computing (STOC), pp. 599–608 (2008)
14. Jain, R., Zhang, S.: New bounds on classical and quantum one-way communication complexity. Theoretical Computer Science 410(26), 2463–2477 (2009)
15. Kalyanasundaram, B., Schintger, G.: The probabilistic communication complexity of set intersection. SIAM Journal on Discrete Mathematics 5(4), 545–557 (1992)
16. Klartag, B., Regev, O.: Quantum one-way communication can be exponentially stronger than classical communication. In: Proceedings of the 44th Annual ACM Symposium on the Theory of Computing (STOC) (to appear, 2011)
17. Klauck, H.: Quantum communication complexity. In: ICALP Satellite Workshops, pp. 241–252 (2000)
18. Klauck, H.: Lower bounds for quantum communication complexity. SIAM Journal on Computing 37(1), 20–46 (2007)
19. Kushilevitz, E., Nisan, N.: Communication Complexity. Cambridge University Press, Cambridge (1997)
20. Linial, N., Shraibman, A.: Lower bounds in communication complexity based on factorization norms. In: Proceedings of the Thirty-Ninth Annual ACM symposium on Theory of Computing (STOC), pp. 699–708 (2007)
21. Littlestone, N.: Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. Machine Learning 2(4), 285–318 (1988)
22. Muthukrishnan, S.M.: Data streams: Algorithms and applications. Foundations and Trends in Theoretical Computer Science 1(2) (2005)
23. Nayak, A.: Optimal lower bounds for quantum automata and random access codes. In: Proceedings of the 40th Annual IEEE Symposium on Foundations of Computer Science (FOCS), pp. 124–133 (1999)
24. Raz, R.: Exponential separation of quantum and classical communication complexity. In: Proceedings of the 31st Annual ACM Symposium on the Theory of Computing (STOC), pp. 358–367 (1999)
25. Razborov, A.: On the distributional complexity of disjointness. Theoretical Computer Science 106, 385–390 (1992)
26. Razborov, A.: Quantum communication complexity of symmetric predicates. Izvestiya: Mathematics 67(1), 145–159 (2003)
27. Sherstov, A.: The pattern matrix method for lower bounds on quantum communication. In: Proceedings of the 40th Annual ACM Symposium on the Theory of Computing, pp. 85–94 (2008)
28. Wigderson, A.: Depth through breadth, or why should we attend talks in other areas? In: Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC), p. 579 (2004), http://www.math.ias.edu/~avi/TALKS/STOC04.ppt
29. Winter, A.: Quantum and classical message identification via quantum channels. Quantum Information and Computation 4(6&7), 563–578 (2004)
30. Yao, A.C.-C.: Some complexity questions related to distributive computing. In: Proceedings of the Eleventh Annual ACM Symposium on Theory of Computing (STOC), pp. 209–213 (1979)

Advice Coins for Classical and Quantum Computation

Scott Aaronson* and Andrew Drucker**

Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02139, USA
aaronson@csail.mit.edu, adrucker@mit.edu
http://www.springer.com/lncs

Abstract. We study the power of classical and quantum algorithms equipped with nonuniform advice, in the form of a coin whose bias encodes useful information. This question takes on particular importance in the quantum case, due to a surprising result that we prove: *a quantum finite automaton with just two states can be sensitive to arbitrarily small changes in a coin's bias.* This contrasts with classical probabilistic finite automata, whose sensitivity to changes in a coin's bias is bounded by a classic 1970 result of Hellman and Cover.

Despite this finding, we are able to bound the power of advice coins for space-bounded classical and quantum computation. We define the classes BPPSPACE/coin and BQPSPACE/coin, of languages decidable by classical and quantum polynomial-space machines with advice coins. Our main theorem is that both classes coincide with PSPACE/poly. Proving this result turns out to require substantial machinery. We use an algorithm due to Neff for finding roots of polynomials in NC; a result from algebraic geometry that lower-bounds the separation of a polynomial's roots; and a result on fixed-points of superoperators due to Aaronson and Watrous, originally proved in the context of quantum computing with closed timelike curves.

Keywords: BQP, finite automata, finite-precision arithmetic, PSPACE, Quantum Computation, root-finding.

1 Introduction

1.1 The Distinguishing Problem

The fundamental task of mathematical statistics is to learn features of a random process from empirical data generated by that process. One of the simplest, yet most important, examples concerns a coin with unknown bias. Say we are given

* This material is based upon work supported by the National Science Foundation under Grant No. 0844626. Also supported by a DARPA YFA grant and a Sloan Fellowship.

** Supported by a DARPA YFA grant.

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 61–72, 2011.

a coin which lands "heads" with some unknown probability q (called the *bias*). In the *distinguishing problem*, we assume q is equal either to p or to $p+\varepsilon$, for some known p, ε, and we want to decide which holds.

A traditional focus is the *sample complexity* of statistical learning procedures. For example, if $p = 1/2$, then $t = \Theta\left(\log\left(1/\delta\right)/\varepsilon^2\right)$ coin flips are necessary and sufficient to succeed with probability $1-\delta$ on the distinguishing problem above. This assumes, however, that we are able to count the number of heads seen, which requires $\log(t)$ bits of memory. From the perspective of computational efficiency, it is natural to wonder whether methods with a much smaller space requirement are possible. This question was studied in a classic 1970 paper by Hellman and Cover [13]. They showed that any (classical, probabilistic) finite automaton that distinguishes with bounded error between a coin of bias p and a coin of bias $p+\varepsilon$, must have $\Omega\left(p\left(1-p\right)/\varepsilon\right)$ states. Their result holds with *no restriction* on the number of coin flips performed by the automaton. This makes the result especially interesting, as it is not immediately clear how sensitive such machines can be to small changes in the bias.

Several variations of the distinguishing problem for space-bounded automata were studied in related works by Hellman [12] and Cover [10]. Very recently, Braverman, Rao, Raz, and Yehudayoff [8] and Brody and Verbin [9] studied the power of restricted-width, *read-once branching programs* for this problem. The distinguishing problem is also closely related to the *approximate majority* problem, in which given an n-bit string x, we want to decide whether x has Hamming weight less than $\left(1/2-\varepsilon\right)n$ or more than $\left(1/2+\varepsilon\right)n$. A large body of research has addressed the ability of constant-depth circuits to solve the approximate majority problem and its variants [1], [3], [4], [17], [20], [21].

1.2 The Quantum Case

In this paper, our first contribution is to investigate the power of *quantum* space-bounded algorithms to solve the distinguishing problem. We prove the surprising result that, in the absence of noise, quantum finite automata with a constant number of states can be sensitive to *arbitrarily small* changes in bias:

Theorem 1 (Informal). *For any $p \in [0,1]$ and $\varepsilon > 0$, there is a quantum finite automaton $M_{p,\varepsilon}$ with just two states (not counting the $|\text{Accept}\rangle$ and $|\text{Reject}\rangle$ states) that distinguishes a coin of bias p from a coin of bias $p+\varepsilon$; the difference in acceptance probabilities between the two cases is at least 0.01. (This difference can be amplified using more states.)*

In other words, the lower bound of Hellman and Cover [13] has no analogue for quantum finite automata. The upshot is that we obtain a natural example of a task that a quantum finite automaton can solve using *arbitrarily* fewer states than a probabilistic finite automaton, not merely exponentially fewer states! Galvao and Hardy [11] gave a related example, involving an automaton that moves continuously through a field φ, and needs to decide whether an integral $\int_0^1 \varphi\left(x\right)dx$ is odd or even, promised that it is an integer. Here, a quantum automaton needs only a single qubit, whereas a classical automaton cannot

guarantee success with any finite number of bits. Naturally, both our quantum automaton and that of [11] only work in the absence of noise.

In the classical case, several variations of this distinguishing task have been explored [13], [10], which modify either the model of computation or the mode of acceptance. We explore some of these variants, adapted to the quantum case, in the full version of the paper.

1.3 Coins as Advice

This unexpected power of quantum finite automata invites us to think further about what kinds of statistical learning are possible using a small number of qubits. In particular, if space-bounded quantum algorithms can detect arbitrarily small changes in a coin's bias, then could a p-biased coin be an incredibly-powerful *information resource* for quantum computation, if the bias p was well-chosen? A bias $p \in (0,1)$ can be viewed in its binary expansion $p = 0.p_1p_2\ldots$ as an infinite sequence of bits; by flipping a p-biased coin, we could hope to access those bits, perhaps to help us perform computations.

This idea can be seen in "Buffon's needle," a probabilistic experiment that in principle allows one to calculate the digits of π to any desired accuracy.[1] It can also be seen in the old speculation that computationally-useful information might somehow be encoded in dimensionless physical constants, such as the fine-structure constant $\alpha \approx 0.0072973525377$ that characterizes the strength of the electromagnetic interaction. But leaving aside the question of which biases $p \in [0,1]$ can be realized by actual physical processes, let us assume that coins of *any* desired bias are available. We can then ask: what computational problems can be solved efficiently using such coins? This question was raised to us by Erik Demaine (personal communication), and was initially motivated by a problem in computational genetics.

In the model that we use, a Turing machine receives an input x and is given access to a sequence of bits drawn independently from an *advice coin* with some arbitrary bias $p_n \in [0,1]$, which may depend on the input length $n = |x|$. The machine is supposed to decide (with high success probability) whether x is in some language L. We allow p_n to depend only on $|x|$, not on x itself, since otherwise the bias could be set to 0 or 1 depending on whether $x \in L$, allowing membership in L to be decided trivially. We let BPPSPACE/coin be the class of languages decidable with bounded error by polynomial-space algorithms with an advice coin. Similarly, BQPSPACE/coin is the corresponding class for polynomial-space quantum algorithms. We impose no bound on the running time of these algorithms.

It is natural to compare these classes with the classes BPPSPACE/poly and BQPSPACE/poly, which consist of all languages decidable by BPPSPACE and BQPSPACE machines respectively, with the help of an arbitrary *advice string* $w_n \in \{0,1\}^*$ that can depend only on the input length $n = |x|$. Compared to the standard advice classes, the power of the coin model is that an advice coin bias p_n can be an arbitrary real number, and so encode infinitely many bits;

[1] See http://en.wikipedia.org/wiki/Buffon%27s_needle

the weakness is that this information is only accessible indirectly through the observed outcomes of coin flips.

It is tempting to try to simulate an advice coin using a conventional advice string, which simply specifies the coin's bias to $\text{poly}(n)$ bits of precision. At least in the classical case, the effect of "rounding" the bias can then be bounded by the Hellman-Cover Theorem. Unfortunately, that theorem (whose bound is essentially tight) is not strong enough to make this work: if the bias p is extremely close to 0 or 1, then a PSPACE machine really *can* detect extremely small changes in p. This means that upper-bounding the power of advice coins is a nontrivial problem even in the classical case. In the quantum case, the situation is even worse, since as mentioned earlier, the quantum analogue of the Hellman-Cover Theorem is false.

Despite these difficulties, we are able to show strong limits on the power of advice coins in both the classical and quantum cases. Our main theorem says that PSPACE machines can effectively extract only $\text{poly}(n)$ bits of "useful information" from an advice coin:

Theorem 2 (Main). $\mathsf{BQPSPACE/coin} = \mathsf{BPPSPACE/coin} = \mathsf{PSPACE/poly}$.

The containment $\mathsf{PSPACE/poly} \subseteq \mathsf{BPPSPACE/coin}$ is easy. On the other hand, proving $\mathsf{BPPSPACE/coin} \subseteq \mathsf{PSPACE/poly}$ appears to be no easier than the corresponding quantum class containment. To prove that $\mathsf{BQPSPACE/coin} \subseteq \mathsf{PSPACE/poly}$, we will need to understand the behavior of a space-bounded advice coin machine M, as we *vary* the coin bias p. By applying a theorem of Aaronson and Watrous [2] (which was originally developed to understand quantum computing with closed timelike curves), we prove the key property that, for each input x, *the acceptance probability* $a_x(p)$ *of* M *is a rational function in* p *of degree at most* $2^{\text{poly}(n)}$. It follows that $a_x(p)$ can "oscillate" between high and low values no more than $2^{\text{poly}(n)}$ times as we vary p. Using this fact, we will show how to identify the "true" bias p^* to sufficient precision with an advice string of $\text{poly}(n)$ bits. What makes this nontrivial is that, in our case, "sufficient precision" sometimes means $\exp(n)$ bits! In other words, the rational functions $a_x(p)$ really *can* be sensitive to doubly-exponentially-small changes to p. Fortunately, we will show that this does not happen too often, and can be dealt with when it does.

In order to manipulate coin biases to exponentially many bits of precision—and to interpret our advice string—in polynomial space, we use two major tools. The first is a space-efficient algorithm for finding roots of univariate polynomials, developed by Neff [14] in the 1990s. The second is a lower bound from algebraic geometry, on the spacing between consecutive roots of a polynomial with bounded integer coefficients. Besides these two tools, we will also need a space-efficient linear algebra algorithm due to Borodin, Cook, and Pippenger [7].

2 Preliminaries

We assume familiarity with basic notions of quantum computation. A detailed treatment of space-bounded quantum Turing machines was given by

Watrous [22]. We also assume familiarity with nonuniform polynomial-sized advice; the advice classes $\mathsf{BPPSPACE/poly}, \mathsf{BQPSPACE/poly}$ consist of languages solvable by classical (resp. quantum) nonuniform polynomial-space bounded algorithms which decide membership with error probability $\leq 1/3$. In this paper, we work with an *asymmetric model*, in which an algorithm M accepts by halting and entering an "Accept" state, but can reject by running forever. Watrous [22] showed that this relaxation does not increase the power of space-bounded computation, and that $\mathsf{BQPSPACE/poly} = \mathsf{BPPSPACE/poly} = \mathsf{PSPACE/poly}$.

We will use two powerful results about polynomials. The first result bounds the *minimum spacing* between zeros of integer polynomials; the second result lets us locate the zeros of univariate polynomials to high precision using a small amount of memory.[2]

Theorem 3 ([5, p. 359, Corollary 10.22]). *Suppose $P(x)$ is a polynomial of degree at most $2^{\mathrm{poly}(n)}$, with integer coefficients with absolute values bounded by $2^{\mathrm{poly}(n)}$. Then if $z, z' \in \mathbb{C}$ are distinct roots of P, we have $|z - z'| \geq 2^{-2^{\mathrm{poly}(n)}}$.*

Theorem 4 ([14], [15], [18]). *There is an algorithm F that takes as input a triple (P, i, j), where P is a degree-d univariate polynomial with rational[3] coefficients whose numerators and denominators are bounded in absolute value by 2^m. F outputs the i^{th} most significant bits of the real and imaginary parts of the binary expansion of the j^{th} zero of P (in some order independent of i, possibly with repetitions). F uses $O(\mathrm{polylog}(d + i + m))$ space.*

2.1 Superoperators and Linear Algebra

We will be interested in S-state quantum finite automata that can include *nonunitary transformations* such as measurements. The state of such an automaton is given by an $S \times S$ *density matrix* [16]. One can transform a density matrix ρ using a *superoperator*, which is any operation of the form

$$\mathcal{E}(\rho) = \sum_j E_j \rho E_j^\dagger, \quad \text{with} \quad E_j \in \mathbb{C}^{S \times S}, \quad \sum_j E_j^\dagger E_j = I \ . \tag{1}$$

We will often work with a "vectorized" representation of mixed states and superoperators. Given a density matrix $\rho \in \mathbb{C}^{S \times S}$, let $\mathrm{vec}(\rho)$ be a vector in $\mathbb{C}^{S^2}$ containing the S^2 entries of ρ. Similarly, given a superoperator $\mathcal{E}$, let $\mathrm{mat}(\mathcal{E}) \in \mathbb{C}^{S^2 \times S^2}$ denote the matrix that describes the action of $\mathcal{E}$ on vectorized mixed states, i.e., that satisfies $\mathrm{mat}(\mathcal{E}) \cdot \mathrm{vec}(\rho) = \mathrm{vec}(\mathcal{E}(\rho))$. The following theorem gives us access to the *fixed-points* of superoperators:

[2] The algorithms of [14], [15], [18] are all stated as *parallel* (NC) algorithms. However, any parallel algorithm can be converted into a space-efficient algorithm, using a standard reduction due to Borodin [6].

[3] [14] takes polynomials with *integer* coefficients as inputs; the result for rational coefficients follows easily by clearing denominators.

Theorem 5 (Aaronson-Watrous [2]). *Let $\mathcal{E}(\rho)$ be a superoperator on an S-dimensional system. Then there exists a second superoperator $\mathcal{E}_{\text{fix}}(\rho)$ on the same system, such that:*

(i) $\mathcal{E}_{\text{fix}}(\rho)$ is a fixed-point of $\mathcal{E}$ for every mixed state ρ: that is, $\mathcal{E}(\mathcal{E}_{\text{fix}}(\rho)) = \mathcal{E}(\rho)$.
(ii) Every mixed state ρ that is a fixed-point of $\mathcal{E}$ is also a fixed-point of $\mathcal{E}_{\text{fix}}$.
(iii) The entries of $\text{mat}(\mathcal{E}_{\text{fix}})$ can be computed from $\text{mat}(\mathcal{E})$ in $\text{polylog}(S)$ space.

2.2 Advice Coin Complexity Classes

We will describe space-bounded quantum algorithms in terms of finite automata. A coin-flipping quantum finite automaton is defined as a *pair* of superoperators $\mathcal{E}_0, \mathcal{E}_1$. Say that a coin has *bias* p if it lands heads with independent probability p every time it is flipped. (A coin here is just a 0/1-valued random variable, with "heads" meaning a 1 outcome.) Let $\$_p$ denote a coin with bias p. When the automaton is given $\$_p$, its state evolves according to the superoperator

$$\mathcal{E}_p := p\mathcal{E}_1 + (1-p)\,\mathcal{E}_0\ . \tag{2}$$

In our model, the superoperators $\mathcal{E}_0, \mathcal{E}_1$ both incorporate a "measurement step" in which the automaton checks whether it is in a designated basis state $|\text{Accept}\rangle$, and if so, halts and accepts. Formally, this is represented by a projective measurement with observables $\{\Gamma_{\text{Acc}}, I - \Gamma_{\text{Acc}}\}$, where $\Gamma_{\text{Acc}} := |\text{Accept}\rangle\langle\text{Accept}|$.

We model a $q(n)$-space quantum Turing machine M with an advice coin as a $2^{q(n)}$-state automaton, with state space $\{|y\rangle\}_{y\in\{0,1\}^{q(n)}}$ and initial state $|0^{q(n)}\rangle$. Given input $x \in \{0,1\}^n$ and advice coin $\$_p$, the machine's state evolves according to a superoperator $\mathcal{E}_p$ (as above), where $\mathcal{E}_0, \mathcal{E}_1$ depend on x and n. Individual entries of the matrix representations of $\mathcal{E}_0, \mathcal{E}_1$ are required to be computable from x in space $\text{poly}(n)$. The machine M has a designated $|\text{Accept}\rangle$ state; we let $v_{\text{Acc}} := \text{vec}(|\text{Accept}\rangle\langle\text{Accept}|)$.

Let ρ_t denote the algorithm's state after t steps, and let $v_t := \text{vec}(\rho_t)$. If we make a standard-basis measurement after t steps, then the probability $a_{x,t}(p)$ of seeing $|\text{Accept}\rangle$ is given by

$$a_{x,t}(p) = \langle\text{Accept}|\,\rho_t\,|\text{Accept}\rangle = v_{\text{Acc}}^{\dagger} v_t. \tag{3}$$

Note that $a_{x,t}(p)$ is nondecreasing in t. Let $a_x(p) := \lim_{t\to\infty} a_{x,t}(p)$. Then BQPSPACE/coin is the class of languages L for which there exists a BQPSPACE machine M, as well as a sequence of advice coin biases $\{p_n\}_{n\geq 1}$, such that for all $x \in \{0,1\}^n$: if $x \in L$, then $a_x(p_n) \geq 2/3$; while If $x \notin L$, then $a_x(p_n) \leq 1/3$.

Note, M may "reject" its input by running forever. We define BPPSPACE/coin similarly, with classical polynomial-space algorithms in place of quantum ones.

3 Quantum Mechanics Nullifies the Hellman-Cover Theorem

Our Theorem 1 shows that there is no quantum analogue of the Hellman-Cover Theorem (described in the Introduction). The key idea is that, in this setting,

a single qubit can be used as an "analog counter," in a way that a classical probabilistic bit cannot. Admittedly, our result would fail were the qubit subject to noise or decoherence, as it would be in a realistic physical situation. We just describe our construction here; the detailed proof of correctness is in the full version. The state of our automaton $M = M_{p,\varepsilon}$ will belong to the Hilbert space spanned by $\{|0\rangle, |1\rangle, |\text{Accept}\rangle, |\text{Reject}\rangle\}$. The initial state is $|0\rangle$. Let

$$U(\theta) := \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \tag{4}$$

be a unitary transformation that rotates counterclockwise by θ, in the "counter subspace" spanned by $|0\rangle$ and $|1\rangle$. Also, let $A, B > 0$ be large values (see the full version). Then M runs the following procedure:

(1) If a 1 bit is encountered (i.e., the coin lands heads), apply $U(\varepsilon(1-p)/A)$.
(2) If a 0 bit is encountered (i.e., the coin lands tails), apply $U(-\varepsilon p/A)$.
(3) With probability $\alpha := \varepsilon^2/B$, "measure" (that is, send $|0\rangle \longrightarrow |\text{Reject}\rangle$ and $|1\rangle \longrightarrow |\text{Accept}\rangle$); otherwise do nothing.

The idea is that the automaton performs a "random walk" in the space of angles between $|0\rangle$ and $|1\rangle$. Under coin bias p, this walk is unbiased; under coin bias $p+\varepsilon$, the walk has a bias that becomes noticeable after $\Theta(1/\varepsilon^2)$ steps, when we expect a measurement to occur. This allows us to distinguish the two cases.

4 Upper-Bounding the Power of Advice Coins

In this section we prove Theorem 2, that $\mathsf{BQPSPACE/coin} = \mathsf{BPPSPACE/coin} = \mathsf{PSPACE/poly}$. First, we claim $\mathsf{PSPACE/poly} \subseteq \mathsf{BQPSPACE/coin}$. This is easy to see: any $\text{poly}(n)$-length advice string a_n can be encoded directly in the bias of an advice coin p_n, and recovered correctly with high probability using $2^{\text{poly}(n)}$ coin flips of $\$_{p_n}$. Thus our main task is to show that $\mathsf{BQPSPACE/coin} \subseteq \mathsf{PSPACE/poly}$. First, let M be any quantum polynomial-space advice coin algorithm, using $s(n) = \text{poly}(n)$ qubits of memory, and with $S = 2^{s(n)}$ states. Fix an input x to M and associated superoperators $\mathcal{E}_0, \mathcal{E}_1, \mathcal{E}_p$. Recalling the notation from Section 2.2, let $B_p := \text{mat}(\mathcal{E}_p)$. Let $\rho_{x,t}(p)$ be the state of M after t steps on input x and coin bias p, and let $v_{x,t}(p) := \text{vec}(\rho_{x,t}(p))$. Let

$$a_{x,t}(p) := v_{\text{Acc}}^{\dagger} v_{x,t}(p) \tag{5}$$

be the probability that M is in the $|\text{Accept}\rangle$ state, if measured after t steps. Let $a_x(p) := \lim_{t\to\infty} a_{x,t}(p)$. As discussed in Section 2.2, the quantities $a_{x,t}(p)$ are nondecreasing in t, so the limit $a_x(p)$ is well-defined.

We now show that—except possibly at a finite number of values—$a_x(p)$ is actually a *rational function* of p, of bounded degree:

Lemma 1. *There exist polynomials $Q(p)$ and $R(p) \neq 0$, of degree at most $S^2 = 2^{\text{poly}(n)}$ in p, such that $a_x(p) = Q(p)/R(p)$ holds whenever $R(p) \neq 0$. Moreover, Q and R have rational coefficients that are computable in $\text{poly}(n)$ space given $x \in \{0,1\}^n$ and the index of the desired coefficient.*

Proof. Throughout, we suppress the dependence on x, so that $a(p) = \lim_{t\to\infty} a_t(p)$ is simply the limiting acceptance probability of a finite automaton $M(\$_p)$ given a coin with bias p. Following Aaronson and Watrous [2], for $z \in (0,1)$ define the matrix $\Lambda_{z,p} \in \mathbb{C}^{S^2\times S^2}$ by

$$\Lambda_{z,p} := z\left[I - (1-z)\,B_p\right]^{-1} \ . \tag{6}$$

The matrix $I-(1-z)\,B_p$ is invertible, since $z>0$ and all eigenvalues of B_p have absolute value at most 1.[4] By Cramer's rule, each entry of $\Lambda_{z,p}$ has form $\frac{f(z,p)}{g(z,p)}$, where f and g are polynomials of degree at most S^2 in both z and p, and $g(z,p)$ is nonzero. Collecting terms, we can write $f(z,p) = c_0(p) + c_1(p)\,z + \cdots + c_{S^2}(p)\,z^{S^2}$, $g(z,p) = d_0(p) + d_1(p)\,z + \cdots + d_{S^2}(p)\,z^{S^2}$. Now let

$$\Lambda_p := \lim_{z\to 0} \Lambda_{z,p}. \tag{7}$$

Aaronson and Watrous [2] showed that Λ_p is precisely the matrix representation $\mathrm{mat}(\mathcal{E}_{\mathrm{fix}})$ of the superoperator $\mathcal{E}_{\mathrm{fix}}$ associated to $\mathcal{E} := \mathcal{E}_p$ by Theorem 5. Thus for all $v \in \mathbb{C}^{S^2}$, we have $B_p(\Lambda_p v) = \Lambda_p v$.

Now, the entries of $\Lambda_{z,p}$ are bivariate rational functions, which have absolute value at most 1 for all z,p. Thus the limit in Eq. (7) must exist, and the coeffients c_k, d_k can be computed in polynomial space using a space-efficient algorithm due to Borodin, Cook, and Pippenger for inversion of matrices with rational entries ([7]; see the full version of our paper).

We claim that every entry of Λ_p can be represented as a rational function of p of degree at most S^2 (a representation valid for all but finitely many p), and that the coefficients of this rational function are computable in polynomial space. To see this, note that the $(i,j)^{th}$ entry of Λ_p has the form

$$(\Lambda_p)_{ij} = \lim_{z\to 0} \frac{f(z,p)}{g(z,p)} = \lim_{z\to 0} \frac{c_0(p) + c_1(p)\,z + \cdots + c_{S^2}(p)\,z^{S^2}}{d_0(p) + d_1(p)\,z + \cdots + d_{S^2}(p)\,z^{S^2}} \ . \tag{8}$$

By basic calculus, the above limit (whenever it exists) equals $c_k(p)/d_k(p)$, where k is the smallest integer such that $d_k(p) \neq 0$. Now let k^* be the smallest integer such that d_{k^*} is not the identically-zero polynomial. Then $d_{k^*}(p)$ has only finitely many zeros. It follows that $(\Lambda_p)_{ij} = c_{k^*}(p)/d_{k^*}(p)$ except when $d_{k^*}(p) = 0$, which is what we wanted to show. Thus in polynomial space, we can loop through all k until we find k^* as above, to compute $c_{k^*}(p)$ and $d_{k^*}(p)$.

Finally, we claim we can write A's limiting acceptance probability $a(p)$ as

$$a(p) = v_{\mathrm{Acc}}^{\dagger} \Lambda_p v_0 \ , \tag{9}$$

where v_0 is the vectorized initial state of A (independent of p). It will follow from Eq. (9) that $a(p)$ has the desired representation, since the map $\Lambda_p \to v_{\mathrm{Acc}}^{\dagger}\Lambda_p v_0$ is linear in the entries of Λ_p and can be performed in polynomial space.

[4] For the latter fact, see [19] and [2, p. 10, footnote 1].

To establish Eq. (9), consider the Taylor series expansion for $\Lambda_{z,p}$, namely $\Lambda_{z,p} = \sum_{t\geq 0} z(1-z)^t B_p^t$, valid for $z \in (0,1)$ (see [2]). The equality $\sum_{t\geq 0} z(1-z)^t = 1$, for $z \in (0,1)$, implies that $v_{\text{Acc}}^\dagger \Lambda_{z,p} v_0$ is a weighted average of the t-step acceptance probabilities $a_t(p)$, for $t \in \{0,1,2,\ldots\}$. Letting $z \to 0$, the weight on each individual step approaches 0. Since $\lim_{t\to\infty} a_t(p) = a(p)$, we obtain Eq. (9), proving Lemma 1.

Lemma 2. *$a_x(p)$ is continuous for all $p \in (0,1)$.*

Lemma 2 above is intuitive, but nontrivial to establish; the proof is in the full version. We now complete the proof of Theorem 2. Let L be a language in BQPSPACE/coin, decided by the quantum polynomial-space advice-coin machine $M(x, \$_p)$ on advice coin biases $\{p_n\}_{n\geq 1}$. We will show that $L \in$ BQPSPACE/poly $=$ PSPACE/poly.

It may not be possible to perfectly specify the bias p_n using $\text{poly}(n)$ bits of advice. Instead, we use our advice string to simulate access to a second bias r_n that is "almost as good" as p_n. This is achieved by the following lemma.

Lemma 3. *Fixing $L, M, \{p_n\}$ as above, there exists a classical polynomial-space algorithm R, as well as a family $\{w_n\}_{n\geq 1}$ of polynomial-size advice strings, for which the following holds. Given an index $i \leq 2^{\text{poly}(n)}$, the computation $R(w_n, i)$ outputs the i^{th} bit of a real number $r_n \in (0,1)$, such that for all $x \in \{0,1\}^n$,*

(i) If $x \in L$, then $\Pr[M(x, \$_{r_n})$ accepts$] \geq 3/5$.
(ii) If $x \notin L$, then $\Pr[M(x, \$_{r_n})$ accepts$] \leq 2/5$.
(iii) The binary expansion of r_n is identically zero, for sufficiently large indices $j \geq h(n) = 2^{\text{poly}(n)}$.

Once Lemma 3 is proved, showing the containment $L \in$ BQPSPACE/poly is easy. First, it is not hard to see that that using the advice w_n from Lemma 3, we can generate r_n-biased coin flips (see the full version). We use this to define a BQPSPACE/poly machine M' that with advice family $\{w_n\}$. Given an input $x \in \{0,1\}^n$, the machine M' uses w_n to simulate $M(x, \$_{r_n})$. Then M' is a BQPSPACE/poly algorithm for L by parts (i) and (ii) of Lemma 3, albeit with error probability 2/5—easily reduced to 1/3 using independent trials. So $L \in$ BQPSPACE/poly $=$ PSPACE/poly, proving Theorem 2.

Proof (of Lemma 3). Fix an input length $n > 0$, and let $p^* := p_n$. For $x \in \{0,1\}^n$, recall that $a_x(p)$ denotes the acceptance probability of $M(x, \$_p)$. We are interested in the way $a_x(p)$ oscillates as we vary p. Define a *transition pair* to be an ordered pair $(x,p) \in \{0,1\}^n \times (0,1)$ such that $a_x(p) \in \{2/5, 3/5\}$. It will be also be convenient to define a larger set of *potential transition pairs*, denoted $\mathcal{P} \subseteq \{0,1\}^n \times [0,1)$, that contains the transition pairs, and whose elements will be easier to enumerate. We defer the precise definition of $\mathcal{P}$.

The advice string w_n will simply specify *the number of distinct potential transition pairs (y,p) such that $p \leq p^*$*. We first give a high-level pseudocode description of the algorithm R, and prove that parts (i) and (ii) of Lemma 3 are

hold; we will then argue that R be implemented in PSPACE, and that part (iii) of the Lemma can also be satisfied. The pseudocode for R is as follows:

```
given (w_n, i)
for all (y, p) ∈ P
    s := 0
    for all (z, q) ∈ P
        if q ≤ p then s := s + 1
    next (z, q)
    if s = w_n then
        let r_n := p + ε (for some small ε = 2^{-2^{poly(n)}})
        output the i^th bit of r_n
    end if
next (y, p)
```

We now prove that parts (i) and (ii) of Lemma 3 are satisfied. We call $p \in [0,1)$ a *transition value* if (y,p) is a transition pair for some $y \in \{0,1\}^n$, and we call p a *potential transition value* if $(y,p) \in \mathcal{P}$ for some $y \in \{0,1\}^n$. Then by definition of w_n, the value r_n produced above is equal to $p_0 + \varepsilon$, where $p_0 \in [0,1)$ is the largest potential transition value less than or equal to p^*. (Note that 0 will always be a potential transition value, so this is well-defined.)

When we define $\mathcal{P}$, we will argue that any distinct potential transition values p_1, p_2 satisfy

$$\min\left\{|p_1 - p_2|, 1 - p_2\right\} \geq 2^{-2^{\mathrm{poly}(n)}}. \tag{10}$$

It follows that if $\varepsilon = 2^{-2^{\mathrm{poly}(n)}}$ is suitably small, then $r_n < 1$, and there are no potential transition values lying in the range $(p_0, r_n]$, or in the range (p_0, p^*).

Now fix any $x \in \{0,1\}^n \cap L$. Since M is a BQPSPACE/coin machine for L with bias p^*, we have $a_x(p^*) \geq 2/3$. If $a_x(r_n) < 3/5$, then Lemma 2 implies that there must be a transition value in the open interval between p^* and r_n. But there are no such transition values. Thus $a_x(r_n) \geq 3/5$. Similarly, if $x \in \{0,1\}^n \setminus L$, then $a_x(r_n) \leq 2/5$. This establishes parts (i) and (ii) of Lemma 3.

Now we formally define the potential transition pairs $\mathcal{P}$. We include $(0^n, 0)$ in $\mathcal{P}$, so that 0 is a potential transition value as required. Now recall, by Lemma 1, that for each $x \in \{0,1\}^n$, the acceptance probability $a_x(p)$ is a rational function $Q_x(p)/R_x(p)$ of degree $2^{\mathrm{poly}(n)}$, for all but finitely many $p \in (0,1)$. Therefore, the function $(a_x(p) - 3/5)(a_x(p) - 2/5)$ also has a rational-function representation: $U_x(p)/V_x(p) = (a_x(p) - 3/5)(a_x(p) - 2/5)$, valid for all but finitely many p. We will include in $\mathcal{P}$ all pairs (x,p) for which $U_x(p) = 0$. It follows from Lemmas 1 and 2 that $\mathcal{P}$ contains all transition pairs, as desired.

We can now establish Eq. (10). Fix any distinct potential transition values $p_1 < p_2$. As $p_2 > 0$, there is some x_2 such that $(x_2, p_2) \in \mathcal{P}$. If $p_1 = 0$, then p_1, p_2 are distinct roots of the polynomial $pU_{x_2}(p)$, whence $|p_1 - p_2| \geq 2^{-2^{\mathrm{poly}(n)}}$ by Theorem 3. Similarly, if $p_1 > 0$, then $(x_1, p_1) \in \mathcal{P}$ for some x_1. Then p_1, p_2 are common roots of $U_{x_1}(p)\,U_{x_2}(p)$, from which it again follows that $|p_1 - p_2| \geq 2^{-2^{\mathrm{poly}(n)}}$. Finally, $1 - p_2 \geq 2^{-2^{\mathrm{poly}(n)}}$ follows since 1 and p_2 are distinct roots of $(1-p)\,U_{x_2}(p)$. Thus Eq. (10) holds.

Next we show that the pseudocode can be implemented in PSPACE. First note that the degrees of U_x, V_x are $2^{\mathrm{poly}(n)}$, whose rational coefficients have numerator and denominator bounded by $2^{\mathrm{poly}(n)}$. Moreover, the coefficients of U_x, V_x are computable in PSPACE from the coefficients of Q_x, R_x (themselves PSPACE-computable). To loop over the elements of $\mathcal{P}$ as in the pseudocode, we can perform an outer loop over $y \in \{0,1\}^n$, and an inner loop over zeros of U_y with the indexing provided by the algorithm from Theorem 4. Any duplicate roots in this indexing can be removed, by comparing each root to all previously visited roots to sufficient precision ($2^{\mathrm{poly}(n)}$ bits suffice, by Theorem 3).

Similarly, if $(y,p), (z,q) \in \mathcal{P}$ then we can determine in PSPACE whether $q \leq p$, as required. The only remaining implementation step is to produce the value r_n in PSPACE, in such a way that part (iii) of Lemma 3 is satisfied. For the chosen value p, and the index $i \leq 2^{\mathrm{poly}(n)}$, we need to produce the i^{th} bit of a value $r_n \in (p, p + 2^{-2^{\mathrm{poly}(n)}})$, such that the binary expansion of r_n is identically zero for sufficiently large $j \geq h(n) = 2^{\mathrm{poly}(n)}$. But this is easily done, since we can compute any desired j^{th} bit of p, for $j \leq 2^{\mathrm{poly}(n)}$, in polynomial space.

5 Open Problems

(1) Let $\mathsf{BQPSPACE/dice}\,(m,k)$ be the class of languages decidable by a BQPSPACE machine that can sample from m distributions $\mathcal{D}_1, \ldots, \mathcal{D}_m$, each of which takes values in $\{1, \ldots, k\}$ (thus, these are "k-sided dice"). We conjecture that $\mathsf{BQPSPACE/dice}\,(1, \mathrm{poly}\,(n)) = \mathsf{BQPSPACE/dice}\,(\mathrm{poly}\,(n), 2) = \mathsf{PSPACE/poly}$. Furthermore, we are hopeful that the techniques of this paper can shed light on these issues.[5] **(2)** Given *any* degree-d rational function $a\,(p)$ such that $0 \leq a\,(p) \leq 1$ for all $0 \leq p \leq 1$, does there exist a d-state (or at least $\mathrm{poly}\,(d)$-state) quantum finite automaton M such that $\Pr[M\,(\$_p)\ \text{accepts}] = a\,(p)$?

Acknowledgments. We thank Erik Demaine for suggesting the advice coins problem to us, and Piotr Indyk for pointing us to the Hellman-Cover Theorem.

References

1. Aaronson, S.: BQP and the polynomial hierarchy. In: Proc. 42nd ACM STOC (2010)
2. Aaronson, S., Watrous, J.: Closed timelike curves make quantum and classical computing equivalent. Proc. Roy. Soc. London A 465, 631–647 (2009)
3. Ajtai, M.: Σ^1_1-formulae on finite structures. Ann. Pure Appl. Logic 24, 1–48 (1983)
4. Amano, K.: Bounds on the size of small depth circuits for approximating majority. In: Albers, S., Marchetti-Spaccamela, A., Matias, Y., Nikoletseas, S.E., Thomas, W. (eds.) ICALP 2009. LNCS, vol. 5555, pp. 59–70. Springer, Heidelberg (2009)

[5] Note that the distinguishing problem for k-sided dice, for $k > 2$, is addressed by the more general form of the theorem of Hellman and Cover [13], while the distinguishing problem for read-once branching programs was explored by Brody and Verbin [9].

5. Basu, S., Pollack, R., Roy, M.F.: Algorithms in Real Algebraic Geometry. Algorithms and Computation in Mathematics. Springer-Verlag New York, Inc., Secaucus (2006)
6. Borodin, A.: On relating time and space to size and depth. SIAM J. Comput. 6(4), 733–744 (1977)
7. Borodin, A., Cook, S., Pippenger, N.: Parallel computation for well-endowed rings and space-bounded probabilistic machines. Information and Control 58(1-3), 113–136 (1983)
8. Braverman, M., Rao, A., Raz, R., Yehudayoff, A.: Pseudorandom generators for regular branching programs. In: Proc. 51st IEEE FOCS (2010)
9. Brody, J., Verbin, E.: The coin problem, and pseudorandomness for branching programs. In: Proc. 51st IEEE FOCS (2010)
10. Cover, T.M.: Hypothesis testing with finite statistics. Ann. Math. Stat. 40(3), 828–835 (1969)
11. Galvao, E.F., Hardy, L.: Substituting a qubit for an arbitrarily large number of classical bits. Phys. Rev. Lett. 90(087902) (2003)
12. Hellman, E.M.: Learning with finite memory. PhD thesis, Stanford University, Department of Electrical Engineering (1969)
13. Hellman, E.M., Cover, T.M.: Learning with finite memory. Ann. of Math. Stat. 41, 765–782 (1970)
14. Neff, C.A.: Specified precision polynomial root isolation is in NC. J. Comput. Sys. Sci. 48(3), 429–463 (1994)
15. Neff, C.A., Reif, J.H.: An efficient algorithm for the complex roots problem. J. Complexity 12(2), 81–115 (1996)
16. Nielsen, M., Chuang, I.: Quantum Computation and Quantum Information. Cambridge University Press, Cambridge (2000)
17. O'Donnell, R., Wimmer, K.: Approximation by DNF: examples and counterexamples. In: Arge, L., Cachin, C., Jurdzinski, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 195–206. Springer, Heidelberg (2007)
18. Pan, V.Y.: Optimal (up to polylog factors) sequential and parallel algorithms for approximating complex polynomial zeros. In: Proc. 27th ACM STOC, pp. 741–750 (1995)
19. Terhal, B., DiVincenzo, D.: On the problem of equilibration and the computation of correlation functions on a quantum computer. Phys. Rev. A 61(022301) (2000)
20. Viola, E.: On approximate majority and probabilistic time. Computational Complexity 18(3), 155–168 (2009)
21. Viola., E.: Randomness buys depth for approximate counting. Electronic Colloquium on Computational Complexity (ECCC), TR10-175 (2010)
22. Watrous., J.: Space-bounded quantum complexity. J. Comput. Sys. Sci. 59(2), 281–326 (1999)

Quantum Commitments from Complexity Assumptions

André Chailloux[1], Iordanis Kerenidis[2,3], and Bill Rosgen[3]

[1] Laboratoire de Recherche en Informatique, Université Paris-Sud
[2] LIAFA, Université Paris Diderot and CNRS
[3] Centre for Quantum Technologies, National University of Singapore

Abstract. We study worst-case complexity assumptions that imply quantum bit-commitment schemes. First we show that $\mathsf{QSZK} \not\subseteq \mathsf{QMA}$ implies a computationally hiding and statistically binding auxiliary-input quantum commitment scheme. We then extend our result to show that the much weaker assumption $\mathsf{QIP} \not\subseteq \mathsf{QMA}$ (which is weaker than $\mathsf{PSPACE} \not\subseteq \mathsf{PP}$) implies the existence of auxiliary-input commitment schemes with quantum advice. Finally, to strengthen the plausibility of the separation $\mathsf{QSZK} \not\subseteq \mathsf{QMA}$ we find a quantum oracle relative to which honest-verifier QSZK is not contained in QCMA.

1 Introduction

The goal of modern cryptography is to design protocols that remain secure under the weakest possible complexity assumptions. Such fundamental protocols include commitment schemes, authentication, one-way functions, and pseudorandom generators. All these primitives have been shown equivalent: for example commitment schemes imply one-way functions [10] and one-way functions imply commitments [7,8,18].

In this paper we study complexity assumptions that imply commitment schemes, which are the basis for many cryptographic constructions, such as zero knowledge protocols for NP [3,6]. A commitment scheme is a two-phase protocol between a sender and a receiver. In the commit phase, the sender interacts with the receiver so that by the end of the phase, the sender is bound to a specific bit, which remains hidden from the receiver until the reveal phase of the protocol, where the receiver learns the bit.

There are two security conditions for such schemes: binding (the sender cannot reveal more than one value) and hiding (the receiver has no information about the bit before the reveal phase). These conditions can hold statistically, i.e. against an unbounded adversary, or computationally, i.e. against a polynomial-time adversary. Without further assumptions these conditions cannot both hold statistically [15,17].

The main complexity assumptions that have been used for the construction of one-way functions, and hence commitments, involve the classes of Computational and Statistical Zero Knowledge. Ostrovsky and Wigderson [20] proved that if Computational Zero Knowledge (ZK) is not trivial then there exists a family of functions that are not 'easy to invert'. The result was extended by Vadhan [24] to show that if ZK does not equal Statistical Zero Knowledge (SZK), then there exists an auxiliary-input one-way function, i.e. one can construct a one-way function given an auxiliary input (or else advice). Auxiliary-input cryptographic primitives are natural when considering worst-case complexity classes: the auxiliary input can encode a 'hard' instance of a problem

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 73–85, 2011.

known only to be hard in the worst case. Last, Ostrovsky and Wigderson also showed that if ZK contains a 'hard-on-average' problem, then 'regular' one-way functions exist.

With the advent of quantum computation and cryptography, one needs to revisit computational security, since many widely-used computational assumptions, such as the hardness of factoring or the discrete logarithm problem, become false when the adversary is a polynomial-time quantum machine [22].

In this paper, we study worst-case complexity assumptions under which quantum commitment schemes exist. As in the classical case, we obtain auxiliary-input commitments: commitments that can be constructed with classical and/or quantum advice. As our commitments are quantum, we define the computational security properties against quantum poly-time adversaries (who also receive an arbitrary quantum auxiliary input).

Theorem 1. *If* $\mathsf{QSZK} \not\subseteq \mathsf{QMA}$ *there exists a non-interactive auxiliary-input quantum commitment scheme that is statistically-binding and computationally-hiding.*

It would be surprising if QSZK is actually contained in QMA. We know that $\mathsf{QSZK} \subseteq \mathsf{QIP}[2]$ [28], where QIP[2] is the class of languages that have quantum interactive proofs with two messages (note that one only needs three messages to get the whole power of quantum interactive proofs). So far, any attempt to reduce QIP[2] or QSZK to QMA or find any plausible assumptions that would imply it, have not been fruitful. This seems harder than in the classical case. The main reason is that the verifier's message cannot be reduced to a public coin message nor to a pure quantum state. His message is entangled with his quantum workspace and this seems inherent for the class QIP[2]. It would be striking if one can get rid of this entanglement and reduce these classes to a single message from the prover.

If we weaken the security condition to hold against quantum adversaries with only classical auxiliary input, then the above assumption also becomes weaker, i.e. $\mathsf{QSZK} \not\subseteq \mathsf{QCMA}$, where QCMA is the class where the quantum verifier receives a single classical message from the prover. We give (quantum) oracle evidence for this by showing that

Theorem 2. *There exists a quantum oracle* A *such that* $\mathsf{QSZK}^A_{\mathsf{HV}} \not\subseteq \mathsf{QCMA}^A$.

The proof appears in the full version of the paper. Our proof of this result extends Aaronson and Kuperberg's result that there is a quantum oracle A such that $\mathsf{QMA}^A \not\subseteq \mathsf{QCMA}^A$ [2]. Subsequent to the completion of this work, Aaronson has shown the stronger result that there is an oracle A such that $\mathsf{SZK}^A \not\subseteq \mathsf{QMA}^A$ [1]. This result implies that our assumption that $\mathsf{QSZK} \not\subseteq \mathsf{QMA}$ is true relative to an oracle.

We then show the existence of commitment schemes based on a much weaker complexity assumption about quantum interactive proofs. More precisely, we look at the class QIP, which was first studied in [27]. This class is believed to be much larger than QSZK. We consider this class and its relation to QMA to show the following

Theorem 3. *If* $\mathsf{QIP} \not\subseteq \mathsf{QMA}$ *there exist non-interactive auxiliary-input quantum commitment schemes (both statistically hiding and computationally binding as well as statistically binding and computationally hiding) with quantum advice.*

Note, that $\mathsf{QIP} = \mathsf{PSPACE}$ [11] and $\mathsf{QMA} \subseteq \mathsf{PP}$ [16], so our assumption is extremely weak, in fact weaker than $\mathsf{PSPACE} \not\subseteq \mathsf{PP}$. Of course, with such a weak assumption we

get a weaker form of commitment: the advice is now quantum. Thus, in order for the prover and the verifier to efficiently perform the commitment for a security parameter n, they need to receive a classical auxiliary input as well as quantum advice of size polynomial in n. This quantum advice is a quantum state on $\mathrm{poly}(n)$ qubits that is not efficiently constructible (otherwise, we could have reduced the quantum advice to classical advice by describing the efficient circuit that produces it). Moreover, the quantum advice we consider does not create entanglement between the players. The key point behind this result is the structure of QIP. More precisely, we use the fact that there exists a QIP-complete problem where the protocol has only three rounds and the verifier's message is a single coin.

All of our commitment schemes are non-interactive. From $\mathsf{QIP} \not\subseteq \mathsf{QMA}$ we construct both statistically hiding and computationally binding commitments as well as statistically binding and computationally hiding ones, whose constructions are conceptually different. In order to prove the security of the first construction, we prove a parallel repetition theorem for protocols based on the swap test that may be of independent interest. From the $\mathsf{QSZK} \not\subseteq \mathsf{QMA}$ assumption we show here only statistically binding and computationally hiding commitments, but computationally binding and statistically hiding commitments can be similarly shown.

2 Definitions

In order to define the statistical distance between quantum states, we use the *trace norm*, given by $\|X\|_{\mathrm{tr}} = \mathrm{tr}\sqrt{X^\dagger X} = \max_U |\mathrm{tr}\, XU|$, where the maximization is taken over all unitaries of the appropriate size. Given one of two quantum states ρ, σ with equal probability, the optimal measurement to distinguish them succeeds with probability $1/2 + \|\rho - \sigma\|_{\mathrm{tr}}/4$ [9]. Note that this measurement is not generally efficient.

The *diamond norm* is a generalization of the trace norm to quantum channels that preserves the distinguishability characterization. Given one of two channels Q_0, Q_1 with equal probability, then the optimal distinguishing procedure using only use one of the channel succeeds with probability $1/2 + \|Q_0 - Q_1\|_\diamond/4$.

In addition to these norms, we will also make use of the *fidelity* between two quantum states [12], which is given by $\mathrm{F}(\rho,\sigma) = \mathrm{tr}\sqrt{\sqrt{\sigma}\rho\sqrt{\sigma}}$. We will use two standard results about the fidelity: for any density matrices ρ and σ we have $1 - \mathrm{F}(\rho,\sigma) \leq \|\rho - \sigma\|_{\mathrm{tr}}/2 \leq \sqrt{1 - \mathrm{F}(\rho,\sigma)^2}$ [5] and $\max_\xi \left(\mathrm{F}(\rho,\xi)^2 + \mathrm{F}(\xi,\sigma)^2\right) = 1 + \mathrm{F}(\rho,\sigma)$ where the maximization is over all density matrices ξ [19,23].

2.1 Quantum Interactive Complexity Classes

The class QMA, first studied in [25], is informally the class of all problems that can be verified by a quantum polynomial-time algorithm with access to a quantum proof.

Definition 4. *A language L is in* QMA *if there is poly-time quantum algorithm V (called the* verifier*) such that*

1. *if $x \in L$, then there exists a state ρ such that $\Pr[V(x,\rho)$ accepts$] \geq a$,*
2. *if $x \notin L$, then for any state ρ, $\Pr[V(x,\rho)$ accepts$] \leq b$,*

where a, b are any efficiently computable functions of $|x|$ with $a > b$ with at least an inverse polynomial gap [13,16]. If ρ is restricted to be classical, the class is QCMA.

The class QIP, first studied in [27], consists of those problems that can be interactively verified in quantum polynomial time. A recent result is that QIP = PSPACE [11].

Definition 5. *A language $L \in$ QIP if there is a poly-time quantum algorithm V exchanging quantum messages with an unbounded prover P such that for any input x*

1. *if $x \in L$ there exists a P such that, (V, P) accepts with probability at least a.*
2. *if $x \notin L$, then for any prover P, (V, P) accepts with probability at most b.*

As in QMA*, we require only that $a > b$ with at least an inverse polynomial gap [14].*

One key property of QIP is that any quantum interactive proof system can be simulated by one using only three messages [14]. In what follows we consider quantum unitary circuits C that output a state in the space $\mathcal{O} \otimes \mathcal{G}$. These spaces can be different for each circuit. $\mathcal{O}$ corresponds to the output space and $\mathcal{G}$ to the garbage space. For any circuit C, we define $|\phi_C\rangle = C|0\rangle$ in the space $\mathcal{O} \otimes \mathcal{G}$ to be the output of the circuit before the garbage space is traced out, and $\rho^C = \mathrm{Tr}_{\mathcal{G}}(|\phi_C\rangle\langle\phi_C|)$ to be the mixed state output by the circuit after the garbage space is traced out. We will also consider more general mixed-state quantum circuits C, that on an input state σ and output a quantum state, denoted by $C(\sigma)$. Unlike unitary circuits, mixed-state circuits are allowed to introduce ancillary qubits and trace out qubits during the computation. Note that circuits of this form can (approximately) represent any quantum channel. The size of a circuit C is equal to the number of gates in the circuit plus the number of qubits used by the circuit, denoted $|C|$. We will also use $|\mathcal{H}|$ to refer to the size of a Hilbert space $\mathcal{H}$ i.e. $|\mathcal{H}| = \lceil \log_2 \dim \mathcal{H} \rceil$. We use $\mathbf{L}(\mathcal{H})$ to refer to the set of all linear operators on $\mathcal{H}$, and $\mathbf{D}(\mathcal{H})$ to denote the subset of these operators that are density matrices. We consider two complete problems for QIP.

Definition 6. *Let μ be a negligible function. We define the promise problem* $\mathrm{QCD} = \{\mathrm{QCD}_Y, \mathrm{QCD}_N\}$ *with input two mixed-state quantum circuits C_0, C_1 of size n as*

- $(C_0, C_1) \in \mathrm{QCD}_Y \Leftrightarrow \|C_0 - C_1\|_\diamond \geq 2 - \mu(n)$
- $(C_0, C_1) \in \mathrm{QCD}_N \Leftrightarrow \|C_0 - C_1\|_\diamond \leq \mu(n)$

Definition 7. *Let μ be a negligible function. We define the promise problem $\Pi = \{\Pi_Y, \Pi_N\}$ with input two mixed-state quantum circuits C_0, C_1 of size n, where the circuit $C_i : \mathbf{D}(\mathcal{X} \otimes \mathcal{Y}) \to \{0,1\}$ for each i.*

- $(C_0, C_1) \in \Pi_Y \Leftrightarrow \exists \rho^0, \rho^1 \in \mathbf{D}(\mathcal{X} \otimes \mathcal{Y})$ *with* $tr_{\mathcal{X}}(\rho^0) = tr_{\mathcal{X}}(\rho^1)$ *such that*

$$\frac{1}{2}\left(\Pr[C_0(\rho^0) = 1] + \Pr[C_1(\rho^1) = 1]\right) = 1$$

- $(C_0, C_1) \in \Pi_N \Leftrightarrow \forall \rho^0, \rho^1 \in \mathbf{D}(\mathcal{X} \otimes \mathcal{Y})$ *with* $tr_{\mathcal{X}}(\rho^0) = tr_{\mathcal{X}}(\rho^1)$ *we have*

$$\frac{1}{2}\left(\Pr[C_0(\rho^0) = 1] + \Pr[C_1(\rho^1) = 1]\right) \leq \frac{1}{2} + \mu(n)$$

QCD is QIP-complete [21] and the QIP-completeness of Π follows from a characterization of QIP due to Mariott and Watrous [16] (we prove this in the full version).

The class QSZK ([26]) is the class of all problems that can be interactively verified by a quantum verifier who learns nothing beyond the truth of the assertion being verified. If the verifier is *honest*, i.e. does not deviate from the protocol, we have

Definition 8. *A language* $L \in \mathsf{QSZK}_{\mathsf{HV}}$ *if*

1. *There is a quantum interactive proof system for* L.
2. *If* $x \in L$, *the state of the verifier after each message can be approximated, within negligible trace distance, by a polynomial-time preparable quantum state.*

If we insist that item 2 holds when the Verifier departs from the protocol, the result is the class QSZK. Watrous has shown that $\mathsf{QSZK}_{\mathsf{HV}} = \mathsf{QSZK}$ [28]. This class has complete problems. We use the following QSZK-complete problem [26].

Definition 9. *Let* μ *be a negligible function.* $\mathrm{QSD} = \{\mathrm{QSD}_Y, \mathrm{QSD}_N\}$ *is the promise problem on input* (C_0, C_1), *unitary circuits of size* n *with* m *output qubits, such that*

- $(C_0, C_1) \in \mathrm{QSD}_Y \Leftrightarrow \left\| \rho^{C_0} - \rho^{C_1} \right\|_{\mathrm{tr}} \geq 2 - \mu(n)$
- $(C_0, C_1) \in \mathrm{QSD}_N \Leftrightarrow \left\| \rho^{C_0} - \rho^{C_1} \right\|_{\mathrm{tr}} \leq \mu(n)$

2.2 Quantum Computational Distinguishability

The following definitions may be found in [28].

Definition 10. *Two mixed states* ρ^0 *and* ρ^1 *on* m *qubits are* (s, k, ε)*-distinguishable if there exists a mixed state* σ *on* k *qubits and a quantum circuit* D *of size* s *that performs a binary outcome measurement on* $(m + k)$ *qubits, such that* $|\Pr[D(\rho^0 \otimes \sigma) = 1] - \Pr[D(\rho^1 \otimes \sigma) = 1]| \geq \varepsilon$. *If* ρ^0 *and* ρ^1 *are not* (s, k, ε)*-distinguishable, then they are* (s, k, ε)*-indistinguishable.*

Let $I \subseteq \{0,1\}^*$ and let an *auxiliary-input state ensemble* be a collection of mixed states $\{\rho_x\}_{x \in I}$ on $r(|x|)$ qubits for polynomial r with the property that they can be efficiently generated given x.

Definition 11. *Two auxiliary-input state ensembles* $\{\rho_x^0\}$ *and* $\{\rho_x^1\}$ *on* I *are* quantum computationally indistinguishable *if for all polynomials* p, s, k *and for all but finitely many* $x \in I$, ρ_x^0 *and* ρ_x^1 *are* $(s(|x|), k(|x|), 1/p(|x|))$*-indistinguishable. Ensembles* $\{\rho_x^0\}$ *and* $\{\rho_x^1\}$ *on* I *are* quantum computationally distinguishable *if there exist polynomials* p, s, k *such that for all* $x \in I$, ρ_x^0 *and* ρ_x^1 *are* $(s(|x|), k(|x|), 1/p(|x|))$*-distinguishable.*

At first glance these definitions of distinguishability and indistinguishability are not complementary. We require distinguishability for all $x \in I$, but require indistinguishability in only all but finitely many $x \in I$. This is because $|x|$ will be our security parameter, and so while a polynomially-bounded adversary may be able to distinguish the two ensembles for a finite number of (small) values of $|x|$, as the parameter grows no efficient algorithm can distinguish the two ensembles.

Key to this definition is that if two ensembles are computationally distinguishable, then for all x there exists an efficient procedure in $|x|$ that distinguishes ρ_x^0 and ρ_x^1 with probability at least $1/2 + 1/p(|x|)$. Note that this is not a uniform procedure: the circuit that distinguishes the two states may depend on x.

Definition 12. *Two auxiliary-input state ensembles $\{\rho_x^0\}$ and $\{\rho_x^1\}$ on I are* quantum statistically indistinguishable *if for any polynomial p and for all but finitely many $x \in I$, $\left\|\rho_x^0 - \rho_x^1\right\|_{\mathrm{tr}} \leq 1/p(|x|)$.*

Definition 13. *Two admissible superoperators Φ^0 and Φ^1 from t qubits to m qubits are (s, k, ε)-distinguishable if there exists a mixed state σ on $t + k$ qubits and a quantum circuit D of size s that performs a binary outcome measurement on $(m + k)$ qubits, such that $|\Pr[D((\Phi^0 \otimes \mathbb{1}_k)(\sigma)) = 1] - \Pr[D((\Phi^1 \otimes \mathbb{1}_k)(\sigma)) = 1]| \geq \varepsilon$, where $\mathbb{1}_k$ denotes the identity superoperator on k qubits. If the superoperators Φ^0 and Φ^1 are not (s, k, ε)-distinguishable, then they are (s, k, ε)-indistinguishable.*

Let $I \subseteq \{0,1\}^*$ and let an *auxiliary-input superoperator ensemble* be a collection of superoperators $\{\Phi_x\}_{x \in I}$ from $q(|x|)$ to $r(|x|)$ qubits for some polynomials q, r, where as in the case of states, given x the superoperators can be performed efficiently in $|x|$.

Definition 14. *Two auxiliary-input superoperator ensembles $\{\Phi_x^0\}$ and $\{\Phi_x^1\}$ on I are* quantum computationally indistinguishable *if for all polynomials p, s, k and for all but finitely many $x \in I$, Φ_x^0 and Φ_x^1 are $(s(|x|), k(|x|), 1/p(|x|))$-indistinguishable. Auxiliary-input ensembles $\{\Phi_x^0\}$ and $\{\Phi_x^1\}$ on I are* quantum computationally distinguishable *if there exist polynomials p, s, k such that for all $x \in I$, Φ_x^0 and Φ_x^1 are $(s(|x|), k(|x|), 1/p(|x|))$-distinguishable.*

If two superoperator ensembles are computationally distinguishable then there is an efficient (nonuniform) procedure (in $|x|$) to distinguish them with probability at least $1/2 + 1/p(|x|)$ for some polynomial p. If the property of being (s, k, ε)-indistinguishable holds for all (unbounded) s and all polynomial $k, 1/\varepsilon$, then we call an ensemble statistically indistinguishable. Note that these definitions provide a strong quantum analogue of the classical non-uniform notion of computational indistinguishability, since the non-uniformity includes an arbitrary quantum state as advice to the distinguisher.

We define a new notion that we will use later on. Intuitively, two circuits that take input in the space $\mathcal{X} \otimes \mathcal{Y}$ and output a single bit are witnessable if there exist two input states that are identical on $\mathcal{Y}$ and are accepted by the two circuits with high probability.

Definition 15. *Two superoperators Φ^0 and Φ^1 from $\mathbf{L}(\mathcal{X} \otimes \mathcal{Y})$ to a single bit are (s, k, p)-witnessable if there exist two input states $\rho^0, \rho^1 \in \mathbf{L}(\mathcal{X} \otimes \mathcal{Y})$ such that*

1. $\frac{1}{2}\left(\Pr[\Phi^0(\rho^0) = 1] + \Pr[\Phi^1(\rho^1) = 1]\right) \geq 1/2 + \frac{1}{p(n)}$
2. *there exists a state $\sigma \in \mathbf{L}(\mathcal{W} \otimes \mathcal{X} \otimes \mathcal{Y})$ with $|\mathcal{W}| = k$ and $\mathrm{tr}_{\mathcal{W}}\, \sigma = \rho_0$, and an admissible superoperator $\Psi : \mathbf{L}(\mathcal{W} \otimes \mathcal{X}) \to \mathbf{L}(\mathcal{X})$ of size s, such that $\rho^1 = (\Psi \otimes \mathbb{1}_{\mathbf{L}(\mathcal{Y})})(\sigma)$ where $\mathbb{1}_{\mathbf{L}(\mathcal{Y})}$ denotes the identity on $\mathbf{L}(\mathcal{Y})$.*

If Φ^0 and Φ^1 are not (s, k, p)-witnessable, then they are (s, k, p)-unwitnessable.

Let $I \subseteq \{0,1\}^*$ and let an *auxiliary-input superoperator ensemble* be a collection of superoperators $\{\Phi_x\}_{x\in I}$ from $q(|x|)$ to 1 bit for a polynomial q, where given x the superoperators can be performed efficiently in $|x|$.

Definition 16. *Auxiliary-input superoperator ensembles $\{\Phi_x^0\}$ and $\{\Phi_x^1\}$ on I are quantum computationally witnessable if there are polynomials s, k, p such that for all $x \in I$, Φ_x^0 and Φ_x^1 are $(s(|x|), k(|x|), p(|x|))$-witnessable. Ensembles $\{\Phi_x^0\}$ and $\{\Phi_x^1\}$ on I are quantum computationally unwitnessable if for all polynomials s, k, p and all but finitely many $x \in I$, Φ_x^0 and Φ_x^1 are $(s(|x|), k(|x|), p(|x|))$-unwitnessable.*

2.3 Quantum Commitments

Definition 17. *A quantum commitment scheme (resp. with quantum advice) is an interactive protocol $Com = (S, R)$ with the following properties*

- *The sender S and the receiver R have common input a security parameter 1^n (resp. both S and R have a copy of a quantum state $|\phi\rangle$ of $\mathrm{poly}(n)$ qubits). The sender has private input the bit $b \in \{0,1\}$ to be committed. Both S and R are quantum algorithms that run in time $\mathrm{poly}(n)$ that may exchange quantum messages.*
- *In the* commit *phase, S interacts with R in order to commit to b.*
- *In the* reveal *phase, S interacts with R in order to reveal b. R decides to accept or reject depending on the revealed value of b and his final state. We say that S reveals b, if R accepts the revealed value. In the honest case, R always accepts.*

A commitment scheme is non-interactive *if the commit and the reveal phase each consist of a single message from S to R. When the commit phase is non-interactive, we call ρ_S^b the state sent by the honest sender during the commit phase when his bit is b.*

Definition 18. *A* non-interactive auxiliary-input quantum commitment scheme (with quantum advice) on I *is a collection of non-interactive quantum commitment schemes (with advice) $\mathcal{C} = \{Com_x = (S_x, R_x)\}_{x\in I}$ such that*

- *there exists a quantum circuit Q of size polynomial in $|x|$, that given as input x for any $x \in I$, can apply the same maps that S_x and R_x apply during the commitment scheme in time polynomial in $|x|$.*
- *(statistically/computationally hiding) the two auxiliary-input state ensembles sent by the honest sender when committing to 0 or 1, which are given by $\{\rho_{S_x}^0\}_{x\in I}$ and $\{\rho_{S_x}^1\}_{x\in I}$, are quantum statistically/computationally indistinguishable.*
- *(statistically/computationally binding) for all but finitely many $x \in I$, for all polynomial p and for any unbounded/polynomial dishonest senders $S_{x,0}^*$, $S_{x,1}^*$ that send the same state in the commit phase*

$$P_{S_x^*} = \frac{1}{2}\left(\Pr[S_{x,0}^* \textit{ reveals } b = 0] + \Pr[S_{x,1}^* \textit{ reveals } b = 1]\right) \leq \frac{1}{2} + \frac{1}{p(|x|)}$$

When referring to a commitment scheme, we will use the (b_s, h_c) and (b_c, h_s) to denote schemes that are statistically binding and computationally hiding and schemes that are computationally binding and statistically hiding, respectively.

3 Quantum Commitments Unless QSZK $\subseteq$ QMA

The idea of the proof is to start from pairs of circuits (C_0, C_1) which are in QSD_Y which means that their mixed state outputs ρ^{C_0} and ρ^{C_1} are statistically far from each other. We want to use ρ^{C_b} as a commitment state for the bit b. Since the states are statistically far away, such a commitment will be statistically binding. For the hiding property, we distinguish two cases. If the Receiver can distinguish in polynomial time (with some quantum auxiliary input) the two states for all but finitely many such pairs of circuits then we show that QSZK $\subseteq$ QMA. If the Receiver cannot distinguish the two states for an infinite set I of pairs of circuits, we show how to construct a non-interactive auxiliary-input quantum (b_s, h_c)-commitment scheme on I. More formally:

Theorem 1. *If* QSZK $\not\subseteq$ QMA*, then there exists a non-interactive auxiliary-input quantum (b_s, h_c)-commitment scheme on an infinite set I.*

Proof. First, we show the following

Lemma 19. *If* QSZK $\not\subseteq$ QMA *then there exist two auxiliary-input state ensembles that are quantum computationally indistinguishable on an infinite set I.*

Proof. Let us consider the complete problem $\mathrm{QSD} = \{\mathrm{QSD}_Y, \mathrm{QSD}_N\}$ for $\mathsf{QSZK}_{\mathsf{HV}}$. We may restrict attention to the honest verifier case, since it is known that QSZK $=$ $\mathsf{QSZK}_{\mathsf{HV}}$ [28]. Let $n = |(C_0, C_1)|$ and define $|\phi_{C_b}\rangle = C_b(|0\rangle)$ in the space $\mathcal{O} \otimes \mathcal{G}$ to be the entire output state of the circuit on input $|0\rangle$ and $\rho^{C_b}_{(C_0,C_1)} = \mathrm{Tr}_{\mathcal{G}}(|\phi_{C_b}\rangle\langle\phi_{C_b}|)$ be the output of circuit C_b on $m(n)$ qubits for a polynomial m.

Recall that the set QSD_Y consists of pairs of circuits (C_0, C_1), such that the trace norm satisfies $\|\rho^{C_0}_{(C_0,C_1)} - \rho^{C_1}_{(C_0,C_1)}\|_{\mathrm{tr}} \geq 2 - \mu(n)$. We now consider the two auxiliary-input state ensembles $\{\rho^{C_0}_{(C_0,C_1)}\}$ and $\{\rho^{C_1}_{(C_0,C_1)}\}$ for $(C_0, C_1) \in \mathrm{QSD}_Y$. Assume for contradiction that they are quantum computationally distinguishable on QSD_Y, i.e. for some polynomials p, s, k and for all $(C_0, C_1) \in \mathrm{QSD}_Y$, the states $\rho^{C_0}_{(C_0,C_1)}$ and $\rho^{C_1}_{(C_0,C_1)}$ are $(s(n), k(n), 1/p(n))$-distinguishable. In other words, for polynomials p, s, k and for all $(C_0, C_1) \in \mathrm{QSD}_Y$ there exists a state σ on $k(n)$ qubits and a quantum circuit Q of size $s(n)$ that performs a two-outcome measurement on $m(n) + k(n)$ qubits, such that

$$|\Pr[Q(\rho^{C_0}_{(C_0,C_1)} \otimes \sigma) = 1] - \Pr[Q(\rho^{C_1}_{(C_0,C_1)} \otimes \sigma) = 1]| \geq \frac{1}{p(n)}.$$

We now claim that this implies that QSZK $\subseteq$ QMA, which is a contradiction. For any input (C_0, C_1) the prover can send the classical polynomial size description of Q to the verifier as well as the mixed state σ with polynomial number of qubits. Then, for all $(C_0, C_1) \in \mathrm{QSD}_Y$, the verifier with the help of Q and σ can distinguish between the two circuits with probability higher than $1/2 + 1/(2p(n))$. On the other hand, for all $(C_0, C_1) \in \mathrm{QSD}_N$, no matter what Q and σ the prover sends, since $\|\rho^{C_0}_{(C_0,C_1)} - \rho^{C_1}_{(C_0,C_1)}\|_{\mathrm{tr}} \leq \mu(n)$ the verifier can only distinguish the two circuits with probability at most $1/2 + \mu(n)/2$. This implies that there is an inverse polynomial gap between the acceptance probabilities in the two cases. By applying standard error reduction tools for QMA [13,16], we obtain a QMA protocol to solve QSD.

This implies that if $\mathsf{QSZK} \not\subseteq \mathsf{QCMA}$ then there exists a non empty set $I \subseteq \mathrm{QSD}_Y$ such that the two auxiliary-input state ensembles $\{\rho^{C_0}_{(C_0,C_1)}\}$ and $\{\rho^{C_1}_{(C_0,C_1)}\}$ are quantum computationally indistinguishable on I. Notice that the set I is infinite. Indeed, if I is finite, then by hard-wiring this finite number of instances into the QMA verifier (who always accepts these instances), we have again that $\mathsf{QSZK} \subseteq \mathsf{QMA}$. ■

We now show how to construct a commitment scheme from these ensembles

Lemma 20. *The two auxiliary-input state ensembles given by $\{\rho^{C_0}_{(C_0,C_1)}\}_{(C_0,C_1)\in I}$ and $\{\rho^{C_1}_{(C_0,C_1)}\}_{(C_0,C_1)\in I}$ that are computationally indistinguishable on the infinite set I imply a non-interactive auxiliary-input quantum (b_s, h_c)-commitment scheme on I.*

Proof. For each $(C_0, C_1) \in I$ we define a scheme with security parameter $n = |(C_0, C_1)|$.
- Commit phase: To commit to bit b, the sender S runs the quantum circuit C_b with input $|0\rangle$ to create $|\phi_{C_b}\rangle = C_b(|0\rangle)$ and sends $\rho^{C_b}_{(C_0,C_1)}$ to the receiver R, which is the portion of $|\phi_{C_b}\rangle$ in the space $\mathcal{O}$.
- Reveal phase: To reveal bit b, the sender S sends the remaining qubits of the state $|\phi_{C_b}\rangle$ to the receiver R, which lie in the space $\mathcal{G}$ (the honest sender sends $|\phi'\rangle = C_b|0\rangle$). The receiver applies the circuit $C_b^\dagger$ on his entire state and then measures all his qubits in the computational basis. He accepts if and only if the outcome is $|0\rangle$.

Note that all operations of the sender and the receiver in the above protocol can be computed in time polynomial in n given the input (C_0, C_1), including the receiver's test during the reveal phase. The protocol is computationally hiding since $\{\rho^{C_0}_{(C_0,C_1)}\}$ and $\{\rho^{C_1}_{(C_0,C_1)}\}$ are quantum computationally indistinguishable.

The fact that the protocol is statistically binding follows from the fact that for the states $\{\rho^{C_0}_{(C_0,C_1)}\}$ and $\{\rho^{C_1}_{(C_0,C_1)}\}$ (for $(C_0, C_1) \in I \subseteq \mathrm{QSD}_Y$) we know that $\|\rho^{C_0}_{(C_0,C_1)} - \rho^{C_1}_{(C_0,C_1)}\|_{\mathrm{tr}} \geq 2-\mu(n)$, for a negligible function μ. More precisely, if ξ is the total quantum state sent by a dishonest sender S^* in the commit and reveal phases of the protocol, then the probability that ξ can be revealed as the bit b is

$$\Pr[S^* \text{ reveals } b \text{ from } \xi] = \mathrm{tr}(|0\rangle\langle 0|C_b^\dagger \xi C_b) = \mathrm{F}(C_b|0\rangle, \xi)^2 \leq \mathrm{F}(\rho^{C_b}_{(C_0,C_1)}, \mathrm{tr}_{\mathcal{G}}\, \xi)^2$$

using the monotonicity of the fidelity with respect to the partial trace. This calculation follows the proof of Watrous that QSZK is closed under complementation [26]. In what follows we consider a dishonest sender that, after the commit phase, sends one of two different states in the reveal phase, so the state held by the Receiver is either ξ_0 or ξ_1. Notice that in either case the Sender sends the same state in the commit phase, so that we have $\mathrm{tr}_{\mathcal{G}}\, \xi_0 = \mathrm{tr}_{\mathcal{G}}\, \xi_1 = \gamma$ for some $\gamma \in \mathbf{D}(\mathcal{O})$. We have

$$\begin{aligned} P_{S^*} &= \frac{1}{2}\left(\Pr[S^* \text{ reveals } b = 0 \text{ from } \xi_0] + \Pr[S^* \text{ reveals } b = 1 \text{ from } \xi_1]\right) \\ &\leq \max_{\gamma \in \mathbf{D}(\mathcal{O})} \frac{1}{2}\left(\mathrm{F}(\rho^{C_0}_{(C_0,C_1)}, \gamma)^2 + \mathrm{F}(\rho^{C_1}_{(C_0,C_1)}, \gamma)^2\right) \\ &= \frac{1}{2}\left(1 + \mathrm{F}(\rho^{C_0}_{(C_0,C_1)}, \rho^{C_1}_{(C_0,C_1)})\right) \leq \frac{1}{2} + \frac{\sqrt{\mu(n)}}{2}. \end{aligned}$$

The final inequality follows from the relationship between the fidelity and the trace norm as well as the fact that $\|\rho^{C_0}_{(C_0,C_1)} - \rho^{C_1}_{(C_0,C_1)}\|_{\mathrm{tr}} \geq 2 - \mu(n)$. This implies that the protocol is statistically binding. ■

By combining the above Lemmas: if QSZK $\not\subseteq$ QMA, then there exists a non-interactive auxiliary-input quantum (b_s, h_c)-commitment scheme on an infinite set I. ■

Notice also that we can use this commitment scheme as a subroutine to construct a scheme that is statistically hiding and computationally binding ([4]).

4 Quantum Commitments Unless QIP $\subseteq$ QMA

In our first construction, we start from pairs of circuits (Q_0, Q_1) which are in QCD_Y which means that there is a common input $|\phi^*\rangle$ such that their outputs ρ^{Q_0} and ρ^{Q_1} are statistically far from each other. We use ρ^{Q_b} as a commitment state for b. The quantum advice needed for the commitment is the following: the Sender receives a copy of $|\phi^*\rangle$ to create the states ρ^{Q_0} and ρ^{Q_1} and the Receiver also gets a copy of $|\phi^*\rangle$ to check via a SWAP test that the Sender did not cheat. Using the fact that the states are statistically far apart and a parallel repetition theorem for our swap-test based protocol we obtain negligible binding error. Last, similarly to the QSZK construction, we show that if QCD cannot be solved in QMA then our scheme is also computationally hiding.

Theorem 21. *If* QIP $\not\subseteq$ QMA*, then there exists a non-interactive auxiliary-input quantum (b_s, h_c)-commitment scheme with quantum advice on an infinite set I.*

We then use instances of the QIP-complete problem Π to construct a (b_c, h_s)-commitment scheme with quantum advice under the assumption that QIP $\not\subseteq$ QMA. We start here from pairs of circuits $Q_0, Q_1 \in \Pi_Y$ and the corresponding input states ρ^0, ρ^1 (see Definition 7) that will be given to the Sender as quantum advice. An honest Sender commits to b by sending half of ρ^b to the Receiver. By definition of ρ^0, ρ^1, the protocol is statistically hiding (in fact perfectly). During the reveal phase, the Sender sends the second half of ρ^b. If $\Pi \notin$ QMA, we show that this protocol is also computationally binding, using our notion of computationally unwitnessable superoperators.

Theorem 22. *If* QIP $\not\subseteq$ QMA*, then there exists a non-interactive auxiliary-input quantum (b_c, h_s)-commitment scheme with quantum advice on an infinite set I.*

Proof. Recall the Complete problem $\Pi = \{\Pi_Y, \Pi_N\}$ from Definition 7 with inputs the mixed-state circuits (Q^0, Q^1) from $\mathbf{D}(\mathcal{X} \otimes \mathcal{Y})$ to a single bit and $n = |(Q^0, Q^1)|$. To show this Theorem, we use the following Lemma. This proof of this Lemma, which is found in the full version of the paper, is very similar to Lemma 19.

Lemma 23. *If* QIP $\not\subseteq$ QMA*, there exist two auxiliary-input superoperator ensembles $\{Q^0\}_{(Q^0,Q^1)\in I}$ and $\{Q^1\}_{(Q^0,Q^1)\in I}$ that are quantum computationally unwitnessable on an infinite set I.*

To finish the proof of the Theorem, we now need to show the following

Lemma 24. *Auxiliary-input superoperator ensembles* $\{Q^0\}_{(Q^0,Q^1)\in I}$ *and* $\{Q^1\}_{(Q^0,Q^1)\in I}$ *that are quantum computationally unwitnessable on an infinite set* $I \subseteq \mathit{\Pi}_Y$ *imply a non-interactive quantum* (b_c, h_s)*-commitment scheme with quantum advice on* I*.*

Proof. Commitment scheme Each $(Q^0, Q^1) \in I \subseteq \mathit{\Pi}_Y$ gives the following scheme

- Let $n = |(Q^0, Q^1)|$ be the security parameter. The sender receives as advice $\rho^0, \rho^1 \in \mathcal{X}^i \otimes \mathcal{Y}^i$ such that $\mathrm{tr}_{\mathcal{X}}\, \rho^0 = \mathrm{tr}_{\mathcal{X}}\, \rho^1$ and $\frac{1}{2}\left(\Pr[Q^0(\rho^0) = 1] + \Pr[Q^1(\rho^1) = 1]\right) \geq 1 - \mu(n)$. For consistency with our definitions, we also suppose that the Receiver gets a copy of ρ^0, ρ^1. These states will not be used in the honest case and they will not harm the security for a cheating Receiver.
- (Commit phase) To commit to b, the Sender sends the state in $\mathcal{Y}^b$ to the Receiver.
- (Reveal phase) To reveal b, the Sender sends the state in $\mathcal{X}^b$. The Receiver applies Q^b on the space $\mathcal{X}^b \otimes \mathcal{Y}^b$ and accepts if he gets 1.

Statistical hiding property. The states that the receiver gets in the commit phase satisfy $\mathrm{tr}_{\mathcal{X}}\, \rho^0 = \mathrm{tr}_{\mathcal{X}}\, \rho^1$ and hence our scheme is perfectly hiding.

Computationally binding property. The property follows from the fact that the two auxiliary-input superoperator ensembles $\{Q^0\}_{(Q^0,Q^1)\in I}$ and $\{Q^1\}_{(Q^0,Q^1)\in I}$ are quantum computationally unwitnessable. Fix $(Q^0, Q^1) \in I$ with $|(Q^0, Q^1)| = n$. After the reveal phase, the Receiver has ρ^b_* in space $\mathcal{X} \otimes \mathcal{Y}$, where b is the revealed bit. Since we consider dishonest senders $S^*_{(Q^0,Q^1)}$ that are quantum polynomial time machines with quantum advice, the states ρ^0_* and ρ^1_* satisfy property 2 of Definition 15. Thus, for all but finitely many $(Q^0, Q^1) \in I$ they do not have property 1 of Definition 15. Then, for such $(Q^0, Q^1) \in I$ we have

$$\begin{aligned} P_{S^*_{(Q^0,Q^1)}} &= \frac{1}{2}\left(\Pr[S^*_{(Q^0,Q^1)} \text{ reveals } b = 0] + \Pr[S^*_{(Q^0,Q^1)} \text{ reveals } b = 1]\right) \\ &= \frac{1}{2}\left(\Pr[Q_0(\rho^0_*) = 1] + \Pr[Q_1(\rho^1_*) = 1]\right) \leq \frac{1}{2} + \frac{1}{p(n)} \end{aligned}$$

for all polynomials p ∎

From the above two Lemmas, unless QIP $\subseteq$ QMA there exists a non-interactive auxiliary-input quantum (b_c, h_s)-commitment scheme with quantum advice on infinite set I. ∎

This result, combined with Theorem 21 completes the proof of Theorem 3.

Acknowledgements

BR is supported by the Centre for Quantum Technologies, which is funded by the Singapore Ministry of Education and the Singapore National Research Foundation. AC and IK are supported by projects ANR-09-JCJC-0067-01, ANR-08-EMER-012 and QCS (grant 255961) of the E.U.

References

1. Aaronson, S.: Impossibility of succinct quantum proofs for collision-freeness. arxiv1101.0403 (2011)
2. Aaronson, S., Kuperberg, G.: Quantum versus classical proofs and advice. Theory of Computing 3(7), 129–157 (2007)
3. Ben-Or, M., Goldreich, O., Goldwasser, S., Håstad, J., Kilian, J., Micali, S., Rogaway, P.: Everything provable is provable in zero-knowledge. In: Goldwasser, S. (ed.) CRYPTO 1988. LNCS, vol. 403, pp. 37–56. Springer, Heidelberg (1990)
4. Crépeau, C., Légaré, F., Salvail, L.: How to convert the flavor of a quantum bit commitment. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 60–77. Springer, Heidelberg (2001)
5. Fuchs, C.A., van de Graaf, J.: Cryptographic distinguishability measures for quantum-mechanical states. IEEE Trans. Inf. Theory 45(4), 1216–1227 (1999)
6. Goldreich, O., Micali, S., Wigderson, A.: Proofs that yield nothing but their validity or all languages in NP have zero-knowledge proof systems. J. ACM 38(3) (1991)
7. Haitner, I., Nguyen, M.H., Ong, S.J., Reingold, O., Vadhan, S.: Statistically hiding commitments and statistical zero-knowledge arguments from any one-way function. SIAM J. Comput. 39(3), 1153–1218 (2009)
8. Håstad, J., Impagliazzo, R., Levin, L.A., Luby, M.: A pseudorandom generator from any one-way function. SIAM J. Comput. 28(4), 1364–1396 (1999)
9. Helstrom, C.W.: Detection theory and quantum mechanics. Inform. Control 10(3) (1967)
10. Impagliazzo, R., Luby, M.: One-way functions are essential for complexity based cryptography. In: IEEE Symp. Found. Comput. Sci. (FOCS), pp. 230–235 (1989)
11. Jain, R., Ji, Z., Upadhyay, S., Watrous, J.: QIP = PSPACE. In: ACM STOC (2010)
12. Jozsa, R.: Fidelity for mixed quantum states. J. Mod. Opt. 41(12), 2315–2323 (1994)
13. Kitaev, A.Y., Shen, A.H., Vyalyi, M.N.: Classical and Quantum Computation. Graduate Studies in Mathematics, vol. 47. American Mathematical Society, Providence (2002)
14. Kitaev, A., Watrous, J.: Parallelization, amplification, and exponential time simulation of quantum interactive proof systems. In: ACM STOC, pp. 608–617 (2000)
15. Lo, H.K., Chau, H.F.: Is quantum bit commitment really possible? Phys. Rev. Lett. 78, 3410 (1997)
16. Marriott, C., Watrous, J.: Quantum Arthur-Merlin games. Comput. Complex. 14(2) (2005)
17. Mayers, D.: Unconditionally secure quantum bit commitment is impossible. Phys. Rev. Lett. 78, 3414 (1997)
18. Naor, M.: Bit commitment using pseudorandomness. J. of Cryptology 4(2), 151–158 (1991)
19. Nayak, A., Shor, P.: Bit-commitment-based quantum coin flipping. Phys. Rev. A 67(1), 012304 (2003)
20. Ostrovsky, R., Wigderson, A.: One-way functions are essential for non-trivial zero-knowledge. In: 2nd Israel Symposium on Theory and Computing Systems, pp. 3–17 (1993)
21. Rosgen, B., Watrous, J.: On the hardness of distinguishing mixed-state quantum computations. In: Conf. Comput. Compl. (CCC), pp. 344–354 (2005)
22. Shor, P.W.: Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM J. Comput. 26(5), 1484–1509 (1997)
23. Spekkens, R.W., Rudolph, T.: Degrees of concealment and bindingness in quantum bit commitment protocols. Phys. Rev. A 65(1), 012310 (2001)

24. Vadhan, S.: An unconditional study of computational zero knowledge. SIAM J. Comput. 36(4), 1160–1214 (2006)
25. Watrous, J.: Succinct quantum proofs for properties of finite groups. In: FOCS 2000 (2000)
26. Watrous, J.: Limits on the power of quantum statistical zero-knowledge. In: FOCS 2002 (2002)
27. Watrous, J.: PSPACE has constant-round quantum interactive proof systems. Theor. Comput. Sci. 292(3), 575–588 (2003)
28. Watrous, J.: Zero-knowledge against quantum attacks. SIAM J. Comput. 39(1), 25–58 (2009)

Limitations on Quantum Dimensionality Reduction

Aram W. Harrow[1,2], Ashley Montanaro[3], and Anthony J. Short[3]

[1] Department of Computer Science & Engineering,
University of Washington, Seattle, USA
[2] Department of Mathematics, University of Bristol, Bristol, UK
[3] Centre for Quantum Information and Foundations, DAMTP,
University of Cambridge, Cambridge, UK

Abstract. The Johnson-Lindenstrauss Lemma is a classic result which implies that any set of n real vectors can be compressed to $O(\log n)$ dimensions while only distorting pairwise Euclidean distances by a constant factor. Here we consider potential extensions of this result to the compression of quantum states. We show that, by contrast with the classical case, there does not exist any distribution over quantum channels that significantly reduces the dimension of quantum states while preserving the 2-norm distance with high probability. We discuss two tasks for which the 2-norm distance is indeed the correct figure of merit. In the case of the trace norm, we show that the dimension of low-rank mixed states can be reduced by up to a square root, but that essentially no dimensionality reduction is possible for highly mixed states.

1 Introduction

The Johnson-Lindenstrauss (JL) Lemma [16] is a dimensionality reduction result which has found a vast array of applications in computer science and elsewhere (see e.g. [14,15,18]). It can be stated as follows:

Theorem 1 (Johnson-Lindenstrauss Lemma [16]). *For all dimensions d, e, there is a distribution $\mathcal{D}$ over linear maps $\mathcal{E} : \mathbb{R}^d \to \mathbb{R}^e$ such that, for all real vectors v, w,*

$$\Pr_{\mathcal{E}\sim\mathcal{D}}[(1-\epsilon)\|v-w\|_2 \leq \|\mathcal{E}(v)-\mathcal{E}(w)\|_2 \leq \|v-w\|_2] \geq 1-\exp(-\Omega(\epsilon^2 e)),$$

where $\|\cdot\|_2$ is the Euclidean (ℓ_2) distance. The lemma is usually applied via the following corollary, which follows by taking a union bound:

Corollary 2. *Given a set S of n d-dimensional real vectors, there is a linear map $\mathcal{E} : \mathbb{R}^d \to \mathbb{R}^{O(\log n/\epsilon^2)}$ that preserves all Euclidean distances in S, up to a multiple of $1-\epsilon$. Further, there is an efficient randomised algorithm to find and implement $\mathcal{E}$.*

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 86–97, 2011.

There are several remarkable aspects of this result. First, the target dimension does not depend on the source dimension d at all. Second, the randomised algorithm can be simply stated as: choose a random e-dimensional subspace with $e = O(\log n/\epsilon^2)$, project each vector in S onto this subspace, and rescale the result by a constant that does not depend on S. Third, this algorithm is *oblivious*: in other words, $\mathcal{E}$ does not depend on the vectors whose dimensionality is to be reduced.

More generally, let ℓ_p^d be the vector space $\mathbb{R}^d$ equipped with the ℓ_p norm $\|\cdot\|_p$. A *randomised embedding* from ℓ_p^d to ℓ_p^e with distortion[1] $1/(1-\epsilon)$ and failure probability δ is a distribution $\mathcal{D}$ over maps $\mathcal{E} : \mathbb{R}^d \to \mathbb{R}^e$ such that, for all $v, w \in \mathbb{R}^d$,

$$\Pr_{\mathcal{E}\sim\mathcal{D}}\left[(1-\epsilon)\|v-w\|_p \leq \|\mathcal{E}(v)-\mathcal{E}(w)\|_p \leq \|v-w\|_p\right] \geq 1-\delta.$$

This definition does not allow the distance between vectors to increase; such embeddings are called contractive. The JL Lemma states that there exists a randomised embedding from ℓ_2^d to ℓ_2^e with distortion $1/(1-\epsilon)$ and failure probability $\exp(-\Omega(\epsilon^2 e))$. Another natural norm to consider in this context is ℓ_1. In this case the situation is less favourable: it has been shown by Charikar and Sahai [8] that there exist $O(d)$ points in ℓ_1^d such that any linear embedding into ℓ_1^e must incur distortion $\Omega(\sqrt{d/e})$. Brinkman and Charikar later gave a set of n points for which any (even non-linear) embedding achieving distortion D requires $n^{\Omega(1/D^2)}$ dimensions [6].

1.1 The JL Lemma in Quantum Information Theory

The JL Lemma immediately gives rise to a protocol for *quantum fingerprinting* [7], or in other words efficient equality testing. Imagine that Alice and Bob each have an n-bit string, and are required to send quantum states of the shortest possible length to a referee, who has to use these states to determine if their bit strings are equal (this is the so-called SMP, or simultaneous message passing, model of communication complexity [17]). Associate each bit string with an orthonormal basis vector of $\mathbb{R}^{2^n}$. Then the JL Lemma guarantees that there exists a map from $\mathbb{R}^{2^n}$ into $\mathbb{R}^{O(n)}$ such that the inner products between all of these 2^n vectors are preserved, up to a small constant. So Alice and Bob each simply apply this map to their vectors, renormalise the output (which makes very little difference to the inner products), and send the $O(\log n)$ qubit states corresponding to the resulting $O(n)$-dimensional vectors to the referee, who applies the swap test to the states [7]. Given two states $|\psi\rangle$, $|\phi\rangle$, this test accepts with probability $\frac{1}{2} + \frac{1}{2}|\langle\psi|\phi\rangle|^2$. As the inner products are approximately preserved by the map into $\mathbb{R}^{O(n)}$, the referee can distinguish between the two cases of the states he receives being equal or distinct, with constant probability.

More generally, Alice and Bob can use a similar SMP protocol to solve the following task: given quantum states $|\psi_A\rangle$, $|\psi_B\rangle$, each picked from a set of k

[1] We use this somewhat clumsy definition of distortion for consistency with prior work.

states, determine $\langle\psi_A|\psi_B\rangle$ up to a constant. Whatever the initial dimension of the states, the JL Lemma (strictly speaking, an easy extension of the JL Lemma to complex vectors) guarantees that they can be compressed to $O(\log k)$ dimensions with at most constant distortion, implying that the referee can estimate $\langle\psi_A|\psi_B\rangle$ up to a constant using only $O(\log\log k)$ qubits of communication.

However, there is a problem with this protocol. While it is oblivious in the sense that it does not depend on the k states which are given as input, it is not oblivious in the following quantum sense: Alice and Bob each need to know what their states are in order to apply the embedding[2]. One would expect the right quantum analogue of a randomised embedding to map quantum states to quantum states in an oblivious fashion. Such an algorithm can be expressed as a distribution over quantum channels (completely positive, trace preserving (CPTP) maps [20,22]), which are the class of physically implementable operations in quantum theory.

Let $\mathcal{B}(d)$ denote the set of d-dimensional Hermitian operators. The distance between quantum states ρ, $\sigma \in \mathcal{B}(d)$ can be measured using the Schatten p-norm $\|\rho-\sigma\|_p$, which is defined as $\|X\|_p = (\sum_i |\lambda_i(X)|^p)^{1/p}$, where $\lambda_i(X)$ is the i'th eigenvalue of X. The case $p = 1$ is known as the trace norm, and $p = 2$ is sometimes known as the Hilbert-Schmidt norm. We have the following definition.

Definition 1. *A* quantum embedding *from $S \subseteq \mathcal{B}(d)$ to $\mathcal{B}(e)$ in the Schatten p-norm, with distortion $1/(1-\epsilon)$ and failure probability δ, is a distribution $\mathcal{D}$ over quantum channels $\mathcal{E} : \mathcal{B}(d) \to \mathcal{B}(e)$ such that, for all ρ, $\sigma \in S$,*

$$\Pr_{\mathcal{E}\sim\mathcal{D}}\left[(1-\epsilon)\|\rho-\sigma\|_p \leq \|\mathcal{E}(\rho)-\mathcal{E}(\sigma)\|_p \leq \|\rho-\sigma\|_p\right] \geq 1-\delta.$$

Rather than only considering embeddings that succeed for all states in $\mathcal{B}(d)$, we generalise the definition to subsets of states. An interesting such subset is the pure states, for which one might imagine stronger embeddings can be obtained. Indeed, a closely related notion has been studied before by Winter [23], and more recently Hayden and Winter [13], under the name of quantum identification for the identity channel. In this setting, the sender Alice has a pure state $|\psi\rangle \in \mathbb{C}^d$ and the receiver Bob is given the description of a pure state $|\phi\rangle \in \mathbb{C}^d$. Alice encodes her state $|\psi\rangle$ as a quantum message using a quantum channel $\mathcal{E} : \mathcal{B}(\mathbb{C}^d) \to \mathcal{B}(\mathbb{C}^e)$ and sends it to Bob, who performs a measurement $(D_\phi, I - D_\phi)$ on the message. The goal is to obtain approximately the same measurement statistics as if Bob had performed the measurement $(|\phi\rangle\langle\phi|, I - |\phi\rangle\langle\phi|)$ on $|\psi\rangle$:

$$\forall\ |\psi\rangle, |\phi\rangle,\ \left|\operatorname{tr}[D_\phi\,\mathcal{E}(|\psi\rangle\langle\psi|)] - |\langle\psi|\phi\rangle|^2\right| \leq \epsilon.$$

Winter showed in [23] that, for constant ϵ, this can be achieved with $e = O(\sqrt{d})$; note that the resulting states $\mathcal{E}(|\psi\rangle\langle\psi|)$ are highly mixed. Winter's result allows the development of a one-way protocol for testing equality of n-bit strings using $\frac{1}{2}\log_2 n + O(1)$ qubits of communication from Alice to Bob, which is still the

[2] On the other hand, if the unphysical operation of postselection is allowed, the JL Lemma can be applied directly.

best known separation between one-way quantum and classical communication complexity for total functions [1]. In our terminology, the result of [23] shows that there exists a quantum embedding from $\mathcal{B}(d)$ to $\mathcal{B}(O(\sqrt{d}))$ that approximately preserves the trace distance between (initially) pure states. But note that one aspect of Winter's result is stronger than we need: he showed the existence of a channel such that the distance is approximately preserved between *all* pairs of states. Here, we are interested in finding distributions $\mathcal{D}$ over channels $\mathcal{E}$ such that, for an arbitrary pair of states, the distance is approximately preserved with high probability; this is potentially a weaker notion. In particular, it is not necessarily true that the individual channel obtained by averaging over $\mathcal{D}$ will preserve the distance between an arbitrary pair of states.

We pause to mention that the JL Lemma has found some other uses in quantum information theory. Cleve et al [9] used it to give an upper bound on the amount of shared entanglement required to win a particular class of nonlocal games. Gavinsky, Kempe and de Wolf [11] used it to give a simulation of arbitrary quantum communication protocols by quantum SMP protocols (with exponential overhead). Embeddings between norms have also been used. Aubrun, Szarek and Werner [4,3] have used a version of Dvoretzky's theorem on "almost-Euclidean" subspaces of matrices under Schatten norms to give counterexamples to the additivity conjectures of quantum information theory. And, very recently, Fawzi, Hayden and Sen [10] have used ideas from the theory of low-distortion embeddings of the "$\ell_1(\ell_2)$" norm to prove the existence of strong entropic uncertainty relations.

1.2 Our Results

In this paper, we show that the dimensionality reduction that can be achieved by quantum embeddings is very limited. We begin, in Section 2, by considering the Schatten 2-norm (which is just the vector 2-norm on matrices). We show that, in stark contrast to the JL Lemma, any quantum embedding which preserves the 2-norm distance between (say) orthogonal pure states with constant distortion and constant failure probability can only achieve at most a constant reduction in dimension. One potential criticism of this result is that the 2-norm is not usually seen as a physically meaningful distance measure, as compared with the trace norm. However, we argue in Section 3 that for certain problems the 2-norm is indeed the correct distance measure. We discuss two problems – equality testing without a reference frame and state discrimination with a random measurement – where the 2-norm appears naturally as the figure of merit.

In Section 4 we turn to the trace norm, for which we have upper and lower bounds. On the upper bound side, we extend the result of Winter [23] to show that low-rank mixed states are also amenable to dimensionality reduction; roughly speaking, d-dimensional mixed states of rank r can be embedded into $O(\sqrt{rd})$ dimensions with constant distortion. On the other hand, we show using the 2-norm lower bound that highly mixed states cannot be embedded into low dimension: there is a lower bound of $\Omega(\sqrt{d}\frac{\|\rho-\sigma\|_1}{\|\rho-\sigma\|_2})$ on the target dimension of any constant distortion trace norm embedding that succeeds with constant

probability for the pairs $U\rho U^\dagger$, $U\sigma U^\dagger$ for all unitary operators U. In particular, this implies an $\Omega(\sqrt{d})$ lower bound for any embedding which succeeds for a unitarily invariant set of states. In the case that $|\rho - \sigma|$ is proportional to a projector (i.e. all non-zero eigenvalues of $\rho - \sigma$ are equal in absolute value), our upper and lower bounds coincide. Due to space limitations, many proofs are omitted; for these, see the full version [12].

Finally, some notes on miscellaneous notation. F_d will denote the unitary operator which swaps (or flips) two d-dimensional quantum systems (i.e. $F_d = \sum_{i,j=1}^{d} |i\rangle\langle j| \otimes |j\rangle\langle i|$), and I_d will denote the d-dimensional identity matrix. Whenever we say that $U \in U(d)$ is a random unitary operator, we mean that U is picked uniformly at random according to Haar measure on the unitary group $U(d)$.

2 Dimensionality Reduction in the 2-Norm

We now show that quantum dimensionality reduction in the 2-norm is very limited.

Theorem 3. *Let $\mathcal{D}$ be a distribution over quantum channels (CPTP maps) $\mathcal{E} : \mathcal{B}(\mathbb{C}^d) \to \mathcal{B}(\mathbb{C}^e)$ such that, for fixed quantum states $\rho \neq \sigma$ and for all unitary operators $U \in U(d)$,*

$$\Pr_{\mathcal{E}\sim\mathcal{D}}[\|\mathcal{E}(U\rho U^\dagger) - \mathcal{E}(U\sigma U^\dagger)\|_2 \geq (1-\epsilon)\|U\rho U^\dagger - U\sigma U^\dagger\|_2] \geq 1 - \delta$$

for some $0 \leq \epsilon, \delta \leq 1$. Then $e \geq (1-\delta)(1-\epsilon)^2 d$.

Note that the above lower bound on target dimension holds for any embedding of a unitarily invariant set of states. For example, taking ρ and σ to be orthogonal pure states and inserting $\epsilon = \delta = 0$ recovers the (unsurprising) result that any embedding that exactly preserves distances between all orthogonal pure states with certainty must satisfy $e \geq d$. More generally, if we have an embedding which succeeds with constant probability and has constant distortion, the target dimension can be no smaller than $\Omega(d)$. In order to prove the theorem, we will need the following two technical lemmas.

Lemma 4. *Let $\mathcal{E} : \mathcal{B}(\mathbb{C}^d) \to \mathcal{B}(\mathbb{C}^e)$ be a quantum channel (CPTP map). Then*

$$\mathrm{tr}[F_e\, \mathcal{E}^{\otimes 2}(F_d)] \leq de.$$

Lemma 5. *Let ρ and σ be d-dimensional quantum states. Then*

$$\int U^{\otimes 2}(\rho - \sigma)^{\otimes 2}(U^\dagger)^{\otimes 2} dU = \frac{\|\rho - \sigma\|_2^2}{d^2 - 1}\left(F_d - \frac{I_{d^2}}{d}\right).$$

The following lemma is the key to most of the results in this paper.

Lemma 6. *Let ρ and σ be quantum states and let $\mathcal{E} : \mathcal{B}(\mathbb{C}^d) \to \mathcal{B}(\mathbb{C}^e)$ be a quantum channel. Then*

$$\int \|\mathcal{E}(U\rho U^\dagger) - \mathcal{E}(U\sigma U^\dagger)\|_2^2 \, dU \leq \frac{d(e^2-1)}{e(d^2-1)} \|\rho - \sigma\|_2^2.$$

Proof. We have

$$\begin{aligned}
&\int \|\mathcal{E}(U\rho U^\dagger) - \mathcal{E}(U\sigma U^\dagger)\|_2^2 \, dU = \int \|\mathcal{E}(U(\rho-\sigma)U^\dagger)\|_2^2 \, dU \\
&= \int \operatorname{tr}[F_e \, \mathcal{E}(U(\rho-\sigma)U^\dagger)^{\otimes 2}] \, dU = \operatorname{tr}\left[F_e \, \mathcal{E}^{\otimes 2}\left(\int U^{\otimes 2}(\rho-\sigma)^{\otimes 2}(U^\dagger)^{\otimes 2} \, dU\right)\right] \\
&= \frac{\|\rho-\sigma\|_2^2}{d^2-1} \operatorname{tr}\left[F_e \, \mathcal{E}^{\otimes 2}\left(F_d - \frac{I_{d^2}}{d}\right)\right] \leq \frac{\|\rho-\sigma\|_2^2}{d^2-1}\left(de - d\operatorname{tr}[\mathcal{E}(I_d/d)^2]\right) \\
&\leq \frac{d(e^2-1)}{e(d^2-1)} \|\rho-\sigma\|_2^2.
\end{aligned}$$

We use linearity of $\mathcal{E}$ in the first equality, and the second equality is the tensor product trick $\operatorname{tr}[X^2] = \operatorname{tr}[F_e X^{\otimes 2}]$ for e-dimensional operators X. The fourth equality is Lemma 5, the first inequality is Lemma 4, and the second inequality is simply $\operatorname{tr}\rho^2 \geq 1/e$ for all e-dimensional states ρ. □

We are finally ready to prove Theorem 3.

Proof (of Theorem 3). We will prove something slightly stronger: that for a random U, the 2-norm is not approximately preserved under a map $\mathcal{E}$ picked from $\mathcal{D}$, unless e is almost as large as d. So assume

$$\Pr_{\mathcal{E}\sim\mathcal{D},\, U\in U(d)}\left[\|\mathcal{E}(U\rho U^\dagger) - \mathcal{E}(U\sigma U^\dagger)\|_2 \geq (1-\epsilon)\|U\rho U^\dagger - U\sigma U^\dagger\|_2\right] \geq 1-\delta,$$

or equivalently

$$\Pr_{\mathcal{E}\sim\mathcal{D},\, U\in U(d)}\left[\|\mathcal{E}(U\rho U^\dagger) - \mathcal{E}(U\sigma U^\dagger)\|_2^2 \geq (1-\epsilon)^2\|\rho-\sigma\|_2^2\right] \geq 1-\delta,$$

where we use the unitary invariance of the 2-norm. By Markov's inequality, this implies that

$$\int_{\mathcal{E}\sim\mathcal{D}} \int \|\mathcal{E}(U\rho U^\dagger) - \mathcal{E}(U\sigma U^\dagger)\|_2^2 \, dU \geq (1-\delta)(1-\epsilon)^2\|\rho-\sigma\|_2^2,$$

implying in turn that there must exist some $\mathcal{E}$ such that

$$\int \|\mathcal{E}(U\rho U^\dagger) - \mathcal{E}(U\sigma U^\dagger)\|_2^2 \, dU \geq (1-\delta)(1-\epsilon)^2\|\rho-\sigma\|_2^2.$$

So let $\mathcal{E} : \mathcal{B}(\mathbb{C}^d) \to \mathcal{B}(\mathbb{C}^e)$ be a quantum channel that does satisfy this inequality. Then we have

$$(1-\delta)(1-\epsilon)^2\|\rho-\sigma\|_2^2 \leq \int \|\mathcal{E}(U\rho U^\dagger) - \mathcal{E}(U\sigma U^\dagger)\|_2^2 \, dU \leq \left(\frac{e}{d}\right)\|\rho-\sigma\|_2^2,$$

where the second inequality follows from Lemma 6, assuming that $e \leq d$. We have shown that $e \geq (1-\delta)(1-\epsilon)^2 d$, completing the proof of the theorem. □

3 Operational Meaning of the 2-Norm

In this section, we discuss the meaning of the 2-norm distance between quantum states. It is usually assumed that the trace norm is the "right" measure of distance between states, and proofs going via the 2-norm usually do so only for calculational simplicity. However, here we argue that the 2-norm is of interest in its own right, by giving two operational interpretations of this distance measure.

3.1 Equality Testing without a Reference Frame

Consider the following equality-testing game. We are given a description of two different states ρ and σ. An adversary prepares two systems in one of the states $\rho \otimes \rho$, $\sigma \otimes \sigma$, $\rho \otimes \sigma$ or $\sigma \otimes \rho$, with equal probability of each. He then applies an unknown unitary U to each system (i.e. he applies $U \otimes U$ to the joint state). Our task is to determine whether the two systems have the same state or different states. This models equality testing in a two-party scenario in which the preparer and tester do not share a reference frame [5]. One protocol for solving this task is simply to apply the swap test [7] to the two states we are given, output "same" if the test accepts, and "different" otherwise. When applied to two states ρ, σ this test accepts with probability $\frac{1}{2} + \frac{1}{2} \operatorname{tr} \rho\sigma$, so for any U the overall probability of success is

$$\frac{1}{4}\left(\frac{1}{2} + \frac{1}{2}\operatorname{tr}[\rho^2]\right) + \frac{1}{4}\left(\frac{1}{2} + \frac{1}{2}\operatorname{tr}[\sigma^2]\right) + \frac{1}{2}\left(\frac{1}{2} - \frac{1}{2}\operatorname{tr}[\rho\sigma]\right) = \frac{1}{2} + \frac{1}{8}\|\rho - \sigma\|_2^2.$$

The following theorem shows that this is optimal.

Theorem 7. *The maximal probability of success of the above game is* $\frac{1}{2} + \frac{1}{8}\|\rho - \sigma\|_2^2$.

Proof. Let $(M, I - M)$ be an arbitrary POVM[3] where the operator M corresponds to the answer "same". Then the probability of success achieved by this POVM for a given U is $\frac{1}{2} + \frac{1}{2}B$, where B is the *bias*, which is equal to

$$\operatorname{tr}\left[M\left(\frac{1}{2}(U(\rho\otimes\rho)U^\dagger + U(\sigma\otimes\sigma)U^\dagger - \frac{1}{2}(U(\rho\otimes\sigma)U^\dagger + U(\sigma\otimes\rho)U^\dagger\right)\right].$$

If the adversary adopts the strategy of picking U uniformly at random, the average bias obtained is

$$\begin{aligned}&\frac{1}{2}\operatorname{tr}\left[M\int U^{\otimes 2}(\rho\otimes\rho + \sigma\otimes\sigma - \rho\otimes\sigma - \sigma\otimes\rho)(U^\dagger)^{\otimes 2}dU\right]\\ =\ &\frac{1}{2}\operatorname{tr}\left[M\int U^{\otimes 2}(\rho-\sigma)^{\otimes 2}(U^\dagger)^{\otimes 2}\right],\end{aligned}$$

which by Lemma 5 is equal to $\frac{\|\rho-\sigma\|_2^2}{2(d^2-1)}\operatorname{tr}\left[M\left(F_d - \frac{I_{d^2}}{d}\right)\right]$.

[3] POVMs (positive operator valued measures) are the most general class of measurements in quantum theory. A POVM is defined by a set of positive operators summing to the identity.

This expression is maximised by setting M equal to a projector onto the subspace spanned by the eigenvectors of $F_d - \frac{I_{d^2}}{d}$ with positive eigenvalues. As F_d has $d(d+1)/2$ eigenvalues equal to 1, and $d(d-1)/2$ eigenvalues equal to -1, we obtain $\operatorname{tr}\left[M\left(F_d - \frac{I_{d^2}}{d}\right)\right] = (d^2-1)/2$. This implies that the average bias is at most $\frac{1}{4}\|\rho-\sigma\|_2^2$. As the worst-case bias can only be lower, this implies the claimed result. □

3.2 Performing a Random Measurement

The second game we will discuss is state discrimination with a fixed or random measurement. Imagine we are given a state which is promised to be either ρ or σ, with equal probability of each, and we wish to determine which is the case. It is well known that the largest bias achievable by choosing an appropriate measurement is $\frac{1}{2}\|\rho-\sigma\|_1$ (recall from the previous section that the bias B and the success probability p have the relationship $p = \frac{1}{2} + \frac{B}{2}$). But how well can we do if the measurement we apply does not in fact depend on ρ and σ?

We will see that $\|\rho-\sigma\|_2$ is closely related to the optimal bias achievable by performing one of the following two measurements, and deciding whether the state is ρ or σ based on the outcome.

- The uniform (isotropic) POVM whose measurement elements consist of normalised projectors onto all states $|\psi\rangle$;
- A projective measurement in a random basis (i.e. applying a random unitary operator and measuring in the computational basis).

In general, the largest bias achievable by measuring a POVM M which consists of measurement operators M_i can be written as $\frac{1}{2}\sum_i |\operatorname{tr}[M_i(\rho-\sigma)]|$. Each measurement operator of the uniform POVM is given by the projector onto some state $|\psi\rangle$, normalised by a factor of d (to check that this is right, note that

$$d\int d\psi |\psi\rangle\langle\psi| = d\left(\frac{I_d}{d}\right) = I_d$$

as expected). So the bias induced by the uniform POVM is $\frac{d}{2}\int d\psi |\langle\psi|(\rho-\sigma)|\psi\rangle|$. In the case of a measurement in a random basis $U \in U(d)$, we can calculate the *expected* bias as follows:

$$\frac{1}{2}\mathbb{E}_U \sum_{i=1}^{d} |\langle i|U^\dagger(\rho-\sigma)U|i\rangle| = \frac{1}{2}\sum_{i=1}^{d} \mathbb{E}_U |\langle i|U^\dagger(\rho-\sigma)U|i\rangle|$$

$$= \frac{1}{2}\sum_{i=1}^{d} \mathbb{E}_U |\langle 1|U^\dagger(\rho-\sigma)U|1\rangle| = \frac{d}{2}\int d\psi |\langle\psi|(\rho-\sigma)|\psi\rangle|;$$

so these quantities are the same. They are also closely related to the 2-norm distance, as we will now see.

Theorem 8. *Let ρ, σ be d-dimensional quantum states. Then*

$$\frac{1}{3}\|\rho - \sigma\|_2 \leq d \int d\psi |\langle\psi|(\rho - \sigma)|\psi\rangle| \leq \|\rho - \sigma\|_2.$$

The lower bound in Theorem 8 was shown by Ambainis and Emerson [2] (see also the proof of Matthews, Wehner and Winter [19]), and the upper bound is not hard. However, as this result does not appear to be widely known, we include a proof (which is essentially the same as that of [19]) in the full version [12].

In fact, the corresponding upper and lower bounds on the bias hold for any fixed POVM whose measurement vectors form a 4-design [2], and the upper bound even holds for any fixed POVM whose vectors form a 2-design. This result can be useful in cases where one wishes to perform state discrimination without necessarily being able to construct the optimal measurement efficiently [21]. See the work [19] for much more detail on the bias achievable in state discrimination with fixed measurements.

4 Dimensionality Reduction in the Trace Norm

In this section we consider embeddings that reduce dimension while preserving the trace norm distance between states. As no quantum channel can increase this distance, we first observe that any such embedding will automatically be contractive.

4.1 Upper Bound

It was previously shown by Winter [23] that, in our language, d-dimensional pure states can be embedded into $\mathcal{B}(O(\sqrt{d}))$ with constant distortion. We now extend this result to general mixed states, by showing that rank r mixed states can be embedded into dimension $O(\sqrt{rd})$ with constant distortion.

The embedding is conceptually very simple: apply a random unitary and trace out a subsystem. However, when the target dimension e does not divide d, we are forced to consider random isometries $V : \mathbb{C}^d \to \mathbb{C}^e \otimes \mathbb{C}^{\lceil d/e \rceil}$ instead of unitaries, where $\lceil x \rceil$ is the smallest integer y such that $y \geq x$. Recall that an isometry is a norm-preserving linear map, i.e. a map taking an orthonormal basis of one space to an orthonormal set of vectors in another (potentially larger) space. A random isometry is defined as a fixed isometry followed by a random unitary. Formally, our embedding is a distribution over the following quantum channels $\mathcal{E}_V$.

Definition 2. *Let d and e be positive integers such that $e \leq d$. For any isometry $V : \mathbb{C}^d \to \mathbb{C}^e \otimes \mathbb{C}^{\lceil d/e \rceil}$, let $\mathcal{E}_V : \mathcal{B}(\mathbb{C}^d) \to \mathcal{B}(\mathbb{C}^e)$ be the quantum channel that consists of performing V, then tracing out (discarding) the second subsystem.*

We now analyse the performance of the embedding obtained by picking a random V and applying this channel.

Theorem 9. *Let d be a positive integer, and let ρ and σ be arbitrary d-dimensional mixed states such that ρ has rank r. Fix ϵ such that $0 < \epsilon < 1$. For any e such that $2\sqrt{rd/\epsilon} \leq e \leq d$, let $\mathcal{D}$ be the distribution on channels $\mathcal{E}_V : \mathcal{B}(\mathbb{C}^d) \to \mathcal{B}(\mathbb{C}^e)$ that is uniform on isometries $V : \mathbb{C}^d \to \mathbb{C}^e \otimes \mathbb{C}^{\lceil d/e \rceil}$. Then*

$$\Pr_{\mathcal{E}_V \sim \mathcal{D}}[\|\mathcal{E}_V(\rho) - \mathcal{E}_V(\sigma)\|_1 \geq (1-\epsilon)\|\rho - \sigma\|_1] \geq 1 - d\exp(-K\epsilon d),$$

for a universal constant K which may be taken to be $(1 - \ln 2)/(2 \ln 2) \approx 0.22$.

Although this result is expressed in terms of the rank of the input states, a similar result would apply to states which are very close (in trace norm) to having low rank, but for simplicity we do not discuss this here.

4.2 Lower Bound

It turns out that Lemma 6 is also strong enough to give a bound on embeddings of the trace norm, via a similar proof to that of Theorem 3. Charikar and Sahai [8] showed that there exist a set of $O(d)$ d-dimensional vectors whose dimension cannot be significantly reduced while preserving their ℓ_1 distances. One might expect the same to be true for the trace norm, as the trace norm on diagonal matrices is just the ℓ_1 norm of the diagonal entries. However, note that this does not follow immediately from Charikar and Sahai's work, as it is conceivable that an embedding mapping diagonal to non-diagonal matrices could do better. Nevertheless, we now show that dimensionality reduction is impossible for some sets of highly mixed states.

Theorem 10. *Let $\mathcal{D}$ be a distribution over quantum channels (CPTP maps) $\mathcal{E} : \mathcal{B}(\mathbb{C}^d) \to \mathcal{B}(\mathbb{C}^e)$ such that, for fixed quantum states $\rho \neq \sigma$ and for all unitary U,*

$$\Pr_{\mathcal{E} \sim \mathcal{D}}[\|\mathcal{E}(U\rho U^\dagger) - \mathcal{E}(U\sigma U^\dagger)\|_1 \geq (1-\epsilon)\|U\rho U^\dagger - U\sigma U^\dagger\|_1] \geq 1 - \delta$$

for some $0 \leq \epsilon, \delta \leq 1$. Then $e \geq (1-\delta)(1-\epsilon)\sqrt{d}\frac{\|\rho-\sigma\|_1}{\|\rho-\sigma\|_2}$. In particular, if ρ and σ are orthogonal pure states, then $e \geq (1-\delta)(1-\epsilon)\sqrt{2d}$, and if ρ and σ are proportional to projectors onto orthogonal $d/2$-dimensional subspaces, $e \geq (1-\delta)(1-\epsilon)d$.

So we see that achieving any significant dimensionality reduction for arbitrary highly mixed states is impossible, and even for pure states the dimension can only be reduced by a square root (a similar result for pure states was shown in [23]).

This implies that the protocol of Theorem 9 is optimal for certain families of states, up to constant factors. Consider the family of pairs $U\rho U^\dagger$, $U\sigma U^\dagger$ for all $U \in U(d)$, where ρ and σ are proportional to projectors onto orthogonal r-dimensional subspaces of $\mathbb{C}^d$. Then $\frac{\|\rho-\sigma\|_1}{\|\rho-\sigma\|_2} = \sqrt{\text{rank}(\rho-\sigma)} = \sqrt{2r}$, implying that embeddings of this family with constant distortion and failure probability have a lower bound on the target dimension of $\Omega(\sqrt{rd})$, which is achieved by the embedding of Theorem 9.

5 Conclusions

We have shown that in the 2-norm, any constant-distortion embedding of a unitarily invariant set of d-dimensional states must have target dimension $\Omega(d)$, in contrast to the classical situation where an exponential reduction can be achieved. In the trace norm, the situation is somewhat better: d-dimensional states of rank r can be embedded in $O(\sqrt{rd})$ dimensions with constant distortion, but there is a lower bound of $\Omega(\sqrt{d}\frac{\|\rho-\sigma\|_1}{\|\rho-\sigma\|_2})$ dimensions on any constant distortion embedding that succeeds for the pairs of states $U\rho U^\dagger$ and $U\sigma U^\dagger$, for all unitary U. Although the trace distance is often the most physically relevant distance measure to consider, we also argued that for certain tasks, the 2-norm distance is in fact the relevant distance measure between states. This occurs when the basis in which the states were prepared is unknown or the measurement apparatus does not depend on the states to be distinguished.

The alert reader will have noticed that, in the case where one is interested in embedding a unitarily invariant set of states, the embedding might as well start by performing a random unitary. Furthermore, as any quantum channel can be represented as an isometry into a larger space followed by tracing out a subsystem, this makes any embedding seem somewhat similar to the embedding used in Theorem 9. But note that the latter embedding is subtly different, as it can be seen as performing a fixed isometry followed by a random unitary, rather than vice versa. Further analysis of this embedding might allow the gap between the upper and lower bounds in the trace norm to be closed.

Another open question is whether bounds could be obtained on the possible dimensionality reduction when multiple copies of the input state are available. For example, if a very large number of copies are allowed, tomography can be performed, the input state can be approximately determined, and the JL Lemma applied. Presumably, even for a lower number of copies, stronger dimensionality reduction is possible than in the single-copy case. One could also ask whether stronger dimensionality reduction can be achieved by allowing some additional classical information; for some results in this direction, see [10].

Acknowledgements

AWH was supported by the EC grant QESSENCE and the DARPA-MTO QuEST program through a grant from AFOSR. AM was supported by an EPSRC Postdoctoral Research Fellowship. AJS was supported by the Royal Society.

References

1. Aaronson, S.: Limitations of quantum advice and one-way communication. Theory of Computing 1, 1–28 (2004), quant-ph/0402095
2. Ambainis, A., Emerson, J.: Quantum t-designs: t-wise independence in the quantum world. In: Proc. 22nd Annual IEEE Conf. Computational Complexity, pp. 129–140 (2007), quant-ph/0701126

3. Aubrun, G., Szarek, S., Werner, E.: Hastings' additivity counterexample via Dvoretzky's theorem (2010), arXiv:1003.4925
4. Aubrun, G., Szarek, S., Werner, E.: Non-additivity of Renyi entropy and Dvoretzky's theorem. J. Math. Phys. 51, 022102 (2010), arXiv:0910.1189
5. Bartlett, S., Rudolph, T., Spekkens, R.: Classical and quantum communication without a shared reference frame. Phys. Rev. Lett. 91(2), 027901 (2003), quant-ph/0302111
6. Brinkman, B., Charikar, M.: On the impossibility of dimension reduction in ℓ_1. J. ACM 52(5), 766–788 (2005)
7. Buhrman, H., Cleve, R., Watrous, J., de Wolf, R.: Quantum fingerprinting. Phys. Rev. Lett. 87(16), 167902 (2001), quant-ph/0102001
8. Charikar, M., Sahai, A.: Dimension reduction in the ℓ_1 norm. In: Proc. 43rd Annual Symp. Foundations of Computer Science, pp. 551–560 (2002)
9. Cleve, R., Høyer, P., Toner, B., Watrous, J.: Consequences and limits of nonlocal strategies. In: Proc. 19th Annual IEEE Conf. Computational Complexity, pp. 236–249 (2004), quant-ph/0404076
10. Fawzi, O., Hayden, P., Sen, P.: From low-distortion norm embeddings to explicit uncertainty relations and efficient information locking (2010), arXiv:1010.3007
11. Gavinsky, D., Kempe, J., de Wolf, R.: Strengths and weaknesses of quantum fingerprinting. In: Proc. 21st Annual IEEE Conf. Computational Complexity, pp. 288–298 (2006), quant-ph/0603173
12. Harrow, A.W., Montanaro, A., Short, A.J.: Limitations on quantum dimensionality reduction (2010), arXiv:1012.2262
13. Hayden, P., Winter, A.: The fidelity alternative and quantum measurement simulation (2010), arXiv:1003.4994
14. Indyk, P.: Algorithmic applications of low-distortion geometric embeddings. In: Proc. 42nd Annual Symp. Foundations of Computer Science, pp. 10–33 (2001)
15. Indyk, P., Motwani, R.: Approximate nearest neighbors: towards removing the curse of dimensionality. In: Proc. 30th Annual ACM Symp. Theory of Computing, pp. 604–613 (1998)
16. Johnson, W., Lindenstrauss, J.: Extensions of Lipschitz mappings into a Hilbert space. Contemporary Mathematics 26, 189–206 (1984)
17. Kushilevitz, E., Nisan, N.: Communication Complexity. Cambridge University Press, Cambridge (1997)
18. Kushilevitz, E., Ostrovsky, R., Rabani, Y.: Efficient search for approximate nearest neighbor in high dimensional spaces. In: Proc. 30th Annual ACM Symp. Theory of Computing, pp. 614–623 (1998)
19. Matthews, W., Wehner, S., Winter, A.: Distinguishability of quantum states under restricted families of measurements with an application to quantum data hiding. Comm. Math. Phys. 291(3), 813–843 (2009), arXiv:0810.2327
20. Nielsen, M.A., Chuang, I.L.: Quantum computation and quantum information. Cambridge University Press, Cambridge (2000)
21. Sen, P.: Random measurement bases, quantum state distinction and applications to the hidden subgroup problem. In: Proc. 21st Annual IEEE Conf. Computational Complexity, p. 287 (2006), quant-ph/0512085
22. Watrous, J.: Theory of quantum information lecture notes (2008), http://www.cs.uwaterloo.ca/~watrous/quant-info/
23. Winter, A.: Quantum and classical message identification via quantum channels. In: Hirota, O. (ed.) Festschrift "A S Holevo 60", pp. 171–188 (2004), quant-ph/0401060

On Tree-Constrained Matchings and Generalizations*

Stefan Canzar[1], Khaled Elbassioni[2], Gunnar W. Klau[1], and Julián Mestre[3]

[1] Centrum Wiskunde & Informatica, Life Sciences Group, Science Park 123, 1098 XG, Amsterdam, The Netherlands
{stefan.canzar,gunnar.klau}@cwi.nl
[2] Max-Planck-Institut für Informatik, Algorithms and Complexity Dept., Saarbrücken, Germany
elbassio@mpi-inf.mpg.de
[3] School of Information Technologies, The University of Sydney, Australia

Abstract. We consider the following Tree-Constrained Bipartite Matching problem: Given two rooted trees $T_1 = (V_1, E_1)$, $T_2 = (V_2, E_2)$ and a weight function $w : V_1 \times V_2 \mapsto \mathbb{R}_+$, find a maximum weight matching $\mathcal{M}$ between nodes of the two trees, such that none of the matched nodes is an ancestor of another matched node in either of the trees. This generalization of the classical bipartite matching problem appears, for example, in the computational analysis of live cell video data. We show that the problem is $\mathcal{APX}$-hard and thus, unless $\mathcal{P} = \mathcal{NP}$, disprove a previous claim that it is solvable in polynomial time. Furthermore, we give a 2-approximation algorithm based on a combination of the local ratio technique and a careful use of the structure of basic feasible solutions of a natural LP-relaxation, which we also show to have an integrality gap of $2 - o(1)$. In the second part of the paper, we consider a natural generalization of the problem, where trees are replaced by partially ordered sets (posets). We show that the local ratio technique gives a $2k\rho$-approximation for the k-dimensional matching generalization of the problem, in which the maximum number of incomparable elements below (or above) any given element in each poset is bounded by ρ. We finally give an almost matching integrality gap example, and an inapproximability result showing that the dependence on ρ is most likely unavoidable.

1 Introduction

This paper contains both approximation and hardness results for the Tree-Constrained Bipartite Matching (TCBM) problem, a natural generalization of the classical maximum matching problem in bipartite graphs. The input of TCBM consists of a weighted bipartite graph $G = (V_1, V_2, E)$ and two rooted trees T_1 and T_2. The vertex set of T_i is V_i for $i = 1, 2$. The objective is to find a maximum weight matching such that the matched vertices in each tree are not

* A full version of this paper is available as the technical report [5].

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 98–109, 2011.

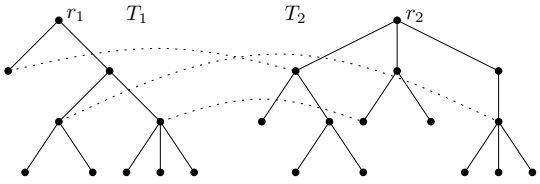


Fig. 1. Example of a feasible tree-constrained bipartite matching. For each matched pair of vertices, indicated by dotted lines, neither of their descendants are matched.

comparable; that is, if $u, v \in V_i$ are matched then u cannot be v's ancestor or vice-versa. Figure 1 illustrates the definition.

TCBM arises naturally in the computational analysis of live cell video data. Studying cell motility using live cell video data helps understand important biological processes, such as tissue repair, the analysis of drug performance, and immune system responses. Segmentation based methods for cell tracking typically follow a two stage approach (see [14] for a survey): The goal of the first *detection* step is to identify individual cells in each frame of the video independently. In a second step, the linkage of consecutive frames, and thus the *tracking* of a cell, is achieved by assigning cells identified in one frame to cells identified in the next frame. However, limited contrast and noise in the video sequence often leads to *over-segmentation* in the first stage: a single cell is comprised of several segments. A major challenge in this application domain is therefore the ability to distinguish biological cell division from over-segmentation.

Mosig et al. [10] and Xiao et al. [12] address this problem by proposing a novel approach for the *linkage* stage. As opposed to previous methods, they match *sets of segments* between neighboring frames rather than singletons, where the segment sets correspond to the nodes of an agglomerative hierarchical clustering tree. A subsequent bipartite matching between the nodes of the clustering trees corresponding to neighboring frames integrates the identification of a cell as a set of segments and the tracking of the cell between two different video frames. Since segment sets representing different cells in the same frame must be disjoint, no two nodes on any root-to-leaf path can be matched at the same time, leading to an instance of TCBM . To assess the quality of such a tree-constrained matching, Mosig et al. consider the relative overlap of the convex hulls of matched segment sets. This *cosegmentation* via TCBM promises to be usefull also in other bioimaging applications, for example in protein-colocalization studies [12].

To track cells not only between two consecutive frames but across a whole video sequence, the bipartite graphs have to be concatenated to a *cell connection graph*, as introduced in [13]. In [10] this is done by solving a standard maximum weight bipartite matching problem for each frame i, which is at the intersection between the tree-constrained alignment of frames $i-1$ and i, and the alignment of frames i and $i+1$. Concluding, the authors mention post-processing the cell connection graph as a promising improvement. Therefore, the generalization of

bipartite tree-constrained matching to a tree-constrained matching in a k-partite k-uniform hypergraph is an important problem for this application. By linking more than two frames at the same time, over-segmentation and cell-division can be distinguished by taking into account the cell behavior over a larger time-scale.

Another natural generalization of the problem is obtained by replacing trees by partially ordered sets (posets), because they permit the representation of alternative clustering hierarchies. For example, various meaningful distance measures between (sets of) segments could make it necessary to assign them to multiple parent clusters. In particular, noise in the video data may make it difficult to determine a unique tree.

Mosig et al. [10] present a linear programming formulation for TCBM and claim that the constraint matrix is totally unimodular, which would imply that the problem is solvable in polynomial time [11]. We disprove this statement by showing an instance with a fractional vertex and proving that the problem is in fact $\mathcal{NP}$-hard and even hard to approximate within a constant. Thus, conditional on $\mathcal{P} \neq \mathcal{NP}$, there is no polynomial time algorithm for our problem.

TCBM and its generalization to k trees are special cases of the maximum weighted independent set (MWIS) problem on 2-interval graphs and k-interval graphs, respectively. The connection is given by ordering the leaves of the trees by depth-first search and identifying each node with the interval of leaves below it. In fact, TCBM captures precisely the subclass of 2-*union* graphs (the first interval of a 2-interval cannot intersect the second interval of another 2-interval) where the two interval families are *laminar* (any two intervals are either disjoint or one is nested in the other). In [3] the fractional local ratio technique was developed and applied to MWIS in k-interval graphs to get a $2k$-approximation algorithm. This result immediately implies a 4-approximation for TCBM.

1.1 Our Results

In this paper, we give a 2-approximation algorithm for TCBM, improving upon the 4-approximation that follows from the work of Bar-Yehuda et al. [3]. Our method is based on a combination of the local ratio technique and a careful use of the structure of basic feasible solutions of a natural LP-relaxation. In Section 2.1 we show a 3-approximation based on fractional local ratio and prove that this is the best guarantee the fractional local ratio technique alone can deliver when rounding one coordinate at a time. The main difference between our approach and that of Bar-Yehuda et al. [3] is that we round basic feasible solutions. This allows us to exploit their structure in the analysis in order to get better approximation guarantees. In Section 2.2, we show how to get a 2-approximation and give an instance for which our LP-relaxation has an integrality gap of $2 - o(1)$. In Section 2.3, we show that the problem is $\mathcal{APX}$-hard. Our results imply that the MWIS problem on 2-union graphs in which both families of intervals are *laminar*, is still $\mathcal{APX}$-hard, but can be approximated within a factor of 2.

In Section 3, we consider the k-dimensional generalization of the problem to posets. In this case, the natural LP-relaxation has an exponential number of constraints, but admits an alternative linear-size LP-formulation. Even though the

result of Bar-Yehuda et al. [3] does not apply directly to the poset case, we show that the fractional local ratio technique yields a $2\sum_{i=1}^{k}\rho(\mathcal{P}_i)$-approximation here, where $\rho(\mathcal{P}_i)$ is the maximum number of incomparable elements below (or above) any given element in poset $\mathcal{P}_i$. We also give an example which shows that the integrality gap of the LP-relaxation is tight within almost a factor of 2. Finally, Section 3.2 gives a reduction from *Maximum Label Cover* showing that the 2-dimensional matching problem with poset constraints is hard to approximate within a factor of $2^{\log^{1-\epsilon}\rho}$, for any $\epsilon > 0$, where $\rho = \max\{\rho(\mathcal{P}_1), \rho(\mathcal{P}_2)\}$, unless $\mathcal{NP} \subseteq DTIME(n^{\text{polylog}\, n})$. Note that the k-dimensional version of TCBM includes as a special case the k-dimensional matching problem, and hence [8] is NP-hard to approximate within a factor of $O(k/\log k)$.

2 Matching Trees

In this section we focus on the basic TCBM problem, formally defined as follows:

Definition 1 (Tree-constrained bipartite matching problem, TCBM). *Given two rooted trees $T_1 = (V_1, E_1)$, $T_2 = (V_2, E_2)$ with roots $r_1 \in V_1$ and $r_2 \in V_2$, and a weight function $w : V_1 \times V_2 \mapsto \mathbb{R}_+$, find a maximum weight matching $\mathcal{M}$ in the complete bipartite graph $G = (V_1, V_2, E)$ with edge weights induced by w, such that $(u, v) \in \mathcal{M}$ implies $(u', v') \notin \mathcal{M}$, if u' is a descendant of u or v' is a descendant of v.*

Consider an instance (T_1, T_2, w) of TCBM with $T_1 = (V_1, E_1)$ and $T_2 = (V_2, E_2)$. Let $r_1 \in V_1$ and $r_2 \in V_2$ denote the roots and $\mathcal{L}_1 \subset V_1$ and $\mathcal{L}_2 \subset V_2$ denote the set of leaves of the trees, respectively. For two vertices p and q denote by $[p, q]$ the path in T_1 (or T_2) between p and q. For each $p_1 \in V_1$ and $p_2 \in V_2$ let $x_{p_1,p_2} \in \{0, 1\}$ be a variable which takes value 1 if and only if p_1 is matched to p_2, i.e. $(p_1, p_2) \in \mathcal{M}$. Consider the following LP-relaxation of the problem.

$$\max \sum_{p_1 \in V_1, p_2 \in V_2} w_{p_1,p_2} x_{p_1,p_2} \qquad \text{(P)}$$

$$\text{s.t.} \sum_{p_1 \in [r_1,\ell], p_2 \in V_2} x_{p_1,p_2} \leq 1 \qquad \text{for all } \ell \in \mathcal{L}_1 \qquad (1)$$

$$\sum_{p_1 \in V_1, p_2 \in [r_2,\ell]} x_{p_1,p_2} \leq 1 \qquad \text{for all } \ell \in \mathcal{L}_2 \qquad (2)$$

$$x_{p_1,p_2} \geq 0 \qquad \text{for all } p_1 \in V_1,\ p_2 \in V_2\ .$$

For the purpose of the analysis of the algorithm we will consider a more general LP formulation $(P_{\mathbf{b}})$, in which the right hand sides of constraints (1) and (2) are replaced by some b_l, $0 \leq b_\ell \leq 1$, for all $\ell \in \mathcal{L}_1 \cup \mathcal{L}_2$. We will use $x(F)$ to denote $\sum_{e \in F} x_e$ for a subset of the edges $F \subseteq E$ in the complete bipartite graph G and $\delta(p)$ to denote the set of edges in E incident to a vertex p.

Overview of the technique. Our main tool will be the *fractional local ratio technique*, applied (recursively) to an optimal *basic* feasible solution (BFS) x^* of the above LP, where the base case of the recursion involves the computation of maximum weight independent sets in interval graphs. As long as there is a pair $(p, q) \in V_1 \times V_2$ with "small fractional ratio", that is the total contribution $\sum_{(p',q')} x^*_{p'q'}$, over all pairs (p', q') that are in conflict with (p, q), is at most 2, we can take this pair (p, q) into the solution and recurse on a new instance with reduced weights. Otherwise (if there is no such pair), we can use a structural result about the BFS's (Lemmas 1 and 3) to reduce the problem to computing maximal independent sets in interval graphs. In the next subsection, we prove this structural result, and show the fractional local ratio approach alone can give a 3-approximation, but not more. We then extend this to a 2-approximation in Section 2.2.

2.1 A 3-Approximation by Fractional Local Ratio

For a feasible fractional solution $\mathbf{x}$ to linear program $(P_\mathbf{b})$ and $i \in \{1, 2\}$ we denote by $\mathcal{L}_i(\mathbf{x})$ the set of nodes $\ell \in V_i$ with $x(\delta(\ell)) > 0$ and $x(\delta(p)) = 0$ for all descendants p of ℓ in T_i.

Lemma 1. *For any basic feasible solution* $\mathbf{x}$ *to linear program* $(P_\mathbf{b})$ *one of the following holds:*

(a) there exist nodes $\ell_1 \in \mathcal{L}_1(\mathbf{x})$ *and* $\ell_2 \in \mathcal{L}_2(\mathbf{x})$ *such that* $x_{\ell_1,\ell_2} > 0$*, or*
(b) for every $x_{p_1,p_2} > 0$ *either* $p_1 \in \mathcal{L}_1(\mathbf{x})$ *and* $x_{p_1,p_2'} = 0$*, for all* $p_2' \neq p_2$*, or* $p_2 \in \mathcal{L}_2(\mathbf{x})$ *and* $x_{p_1',p_2} = 0$*, for all* $p_1' \neq p_1$*.*

Proof. Assume (a) does not hold, i.e. $x_{\ell_1,\ell_2} = 0$ for all $\ell_1 \in \mathcal{L}_1(\mathbf{x})$ and $\ell_2 \in \mathcal{L}_2(\mathbf{x})$. For an arbitrary node $p_1 \in V_1$, any two constraints (1) for leaves in the subtree rooted at p_1 linearly depend on the set of non-negativity constraints for variables x_{p_1',p_2}, with $p_2 \in V_2$ and p_1' being a descendant of p_1. Thus, every $\ell_1 \in \mathcal{L}_1(\mathbf{x})$ implies at most one linear independent tight constraint (1) for one of the leaves in its subtree. Since a symmetric argument applies to nodes $\ell_2 \in \mathcal{L}_2(\mathbf{x})$, the number of linearly independent tight constraints (1), (2) for a given basic feasible solution $\mathbf{x}$ is at most $|\mathcal{L}_1(\mathbf{x})| + |\mathcal{L}_2(\mathbf{x})|$.

On the one hand, since $x(\delta(\ell)) > 0$ for all $\ell \in \mathcal{L}_1(\mathbf{x}) \cup \mathcal{L}_2(\mathbf{x})$ and since no positive edge between two leaves exists, for every node in $\mathcal{L}_1(\mathbf{x}) \cup \mathcal{L}_2(\mathbf{x})$ there is at least one distinct non-zero edge incident to it. On the other hand, in a basic feasible solution the number of non-zero variables is at most the number of linearly independent tight constraints (1), (2), which in turn is at most $|\mathcal{L}_1(\mathbf{x})| + |\mathcal{L}_2(\mathbf{x})|$. Therefore, exactly one distinct non-zero edge is incident to every node in $\mathcal{L}_1(\mathbf{x}) \cup \mathcal{L}_2(\mathbf{x})$ and all other edges must be 0. □

Based on this property of basic feasible solutions to $(P_\mathbf{b})$ we next show that there always exists an edge with local ratio at most 3. For this, let $N[(p, q)]$ be the set of edges $(p', q') \in E$ that are in conflict with edge (p, q) (including (p, q) itself), i.e.,

$$N[(p,q)] = \{(p',q') \in E \mid p' \in [r_1, p] \vee p \in [r_1, p'] \vee q' \in [r_2, q] \vee q \in [r_2, q']\} \ .$$

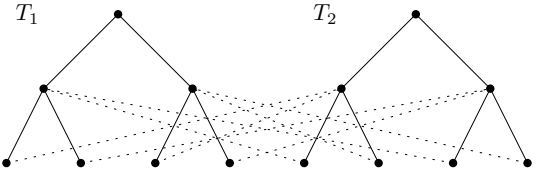


Fig. 2. Tight example for the fractional local ratio of an edge, $k = 3$

Lemma 2. *Let* $\mathbf{x}$ *be a basic feasible solution to* $(P_{\mathbf{b}})$*. There exists an edge* $(p, q) \in E$ *with* $x_{p,q} > 0$ *such that* $\sum_{(p',q') \in N[(p,q)]} x_{p',q'} \leq 3$.

The following lemma shows that we can apply Lemma 2 inductively in a local ratio framework.

Lemma 3. *Let* $\mathbf{x}$ *be a basic feasible solution to* $(P_{\mathbf{b}})$*. For an edge* (p, q) *with* $x_{p,q} > 0$*, let* $\mathbf{b}'$ *be such that* $b'_\ell = b_\ell - x_{p,q}$ *if* $p \in [r_1, \ell]$*, respectively* $q \in [r_2, \ell]$*, and* $b'_\ell = b_\ell$ *otherwise. Then* $\mathbf{x}'$ *with* $x'_{p,q} = 0$ *and* $x'_{e'} = x_{e'}$ *for* $e' \neq (p, q)$*, is a basic feasible solution to* $(P_{\mathbf{b}'})$.

Proof. Since the right hand side of constraints in $(P_{\mathbf{b}'})$ for all root to leaf paths through p, respectively q, are decreased by exactly $x_{p,q}$, $\mathbf{x}'$ is still feasible for $(P_{\mathbf{b}'})$. Suppose that $\mathbf{x}'$ can be expressed as a convex combination of two feasible solutions $\mathbf{y}$ and $\mathbf{z}$ (of $(P_{\mathbf{b}'})$). Then setting $y_{p,q} = x_{p,q}$ and $z_{p,q} = x_{p,q}$ would yield a convex combination of $\mathbf{x}$ (in $(P_{\mathbf{b}})$), a contradiction. Thus, $\mathbf{x}'$ must be a basic feasible solution to $(P_{\mathbf{b}'})$. □

Now by applying the fractional local ratio technique of [3] we immediately obtain the following result.

Theorem 1. *There is a 3-approximation algorithm for TCBM.*

We next give an example instance that shows that using the fractional local ratio method, our approximation guarantee is tight.

Lemma 4. *There exists an instance of TCBM such that the optimal solution to* (P) *is unique and the fractional local ratio of every edge is at least* $3 - o(1)$.

Proof. Let T_1 and T_2 be trees of height two (i.e., they have three levels) where each internal node has $k - 1$ children. The edges E connecting the nodes of T_1 and T_2 are as follows. Let x be a leaf of T_1. If x is the ith child of its *parent* in T_1, we connect x with the ith child of the *root* of T_2. We connect in a similar fashion the leaves of T_2 with the children of the root of T_1. All edges have a weight of 1. Figure 2 illustrates the construction for $k = 3$.

It can be verified that the optimal fractional solution must set the value of every edge to $\frac{1}{k}$. Let (u, v) be an edge where u is a leaf and v is an internal node. Notice that (u, v) is in conflict with $k - 1$ edges incident on children of v, $k - 2$ edges incident on v and $k - 1$ edges incident on u's parent. Since each edge carries a fractional contribution of $\frac{1}{k}$ their combined fractional value is $3 - \frac{4}{k}$. □

Algorithm 1. Tree-Matching$(F, \mathbf{w})$

Require: A BFS $\mathbf{x}$ to $(P_{\mathbf{b}})$, $0 \leq \mathbf{b} \leq 1$, edge set F, and weights on the edges $\mathbf{w}$.

1: **if** $F = \emptyset$ **then**
2: **return** $\emptyset$
3: Define $F_0 = \{e \in F \mid w_e \leq 0\}$
4: **if** $F_0 \neq \emptyset$ **then**
5: **return** Tree-Matching$(F \setminus F_0, w)$
6: **if** $\forall e \in F : x(N[e]) > 2$ **then**
7: $\mathcal{I}_1 \leftarrow$ MWIS in IG(T_1) w.r.t. weights $\bar{w}^1_p = \max_{(p,q)\in F} w_{p,q}$
8: $\mathcal{I}_2 \leftarrow$ MWIS in IG(T_2) w.r.t. weights $\bar{w}^2_q = \max_{(p,q)\in F} w_{p,q}$
9: **if** $\bar{w}^1(\mathcal{I}_1) \geq \bar{w}^2(\mathcal{I}_2)$ **then**
10: **return** $\bigcup_{p\in\mathcal{I}_1} \{\text{argmax}_{(p,q)\in F} w_{p,q}\}$
11: **else**
12: **return** $\bigcup_{q\in\mathcal{I}_2} \{\text{argmax}_{(p,q)\in F} w_{p,q}\}$
13: **else**
14: Let $e' \in F$ be s.t. $x(N[e']) \leq 2$
15: Decompose $\mathbf{w}$ by $\mathbf{w} = \mathbf{w}^1 + \mathbf{w}^2$ where $w^1_e := \begin{cases} w_{e'} & \text{if } e \in N[e'], \\ 0 & \text{otherwise.} \end{cases}$
16: $\mathcal{M}' \leftarrow$ Tree-Matching$(F, \mathbf{w}^2)$
17: **if** $N[e'] \cap \mathcal{M}' = \emptyset$ **then**
18: **return** $\mathcal{M} = \mathcal{M}' \cup \{e'\}$
19: **else**
20: **return** $\mathcal{M} = \mathcal{M}'$

2.2 A 2-Approximation

The algorithm requires a basic feasible solution $\mathbf{x}$ to (P) and is initially called with an edge set F, in which all edges e with $x_e = 0$ have been removed. The idea is to recurse in a local ratio manner as long as we can find an edge with local ratio at most two. If this is not possible anymore, we exploit the specific structure of basic feasible solutions by computing *maximum weight independent sets* (MWIS) in the interval graphs IG(T_1) and IG(T_2) induced by the two trees: For $i \in \{1, 2\}$, IG(T_i) is the interval graph obtained by ordering the leaves of T_i by depth-first search and identifying each node of T_i with the interval of leaves below it. As usual, we define the maximum (in lines 7-8) over an empty set to be 0. It is not difficult to see that the matching $\mathcal{M}$ returned by the algorithm is feasible for (P). It remains to assess the quality of this solution.

Theorem 2. *Let* $\mathbf{x}$ *be a basic feasible solution to linear program* $(P_{\mathbf{b}})$. *The matching* $\mathcal{M}$ *returned by Algorithm 1 satisfies* $w(\mathcal{M}) \geq \frac{1}{2} \cdot \mathbf{w} \cdot \mathbf{x}$.

Proof. The proof is by induction on the number of edges having positive weight. In the base case either there is no edge with positive weight (lines 1-2) or no edge e' in line 14 exists. In the former case, the induction hypothesis clearly holds. In the latter case, according to Lemma 1 the non-zero edges can be partitioned into sets F_1 and F_2, containing edges with one endpoint in $\mathcal{L}_1(\mathbf{x})$, respectively one endpoint in $\mathcal{L}_2(\mathbf{x})$. For $i \in \{1, 2\}$, let

$$\mathcal{P}_{IG}(T_i) := \left\{ \mathbf{y} \in \mathbb{R}_+^{V_i} \ : \sum_{p \in V_i : p \in [r_i, \ell]} y_p \leq 1, \quad \text{for all } \ell \in \mathcal{L}_i \right\}$$

be the fractional independent set polytope in the interval graph represented by tree T_i. It is well-known (see e.g. [7]) that $\mathcal{P}_{IG}(T_i)$ is integral. Given the basic feasible solution $\mathbf{x}$ of $(P_\mathbf{b})$, define $\mathbf{y}^i \in \mathbb{R}^{V_i}$, for $i \in \{1, 2\}$, as follows: $y^i_p = \sum_{q:(p,q)} x_{p,q}$, for $p \in V_i$. The feasibility of $\mathbf{x}$ to $(P_\mathbf{b})$ implies that $\mathbf{y}^i \in \mathcal{P}_{IG}(T_i)$, for $i \in \{1, 2\}$. Let $\mathcal{I}_1$ and $\mathcal{I}_2$ be the independent sets computed in steps 7 and 8 of the algorithm, and $\mathcal{M}'$ be the matching computed in step 10 or 12. Then

$$\begin{aligned}
\mathbf{w} \cdot \mathbf{x} &= \sum_{(p,q) \in F_1} w_{p,q} x_{p,q} + \sum_{(p,q) \in F_2} w_{p,q} x_{p,q} \\
&\leq \sum_{p \in V_1} \bar{w}^1_p \sum_{(p,q) \in F_1} x_{p,q} + \sum_{q \in V_2} \bar{w}^2_q \sum_{(p,q) \in F_2} x_{p,q} \qquad (3) \\
&= \sum_{p \in V_1} \bar{w}^1_p y_p + \sum_{q \in V_2} \bar{w}^2_q y_q \leq \max_{\mathbf{y}' \in \mathcal{P}_{IG}(T_1)} \bar{\mathbf{w}}^1 \cdot \mathbf{y}' + \max_{\mathbf{y}' \in \mathcal{P}_{IG}(T_2)} \bar{\mathbf{w}}^2 \cdot \mathbf{y}' \qquad (4) \\
&= \bar{w}(\mathcal{I}_1) + \bar{w}(\mathcal{I}_2) \leq 2 \cdot \max\{\bar{w}(\mathcal{I}_1), \bar{w}(\mathcal{I}_2)\} = 2 \cdot w(\mathcal{M}'). \qquad (5)
\end{aligned}$$

Inequality (3) follows from the definition of the weights $\bar{\mathbf{w}}^1$ and $\bar{\mathbf{w}}^2$ in lines 7, 8; inequalities (4) and (5) follow respectively from the fact that $\mathbf{y}^i \in \mathcal{P}_{IG}(T_i)$, and the integrality of $\mathcal{P}_{IG}(T_i)$, for $i \in \{1, 2\}$; and the last equality is due to the choice the algorithm makes in line 9. Note that the matchings constructed in lines 10 respectively 12 are feasible solutions to the TCBM problem. Indeed, due to the structure of a basic feasible solution (Lemma 1 which will also continue to hold inductively by Lemma 3), the edges that induce an independent set in one tree end in leaves of the second tree and therefore do not conflict.

We next prove the inductive step (the rest of the argument is the same as in [2]; we include it for completeness). If F_0 is non-empty in step 4, extending $\mathbf{w}$ in the induction hypothesis by the non-positive components that were deleted in line 5 cannot make the inequality invalid. Let $\mathbf{y}'$ and $\mathbf{y}$ be the characteristic vectors of matchings $\mathcal{M}'$ and $\mathcal{M}$, obtained in lines 16 and lines 18-20, respectively. Let e' be the edge chosen in line 14. By the decomposition of $\mathbf{w}$ in line 15, $\mathbf{w}^2$ implies a smaller number of edges with positive weight than $\mathbf{w}$. By the induction hypothesis, $\mathbf{w}^2 \cdot \mathbf{y}' \geq \frac{1}{2} \cdot \mathbf{w}^2 \cdot \mathbf{x}$. From $w^2_{e'} = 0$ it also follows that $\mathbf{w}^2 \cdot \mathbf{y} \geq \frac{1}{2} \cdot \mathbf{w}^2 \cdot \mathbf{x}$. Since at least one edge in $N[e']$ is in $\mathcal{M}$ and $x(N[e']) \leq 2$ (line 14), it also holds that $\mathbf{w}^1 \cdot \mathbf{y} \geq \frac{1}{2} \cdot \mathbf{w}^1 \cdot \mathbf{x}$. The claim follows. □

We conclude this section by giving an example showing that the integrality gap of (IP) matches the approximation factor attained by our algorithm.

Lemma 5. *The integrality gap of* (IP) *is* $2 - o(1)$.

Proof. Our bad instance consists of two stars of height 1 (i.e., they have two levels) where each internal node has $k - 1$ children. The leaf nodes of one star are connected to the root node of the other star. An integral solution can pick at most one edge. However, a fractional solution that sets the value of every edge to $\frac{1}{k}$ is feasible and has value $2 - \frac{2}{k}$. □

2.3 Hardness and Inapproximability Results

In this section we prove hardness of tree-constrained bipartite matching even if the weights in the matching are restricted to the values zero and one. Subsequently, we show by an approximation-preserving reduction from a restricted MAX SAT version that TCBM does not admit a PTAS.

Theorem 3. *For an instance* $I = (T_1, T_2, w)$ *of* TCBM*, with* $w : V_1 \times V_2 \mapsto \{0, 1\}$*, and an integer* k*, it is* $\mathcal{NP}$*-complete to decide whether there exists a tree-constrained bipartite matching of weight at least* k*.*

Proof. Clearly, the problem is in $\mathcal{NP}$. To prove that it is $\mathcal{NP}$-hard, we devise a polynomial-time reduction τ from SAT, the problem of deciding whether a Boolean formula has a satisfying assignment. Given a CNF formula ϕ with m clauses over n variables, we construct a TCBM instance $I = (T_1, T_2, w)$, such that ϕ is satisfiable if and only if I admits a matching of weight $n + m$.

Tree T_1 is the star S_{n+m}, with one leaf per variable and clause. Depth-2 tree T_2 has for each literal occurring in ϕ a node at level 1. Such a node, corresponding to some literal l_k, has a child node for each occurrence of literal l_k in ϕ. What remains is the definition of the weight function w. For each leaf u in T_1, representing some variable x_i, we define $w(u, v) := 1$ and $w(u, v') := 1$, where v, v' are level-1 nodes in T_2 that correspond to literals x_i and $\neg x_i$, respectively. For each leaf u in T_1, representing some clause C_i of ϕ, we set, for all literals l_j occurring in C_i, $w(u, v) := 1$, where v is a child node of a level-1 node in T_2 that corresponds to literal l_j. Hereby we pick distinct child nodes v in T_2, such that each level-2 node is incident to exactly one edge of weight 1. All remaining weights are set to 0. See Figure 3 for an illustrative example of this construction.

Finally, it can be shown that ϕ is satisfiable if and only if $\tau(\phi)$ admits a matching $\mathcal{M}$ of weight $m + n$. □

We next prove that TCBM is $\mathcal{APX}$-hard. The reduction is made from 3-OCC-MAX 2SAT, a restricted form of MAX SAT, where each clause contains two literals and each variable occurs at most three times.

Theorem 4 ([4]). *For any* $\epsilon > 0$ *it is* $\mathcal{NP}$*-hard to decide whether an instance of* 3-OCC-MAX 2SAT *with* $2016n'$ *clauses (and* $1344n'$ *variables) has a truth assignment that satisfies at least* $(2012 - \epsilon)n'$ *clauses, or at most* $(2011 + \epsilon)n'$*.*

Theorem 5. *For any* $\epsilon > 0$*, it is* $\mathcal{NP}$*-hard to approximate* TCBM *within factor* $6044/6043 - \epsilon$*.*

Proof. Our reduction τ' from 3-OCC-MAX 2SAT to TCBM differs from reduction τ described in the proof of Theorem 3 only in the definition of the weight function w. For leaves u in T_1 representing some variable x_i and level-1 nodes v in T_2 corresponding to literal x_i or $\neg x_i$, we set $w(u, v) := 3$, and leave w unchanged otherwise. Then it is easy to see that the maximum number of satisfiable clauses in ϕ is k if and only if the maximum weight of a tree-constrained matching in $\tau'(\phi)$ is $3n + k$. Since the instance constructed in [4] uses $1344n'$ variables, $k = (2012 - \epsilon)n'$ and $k = (2011 + \epsilon)n'$ correspond to tree constrained matchings of weight $(6044 - \epsilon)n'$ and $(6043 + \epsilon)n'$ respectively. □

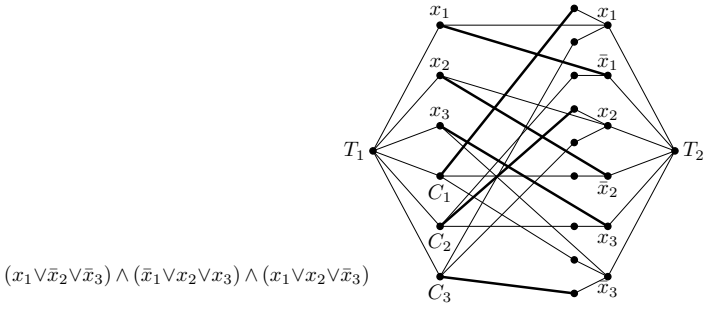


Fig. 3. Left: SAT instance in conjunctive normal form. Right: Transformed tree-constrained bipartite matching instance, only edges with unit weight are shown. A maximum weight tree-constrained bipartite matching, which corresponds to a satisfying truth assignment, is shown in bold.

3 Matching Posets

Definition 2 (Poset-constrained k-partite matching problem, k-PCM). *Given k posets $\mathcal{P}_i = (V_i, \preceq_i)$, $1 \leq i \leq k$, and a weight function $w : V_1 \times V_2 \times \cdots \times V_k \mapsto \mathbb{R}_+$, find a maximum weight k-dimensional matching $\mathcal{M}$ in the complete k-partite k-uniform hypergraph $H = (V_1, \ldots, V_k, E)$ with hyper-edge weights induced by w, such that $(p_1, \ldots, p_k) \in \mathcal{M}$ implies $(q_1, \ldots, q_k) \notin \mathcal{M}$ if there exists $1 \leq i \leq k$ with q_i being comparable to p_i in $\mathcal{P}_i$.*

3.1 A Fractional Local Ratio Algorithm

Unlike the tree case, this problem cannot be directly reduced to MWIS in k-interval graphs, and therefore, the $2k$-approximation of Bar-Yehuda et al. [3] does not readily apply. However, we show that the fractional local ratio technique can still be used to solve the poset case. We work with the following linear programming formulation.

$$\max \sum_{p \in E} w_p x_p \qquad \text{(MP)}$$

$$\text{s.t.} \sum_{p : p_i \in C} x_p \leq 1 \qquad \forall \text{ chain } C \text{ in } \mathcal{P}_i, i = 1, \ldots, k \qquad (6)$$

$$x_p \geq 0 \qquad \forall\, p \in E \ .$$

We remark that even though the above linear program is exponentially large, there is a simple separation oracle based on the polynomial time algorithm for computing a longest path in an acyclic directed graph. In fact, there is an alternative linear-size LP formulation, see [5].

As we did before in the tree case, the crux of the analysis is to show that there is an edge $p \in E$ with low fractional local ratio. Our bound will depend on the *maximum upward independence number* of the individual posets.

Definition 3. *For a given poset $\mathcal{P} = (V, \preceq)$, the maximum upward independence number of $\mathcal{P}$, denoted by $\rho(\mathcal{P})$ is defined as*

$$\max_{v \in V} \text{ size of largest antichain in } (\{u \in V : v \preceq u\}, \preceq).$$

We show that the fractional local ratio of any feasible solution is bounded by the maximum upward independence number of the posets in our instance.

Lemma 6. *Let $\mathbf{x}$ be a feasible solution to (MP). There is some $p \in E$ such that*

$$\sum_{q \,:\, \exists i \,\bullet\, q_i \preceq_i p_i \vee p_i \preceq_i q_i} x_q \leq 2 \sum_i \rho(\mathcal{P}_i).$$

Notice that if we consider the poset $\mathcal{P}$ induced by some tree T, the maximum upward independence number is 1. This is because for any vertex v of T, the poset induced by $\{u \in T : v \preceq u\}$ is a total order; namely, the path from v to the root of the tree. Using the fractional local ratio framework of Bar-Yehuda et al. [3] we immediately obtain the following result.

Theorem 6. *There is a $2\sum_i \rho(\mathcal{P}_i)$ approximation algorithm for k-PCM.*

It can be shown that the dependency on $\sum_i \rho(\mathcal{P}_i)$ in the approximation ratio is necessary for any algorithm based on the linear program (MP).

Lemma 7. *There are instances of k-PCM where the integrality gap of (MP) is $\left(1 - \frac{1}{k}\right)\sum_i \rho(\mathcal{P}_i)$ for arbitrary large k.*

3.2 Hardness

In this section we show that the dependence of the approximation factor on the maximum poset width $\rho(\mathcal{P})$ is unavoidable, by showing that, under plausible complexity assumptions, 2-PCM is hard to approximate within $2^{\log^{1-\epsilon}\rho}$ for any $\epsilon > 0$, where ρ is the maximum width of the posets.

We will use a reduction from the *maximum label-cover* problem [1]. For convenience we use the following equivalent definition [9].

Definition 4 (MAXREP). *Given a bipartite graph $G = (A, B, E)$, with a partition of A and B into k disjoint sets $A_1, \ldots, A_k$ and $B_1, \ldots, B_k$, respectively, find subsets of vertices $A' \subseteq A$ and $B' \subseteq B$, such that, $|A' \cap A_i| \leq 1$ and $|B' \cap B_i| \leq 1$, for $i = 1, \ldots, k$, so as to maximize the number of edges*

$$E(A', B') := \{\{a, b\} \in E \ : A' \cap A_i = \{a\} \text{ and } B' \cap B_j = \{b\} \text{ for some } i,\ j\}.$$

Theorem 7 ([6, 9]). MAXREP *on a graph with $|A| = |B| = n$ cannot be approximated within a factor of $2^{\log^{1-\epsilon} n}$, for any $\epsilon > 0$, unless $NP \subseteq DTIME(n^{\text{polylog}\, n})$.*

Theorem 8. *For any $\epsilon > 0$ and $\rho := \max\{\rho(\mathcal{P}_1), \rho(\mathcal{P}_2)\}$ there is no $2^{\log^{1-\epsilon}\rho}$-factor approximation for 2-PCM unless $NP \subseteq DTIME(n^{\text{polylog}\, n})$.*

Acknowledgments. We thank Axel Mosig for introducing us to the problem and for helpful discussions. We also thank an anonymous reviewer of an earlier version for pointing out the connection between our problem and the work of Bar-Yehuda et al. [3]. Thanks also to Yuk Hei Chan for helpful discussions.

References

[1] Arora, S., Babai, L., Stern, J., Sweedyk, Z.: The hardness of approximate optima in lattices, codes, and systems of linear equations. Journal of Computer and System Sciences 54(2), 317–331 (1997)
[2] Bar-Noy, A., Bar-Yehuda, R., Freund, A., Naor, J., Schieber, B.: A unified approach to approximating resource allocation and scheduling. Journal of the ACM 48, 1069–1090 (2001)
[3] Bar-Yehuda, R., Halldórsson, M.M., Naor, J., Shachnai, H., Shapira, I.: Scheduling split intervals. SIAM Journal on Computing 36(1), 1–15 (2006)
[4] Berman, P., Karpinski, M.: On some tighter inapproximability results. In: Wiedermann, J., Van Emde Boas, P., Nielsen, M. (eds.) ICALP 1999. LNCS, vol. 1644, pp. 200–209. Springer, Heidelberg (1999)
[5] Canzar, S., Elbassioni, K., Klau, G.W., Mestre, J.: On tree-constrained matchings and generalizations. Technical Report MAC1102, Centrum Wiskunde & Informatica (CWI), Amsterdam, the Netherlands (2011)
[6] Feige, U., Lovász, L.: Two-prover one-round proof systems: their power and their problems. In: Proc. of STOC, pp. 733–744. ACM, New York (1992)
[7] Grötschel, M., Lovász, L., Schrijver, A.: Geometric Algorithms and Combinatorial Optimization. Springer, New York (1988)
[8] Hazan, E., Safra, S., Schwartz, O.: On the complexity of approximating k-dimensional matching. In: Arora, S., Jansen, K., Rolim, J.D.P., Sahai, A. (eds.) RANDOM 2003 and APPROX 2003. LNCS, vol. 2764, pp. 83–97. Springer, Heidelberg (2003)
[9] Kortsarz, G.: On the hardness of approximating spanners. Algorithmica 30(3), 432–450 (2001)
[10] Mosig, A., Jäger, S., Wang, C., Nath, S., Ersoy, I., Palaniappan, K., Chen, S.-S.: Tracking cells in life cell imaging videos using topological alignments. Algorithms for Molecular Biology 4, 10 (2009)
[11] Wolsey, L.A., Nemhauser, G.L.: Integer and Combinatorial Optimization. Wiley Interscience, Hoboken (1999)
[12] Xiao, H., Li, Y., Du, J., Mosig, A.: Ct3d: Tracking Microglia Motility in 3D Using a Novel Cosegmentation Approach. Bioinformatics 27(4), 564–571 (2011)
[13] Yang, F., Mackey, M.A., Ianzini, F., Gallardo, G., Sonka, M.: Cell segmentation, tracking, and mitosis detection using temporal context. In: Duncan, J.S., Gerig, G. (eds.) MICCAI 2005. LNCS, vol. 3749, pp. 302–309. Springer, Heidelberg (2005)
[14] Zimmer, C., Zhang, B., Dufour, A., Thebaud, A., Berlemont, S., Meas-Yedid, V., Marin, J.-C.O.: On the digital trail of mobile cells. Signal Processing Magazine 23(3), 54–62 (2006)

Tight Bounds for Linkages in Planar Graphs

Isolde Adler[1], Stavros G. Kolliopoulos[2], Philipp Klaus Krause[1], Daniel Lokshtanov[3], Saket Saurabh[4], and Dimitrios Thilikos[2]

[1] Goethe-Universität, Frankfurt am Main
[2] National and Kapodistrian University of Athens
[3] University of California, San Diego
[4] Institute of Mathematical Sciences, Chennai

Abstract. The Disjoint-Paths Problem asks, given a graph G and a set of pairs of terminals $(s_1, t_1), \ldots, (s_k, t_k)$, whether there is a collection of k pairwise vertex-disjoint paths linking s_i and t_i, for $i = 1, \ldots, k$. In their $f(k) \cdot n^3$ algorithm for this problem, Robertson and Seymour introduced the *irrelevant vertex technique* according to which in every instance of treewidth greater than $g(k)$ there is an "irrelevant" vertex whose removal creates an equivalent instance of the problem. This fact is based on the celebrated *Unique Linkage Theorem*, whose – very technical – proof gives a function $g(k)$ that is responsible for an immense parameter dependence in the running time of the algorithm. In this paper we prove this result for planar graphs achieving $g(k) = 2^{O(k)}$. Our bound is radically better than the bounds known for general graphs. Moreover, our proof is new and self-contained, and it strongly exploits the combinatorial properties of planar graphs. We also prove that our result is optimal, in the sense that the function $g(k)$ cannot become better than exponential. Our results suggest that any algorithm for the Disjoint-Paths Problem that runs in time better than $2^{2^{o(k)}} \cdot n^{O(1)}$ will probably require drastically different ideas from those in the irrelevant vertex technique.

1 Introduction

One of the most studied problems in graph algorithms is the Disjoint-Paths Problem (DPP): *Given a graph G, and a set of k pairs of terminals, $(s_1, t_1), \ldots, (s_k, t_k)$, decide whether G contains k vertex-disjoint paths $P_1, \ldots, P_k$ where P_i has endpoints s_i and t_i, $i = 1, \ldots, k$.* In addition to its numerous applications in areas such as network routing and VLSI layout, this problem has been the catalyst for extensive research in algorithms and combinatorics [22]. DPP is NP-complete, along with its edge-disjoint or directed variants, even when the input graph is planar [23,15,12,14]. The celebrated algorithm of Roberson and Seymour solves it however in $f(k) \cdot n^3$ steps, where f is some computable function [17]. This implies that when we parameterize DPP by the number k of terminals, the problem is fixed-parameter tractable. The Robertson-Seymour algorithm is

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 110–121, 2011.

the central algorithmic result of the Graph Minors series of papers, one of the deepest and most influential bodies of work in graph theory.

The basis of the algorithm in [17] is the so called *irrelevant-vertex technique* which can be summarized very roughly as follows. As long as the input graph G violates certain structural conditions, then it is possible to find a vertex v that is *solution-irrelevant:* every collection of paths certifying a solution to the problem can be rerouted to an *equivalent* one, that links the same pairs of terminals, but in which the new paths avoid v. One then iteratively removes such irrelevant vertices until the structural conditions are met. By that point the graph has been simplified enough so that the problem can be attacked via dynamic programming.

The following two structural conditions are used by the algorithm in [17]: (i) G excludes a clique, whose size depends on k, as a minor and (ii) G has treewidth bounded by some function of k. When it comes to enforcing Condition (ii), the aim is to prove that in graphs without big clique-minors and with treewidth at least $g(k)$ there is always a solution-irrelevant vertex. This is the most complicated part of the proof and it was postponed until the later papers in the series [18,19]. The bad news is that the complicated proofs also imply an *immense* dependence, as expressed by the function f, of the running time on the parameter k. This puts the algorithm outside the realm of feasibility even for elementary values of k.

The ideas above were powerful enough to be applicable also to problems outside the context of the Graph Minors series. During the last decade, they have been applied to many other combinatorial problems and now they constitute a basic paradigm in parameterized algorithm design (see, e.g., [3,4,7,8,9,11]). However, in most applications, the need for overcoming the high parameter dependence emerging from the structural theorems of the Graph Minors series, especially those in [18,19], remains imperative. Hence two natural directions of research are: simplify parts of the original proof for the general case or focus on specific graph classes that may admit proofs with better parameter dependence. An important step in the first direction was taken recently by Kawarabayashi and Wollan in [10] who gave an easier and shorter proof of the results in [18,19]. While the parameter dependence of the new proof is certainly much better than the previous, immense, function, it is still huge: a rough estimation from [10] gives a lower bound for $g(k)$ of magnitude $2^{2^{2^{\Omega(k)}}}$ which in turn implies a lower bound for $f(k)$ of magnitude $2^{2^{2^{2^{\Omega(k)}}}}$.

In this paper we offer a solid advance in the second direction, focusing on planar graphs (see also [16,21] for previous results on planar graphs). We prove that, for planar graphs, $g(k)$ is singly exponential. In particular we prove the following result.

Theorem 1. *There is a constant c such that every n-vertex planar graph G with treewidth at least c^k contains a vertex v such that every solution to* DPP *with input G and k pairs of terminals can be replaced by an equivalent one avoiding v.*

Given the above result, our Theorem 6 shows how to reduce, in $O(n^2)$ time, an instance of DPP to an an equivalent one whose graph G' has treewidth $2^{O(k)}$. Then, using dynamic programming, a solution, if one exists, can be found in $k^{O(\text{treewidth}(G'))} \cdot n = 2^{2^{O(k)}} \cdot n$ steps.

The proof of Theorem 1 deviates significantly from those in [18,19,10]. It is self-contained and exploits extensively the combinatorics of planar graphs. Moreover, we give strong evidence that a parameterized algorithm for DPP with singly exponential dependence, if one exists, should require entirely different techniques. Indeed, in that sense, the result in Theorem 1 is tight:

Theorem 2. *There exists an instance of the* DPP *on a $2^{\Omega(k)}$-treewidth planar graph G that has a unique solution spanning all the vertices of G.*

Notice that, due to the recent lower bounds in [13], the Disjoint-Paths Problem cannot be solved in $2^{o(w \log w)} \cdot n^{O(1)}$ for graphs of treewidth at most w, unless the Exponential Time Hypothesis fails. This result, along with Theorem 2, reveals the limitations of the irrelevant vertex technique: any algorithm for the Disjoint-Paths Problem whose parameter dependence that is better than doubly exponential, will probably require drastically different techniques.

2 Preliminaries

Graphs are finite, undirected and simple. We denote the vertex set of a graph G by $V(G)$ and the edge set by $E(G)$. Every edge is a two-element subset of $V(G)$. A graph H is a *subgraph* of a graph G, denoted by $H \subseteq G$, if $V(H) \subseteq V(G)$ and $E(H) \subseteq E(G)$. A *path* in a graph G is a sequence $P = v_1, \ldots, v_n$ of pairwise distinct vertices of G, such that $v_i v_{i+1} \in E(G)$ for all $1 \leq i \leq n-1$. For a graph G with $e = vw \in E(G)$ let G/e denote the graph obtained from G by *contracting* e, i.e. $V[G/e] := (V(G) \setminus \{v, w\}) \cup \{x_e\}$, where x_e is a new vertex, and $E(G/e) := \big(E(G) \setminus \{uu' \mid uu' \cap e \neq \emptyset\}\big) \cup \{ux_e \mid uv \in E(G) \text{ or } uw \in E(G)\}$. A graph H is a *minor* of a graph G, if H can be obtained from a subgraph of G by a sequence of edge contractions. We use standard graph terminology as in [5]. The *disjoint paths problem* (DPP) is the following problem.

DPP

Input: A graph G, and pairs of terminals $(s_1, t_1), \ldots, (s_k, t_k) \in V(G)^{2k}$

Question: Are there k pairwise vertex disjoint paths $P_1, \ldots, P_k$ in G such that P_i has endpoints s_i and t_i?

We will call such a sequence $P_1, \ldots, P_k$ a *solution* of the DPP.

Given an instance $(G, (s_1, t_1), \ldots, (s_k, t_k))$ of DPP we say that a non-terminal vertex $v \in V(G)$ is *irrelevant*, if $(G, (s_1, t_1), \ldots, (s_k, t_k))$ is a YES-instance if and only if $(G \setminus v, (s_1, t_1), \ldots, (s_k, t_k))$ is a YES-instance. From now on G will always be an instance to DPP accompanied by k terminal pairs.

Definition 1 (Grid). *Let $m, n \geq 1$. The $(m \times n)$* grid *is the Cartesian product of a path of length $m-1$ and a path of length $n-1$. In case of a square grid where $m = n$, then we say that n is the* size *of the grid.*

A *subdivided grid* is a graph obtained from a grid by replacing some edges of the grid by pairwise internally vertex disjoint paths of length at least one. *Embeddings* of graphs in the plane, *plane graphs*, *planar graphs* and *faces* are defined in the usual way. A graph is *outerplanar*, if it has an embedding in the plane where all vertices are incident to the infinite face.

A *cycle* in a graph G is a subgraph $H \subseteq G$, such that $V(H) = \{x_0, x_1, \ldots, x_{k-1}\}$, $E(H) = \{x_0x_1, x_1x_2, \ldots, x_{k-2}x_{k-1}, x_{k-1}x_0\}$ for some $k \in \mathbb{N}$, $k \neq 1$, and $i \neq j \Rightarrow x_i \neq x_j$. Notice that we allow cycles to consist of a single vertex.

We use the fact that a subdivided grid has a unique embedding in the plane (up to homeomorphism). For (1×1) grids and subdivided (2×2) grids this is clear, and for $(n \times n)$ grids with $n \geq 2$ this follows from Tutte's Theorem stating that 3-connected graphs have unique embeddings in the plane (up to homeomorphism). This implies that subdivisions of $(n \times n)$ grids have unique embeddings as well. The *perimeter* of a subdivided grid H is the cycle in H that is incident to the outer face (in every planar embedding of H). We refer the reader to [2] for the definition of *tree-width* of a graph $tw(G)$.

A *directed graph* is a pair $D = (V, E)$ where V is a set and $E \subseteq V \times V$. We call the elements of E *directed edges*. For a directed edge $(u, v) \in E$ we say that u is the *tail* of (u, v), $u = \text{tail}(u, v)$, and v is the *head* of (u, v), $v = \text{head}(u, v)$.

3 Upper Bounds

The main result of this section is Theorem 6 stating that there is an $O(n^2)$-step algorithm that, given an instance G of DPP of treewidth $2^{\Omega(k)}$, can find a set of irrelevant vertices whose removal from G creates an equivalent instance of treewidth $2^{O(k)}$.

3.1 Basic Definitions

Observation 1. *Let G, $(s_1, t_1, \ldots, s_k, t_k)$ be a planar instance of* DPP *and let $h \in \mathbb{N}$. If G contains a subdivided $((h\sqrt{2k+1}) \times (h\sqrt{2k+1}))$ grid, then G contains a subdivided $h \times h$ grid H such that in every embedding of H all terminals lie outside the open disc bounded by the perimeter of H.*

Notice that Observation 1 ensures that once we have a large grid we can also assume that we have a large grid *that does not contain any terminal vertices*. Next we define a specific kind of embedding of cycles that helps us enforce structure in the proof.

Definition 2 (Tight concentric cycles). *Let G be a plane graph and let $C_0, \ldots, C_n$ be a sequence of cycles in G such that each cycle bounds a closed disc D_i in the plane. We call $C_0, \ldots, C_n$* concentric*, if for all $i \in \{0, \ldots, n-1\}$, the*

cycle C_i is contained in the interior of D_{i+1}. The concentric cycles $C_0, \ldots, C_n$ are tight*, if, in addition, C_0 is a single vertex and for every $i \in \{0, \ldots, n-1\}$, $D_{i+1} \setminus D_i$ does not contain a cycle C bounding a disc D in the plane with $D_{i+1} \supsetneq D \supseteq D_i$.*

Remark 1. Let G be a plane graph and let $C_0, \ldots, C_n$ be tight concentric cycles in G bounding closed discs $D_1, \ldots, D_n$, respectively, in the plane. Let P be a path connecting vertices u and v with $u, v \notin D_n$. If a vertex of P is contained in the interior of D_i (i.e. in $D_i \setminus C_i$), then P has a vertex on C_{i-1}.

Remark 2. If a graph contains a $((2n+1) \times (2n+1))$ grid minor, it contains a sequence $C_0, \ldots, C_n$ of tight concentric cycles.

A *linkage* in a graph G is a family of pairwise disjoint paths in G. The *endpoints* of a linkage L are the endpoints of the paths in L, and the *pattern* of L is the matching on the endpoints induced by the paths, i.e. the pattern is the set $\big\{\{s,t\} \mid L \text{ contains a path from } s \text{ to } t\big\}$.

Definition 3 (Segment & Handle). *Let G be a plane graph, let C be a cycle in G bounding a closed disc D in the plane and let P be a path in G such that its endpoints are outside of D. We say that a path P_0 is a* D-segment *(resp.* D-handle*) of P, if P_0 is a non-empty maximal subpath of P whose endpoints are on C, and $P_0 \subseteq D$ (resp. $P_0 \cap D$ contains only the endpoints of P_0). For a linkage $\mathcal{P}$ in G we say that a path P_0 is a* D-segment *(D-handle*) of $\mathcal{P}$, if P_0 is a* D-segment *(D-handle*) of some path P of $\mathcal{P}$.*

Remark 3. Let G be a plane graph, let C be a cycle in G bounding a closed disc D in the plane and let $P = su_1 \ldots u_q t$ be a path in G such that s and t are outside D. Suppose $x_a, x_b, x_c, \ldots, x_j$ is the order in which the vertices of path P appear on the cycle C when we traverse it from s to t. Then the subpath of P between x_a and x_b is a D-segment while the subpath of P between x_b and x_c is a D-handle, and D-segments and D-handles alternate.

From now on we will assume that G is a plane graph containing a sequence $C_0, \ldots, C_n$ of concentric cycles bounding closed discs $D_0, \ldots, D_n$, respectively, in the plane. Furthermore there are no terminals contained in D and G is an YES-instance. That is, there are paths between s_i and t_i such that they are mutually disjoint. These paths form a linkage that will be denoted by $\mathcal{P}$. From now onwards whenever we say *linkage* we mean a set of disjoint paths between pairs (s_i, t_i). We will often refer to the D_n-segments (D_n-handles) of $\mathcal{P}$ simply as the *segments (handles)* of $\mathcal{P}$.

Definition 4 (I(Handle) and β(Handle)). *Let G be a plane graph containing a sequence $C_0, \ldots, C_n$ of concentric cycles bounding closed discs $D_0, \ldots, D_n$, respectively, in the plane and $\mathcal{P}$ be a linkage. Let P be a D_n-handle and let its endpoints be x and y. Let $C_n[x,y]$ denote the path between x, y on the cycle C_n such that the finite face bounded by $P \cup C_n[x,y]$ does not contain the interior of D_n. By $I(P)$ we denote the subgraph of G that has boundary $P \cup C_n[x,y]$, and we let $\beta(P) := C_n[x,y]$.*

Definition 5 (Cheap solution). *Let G be a plane graph containing a sequence $C_0, \ldots, C_n$ of concentric cycles bounding closed discs $D_0, \ldots, D_n$, respectively, in the plane. For a linkage $\mathcal{P}$ of G, define its* cost *$c(\mathcal{P})$ as the number of edges of $\mathcal{P}$ that do not belong to $\bigcup_{i=0}^{n} C_i$. A linkage $\mathcal{P}$ is called* cheap*, if there is no other linkage $\mathcal{Q}$, such that $c(\mathcal{Q}) < c(\mathcal{P})$.*

Observe that the contribution of a D_n-handle of $\mathcal{P}$ to $c(\mathcal{P})$ is always positive. Edges of D_n-segments contribute to $c(\mathcal{P})$ whenever they do not belong to a concentric cycle. We assume for the remainder of Section 3 that we are given a cheap solution $\mathcal{P}$ to our input instance and we explore its structure.

3.2 Simple Properties of a Cheap Solution $\mathcal{P}$

Lemma 1. *If $\mathcal{P}$ is a cheap solution to the input instance then there is no segment P of $\mathcal{P}$ with vertices appearing in the order $\ldots, v_0, \ldots, v_1, \ldots, v_2, \ldots$ where v_0 and v_2 are vertices of C_ℓ, and v_1 is a vertex of C_j, for $n \geq j > \ell \geq 0$.*

Lemma 2. *Let $\mathcal{P}$ be a cheap solution to the input instance and Q be a handle. Then there is terminal inside $I(Q)$.*

We remark that Lemma 2 is true in more general setting. We will use the generalized version in a proof later. Let D be a disc with the boundary cycle C and $\mathcal{T}$ be a subpath of a path in $\mathcal{P}$ with endpoints x and y on the disc and no vertices in the interior of the disc. Let $C[x, y]$ denote the path between x, y on cycle C such that the finite face bounded by $\mathcal{T} \cup C[x, y]$ does not contain the interior of D. By $I(\mathcal{T})$ we denote the subgraph that has boundary $\mathcal{T} \cup C[x, y]$. A proof similar to the one in Lemma 2 gives us the following.

Lemma 3. *Let $\mathcal{P}$ be a cheap solution to the input instance and $\mathcal{T}$ be a subpath of a path in $\mathcal{P}$ with endpoints on the disc and no points in the interior of the disc. Then there is a terminal inside $I(\mathcal{T})$.*

3.3 Bounding the Number of Segment Types

In this section we define a notion of segment types and obtain an upper bound on the number of segment types.

Definition 6 (Segment Type). *Let $\mathcal{P}$ be a solution to the input instance. Let R and S be two D_n-segments. Let Q and Q' be the two paths on C_n connecting an endpoint of R with an endpoint of S and passing through no other endpoint of R or S. We say that R and S are* equivalent*, and we write $R \parallel S$, if no D_n-segment of $\mathcal{P}$ has both endpoints on Q and no D_n-segment has both endpoints on Q'. A* type *of D_n-segments is an equivalence class of D_n-segments under the relation $\parallel$.*

Definition 7 (Segment graph). *We start with the subgraph of G contained in D_n. Retain only the edges and vertices of $\bigcup \mathcal{P} \cup C_n$. Choose an edge. If it is part of C_n, contract it unless it connects endpoints of segments of different type. If*

it is not part of C_n, contract it unless it connects endpoints of segments. Repeat until there are no contractable edges left. Remove duplicate edges and loops, such that the graph becomes simple again. The resulting graph is the segment graph *of D_n (clearly, segment graphs are outerplanar graphs).*

Definition 8 (Tongue tip). *A D_n-segment type is called* tongue tip*, if it is a single vertex in the segment graph of D_n.*

Definition 9 (Segment dual graph). *We take the dual graph of the segment graph of D_n. Delete the vertex that represents the infinite face. Add the vertices representing the tongue tips of the segment graph and connect them to the vertices representing neighboring faces in the segment graph. The resulting graph is the* segment dual graph *of D_n.*

Remark 4. Since the segment graph is outerplanar, the segment dual graph is a tree. All inner nodes of the segment dual graph have degree at least 3.

The next lemma is based on Lemmata 2 and 3.

Lemma 4 (Tongue-taming). *Let $\mathcal{P}$ be a cheap solution to the input instance. Then there are at most $2k-1$ tongue tips.*

Theorem 3. *Let $\mathcal{P}$ be a cheap solution to the input instance then $\mathcal{P}$ has at most $4k-3$ different types of D_n-segments.*

3.4 Bounding the Size of Segment Types

In this section we find a bound on the size of segment types in cheap solutions and we combine it with the bound on the number of segment types obtained in the previous section to find irrelevant vertices. Indeed, we find that cheap solutions only pass through a bounded number of concentric cycles.

We find the bound on the size of segment types by rerouting in the presence of a large segment type. In a first step, we allow ourselves to freely reroute in a disc (making sure that the graph remains planar), and we bound the number of segments of solution paths in the disc. In a second step, we realize our rerouting in a sufficiently large grid.

Lemma 5. *Let Σ be an alphabet of size $|\Sigma| = k$. Let $w \in \Sigma^*$ be a word over Σ. If $|w| > 2^k$, then w contains an infix y with $|y| \geq 2$, such that every letter occurring in y occurs an even number of times in y.*

The following lemma is essentially the main combinatorial result from [1]. The proof is included here for the sake of completeness.

Lemma 6 (Rerouting in a disc). *Let G be a plane graph with k pairs of terminals such that the DPP has a solution $\mathcal{P}$. Let G contain a cycle C bounding a closed disc D in the plane, such that no terminal lies in D. Assume that every D-segment of $\mathcal{P}$ is simply an edge and, except for vertices and edges of D-segments, the interior of D contains no other vertices or edges of G. Then, if*

there is a segment type that contains more than 2^k segments, then we can replace the outerplanar graph O consisting of all D-segments of $\mathcal{P}$ by a new outerplanar graph O' such that in $(G \setminus O) \cup O'$ the DPP (with the original terminals) has a solution and $|E(O')| < |E(O)|$.

Definition 10. *Let $n, m \in \mathbb{N}$. An* untidy $(n \times m)$ grid *is a graph obtained from a set $\mathcal{H}$ of n pairwise vertex-disjoint (*horizontal*) paths and a set $\mathcal{V} = \{V_1, \ldots, V_m\}$ of m pairwise vertex-disjoint (*vertical*) paths as follows: Every path in $\mathcal{V}$ intersects every path in $\mathcal{H}$ in precicely one non-empty path, and each path $H \in \mathcal{H}$ consists of m vertex-disjoint segments such that V_i intersects H only in its ith segment (for every $i \in \{1, \ldots, m\}$). A* subdivided untidy $(n \times m)$ grid *is obtained from an untidy $(n \times m)$ grid by subdividing edges.*

Let τ be a segment type in the plane graph. Recall that all the segments in a type are "parallel" to each other. We say that segments $S_1, \ldots, S_n \in \tau$ are *consecutive*, if they appear in this order (or in the reverse order) in the plane. Segment types that go far into the concentric cycles yield subdivided untidy grids. More precisely, we show the following lemma, that is an easy consequence of Lemma 1.

Lemma 7. *Let $l, n, r \in \mathbb{N}$ with $n \geq l - 1$. Let $\mathcal{P}$ be a cheap solution to the input instance. If there is a type τ of D_n-segments of $\mathcal{P}$ with $|\tau| \geq r$ such that r consecutive segments of τ each contain a vertex of D_{n-l+1}, then G contains a subdivided untidy $(2l \times r)$ grid as a subgraph, with the r consecutive segments of τ as vertical paths, and suitable subpaths of $C_n, \ldots, C_{n-l+1}, C_{n-l+1}, \ldots, C_n$ (in this order) as horizontal paths.* □

The following lemma, whose proof is based on Lemmata 6 and 7, shows that we can reroute a sufficiently large segment type in the case that many segments of the type go far into the concentric cycles.

Lemma 8 (Rerouting in an untidy grid). *Let $n, k \in \mathbb{N}$ with $n \geq 2^{k-1} - 1$. Let $\mathcal{P}$ be a cheap solution to the input instance. Then $\mathcal{P}$ has no type τ of D_n-segments with $|\tau| \geq 2^k + 1$, such that each of $2^k + 1$ consecutive segments in τ contains a vertex in $D_{n-2^{k-1}+1}$.*

The following remark says that if we have a segment type of sufficiently large cardinality, then many segments will go far into the concentric cycles.

Lemma 9. *Let $n, l, r \in \mathbb{N}$. Let $\mathcal{P}$ be a cheap solution to the input instance. Let τ be a type of D_n-segments with $|\tau| \geq 2l + r$. Then $n \geq l - 1$ and τ contains r consecutive segments such that each of them has a vertex in D_{n-l+1}.*

The next theorem is based on Lemmata 8 and 9 and Theorem 3.

Theorem 4. *Let $\mathcal{P}$ be a cheap solution to the input instance. Then there are at most $(8k - 6) \cdot 2^k + 4k - 3$ D_n-segments of $\mathcal{P}$.*

Theorem 4 along with Lemma 1 yield the following.

Theorem 5 (Irrelevant Vertex). *Let G be a plane graph with k pairs of terminals, $n = (8k-6)\cdot 2^k + 4k - 2$, and let G contain a sequence $C_0, \ldots, C_n$ of concentric cycles bounding closed discs $D_0, \ldots, D_n$, respectively, in the plane, such that no terminal of the* DPP *lies in D_n. Let $C_0 = \{v\}$, and assume that the* DPP *has a solution. Then the* DPP *has a solution that avoids v.*

Given a plane graph G and a vertex v we show how to check whether a particular vertex v satisfies the conditions of Theorem 5. We set $C_0 = \{v\}$ and given C_i we construct C_{i+1} by performing a depth first search from a neighbor u of a vertex in C_i, always chosing the rightmost edge leaving the vertex we are visiting. This search will either output an innermost cyclic walk (which then can be pruned to a cycle) around C_i or determine that no such walk exists. In the case that a cycle C_{i+1} is output, we check whether the cyclic walk contained a terminal s_i or t_i. If it did, it means that this terminal lies on C_{i+1} or in its interior. At this point (or when the search outputs that no cycle around C_i exists), we have determined that there are i tight concentric cycles around v with no terminal in the interior of C_i. If $i > (8k-6)\cdot 2^k + 4k - 2$ this implies that v satisfies the conditions of Theorem 5. Clearly this procedure can be implemented to run in linear time. This yields the following theorem.

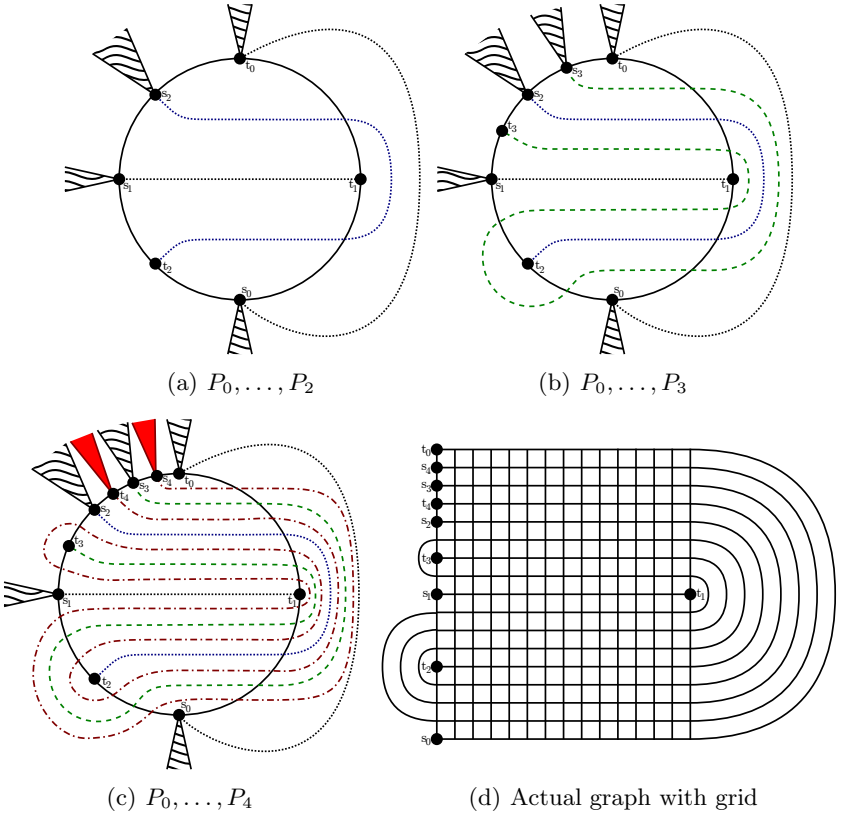
Theorem 6. *Let G be a plane graph with k pairs of terminals, there is an $O(|V(G)|^2)$ time algorithm that outputs an induced subgraph G' of G such that $tw(G') \leq 72\sqrt{2}k^{\frac{3}{2}} \cdot 2^k$ and G is a YES-instance for* DPP *if and only if G' is.*

4 The Lower Bound

Let $H \subseteq G$ be a subgraph of the plane graph G. An *inner vertex* of H is a vertex that is not part of the boundary of H.

Definition 11 (Crossing). *Let $H \subseteq G$ be a subgraph of the plane graph G. We say that a path* crosses *the subgraph H if it contains an inner vertex of H and its endpoints are not inner vertices of H. For $k \in \mathbb{N}$ we say that a path $P = p_0, p_1, \ldots, p_n$* crosses *$H$ k times, if it can be split into k paths $P_0 = p_0, p_1, \ldots, p_{i_1}, P_1 = p_{i_1}, p_{i_1+1}, \ldots p_{i_2}, \ldots, P_{k-1} = p_{i_{k-1}}, p_{i_{k-1}+1}, \ldots, p_n$ with each $P_i, i = 0, \ldots k-1$ crossing H. The parts of the P_i that do not lie outside of H are called crossings of H.*

Intuitively, we construct our example from a grid H of sufficient size. We add endpoints s_0 and t_0 on the boundary of the grid, mark the areas opposite to the grid as not part of the graph and connect s_0 to t_0 without crossing the grid. Now we continue to mark vertices by s_i and t_i in such a way that P_i has to cross H as often as possible (in order to avoid crossing $P_j, j < i$). Once s_i and t_i have been added we remove the area opposite to the grid from s_i from the graph. Figure 1(a) shows the situation after doing this for i up to 2. In this construction P_0 does not cross the grid at all, while P_1 crosses it once and P_{i+1} crosses it twice as often as P_i for $i > 0$: Let k_i be the number of times P_i crosses



(a) $P_0, \ldots, P_2$ (b) $P_0, \ldots, P_3$

(c) $P_0, \ldots, P_4$ (d) Actual graph with grid

Fig. 1. Construction of graph and solution

the grid. $k_0 = 0, k_1 = 1, k_{i+1} = 2k_i, k_i = 2^{i-1}, i > 0$. After the last P_i has been added, the areas opposite to the grid from both s_i and t_i are removed from the graph as seen in Figure 1(c).

Formally, to construct problem and graph with $k+1$ terminals, we use a $((2^k+1) \times (2^k+1))$ grid. Let the vertices on the left boundary of the grid be $n_0, \ldots, n_{2^k}$. Terminals are assigned as follows: t_0 is the topmost vertex on the left boundary on the grid, t_1 the middle vertices on the right boundary. For all other terminals: $s_i := n_{2^{k-i}}, t_i := n_{3 \cdot 2^{k-i}}$. Then add edges going around the t_i to the graph: For $i > 1, t_i = n_j$ add $n_{j-1}n_{j+1}, n_{j-2}n_{j+1}, \ldots, n_{j-2^{k-i}-1}n_{j+2^{k-i}-1}$, and on the right boundary of H do the analogue for t_1. See Figure 1(d) for a graph constructed this way.

Theorem 7. *There is only one solution to the constructed DPP, all vertices of the graph lie on paths of the solution and the grid is crossed* $2^k - 1$ *times by such paths.*

In particular, H has no *irrelevant* vertex in the sense of [19].

Corollary 1. *There is a planar graph G with $k+1$ pairs of terminals such that*

- *G contains a $((2^k+1)\times(2^k+1))$ grid as a subgraph,*
- *the disjoint paths problem on this input has a unique solution,*
- *the solution uses all vertices of G; in particular, no vertex of G is irrelevant.*

Vital linkages and tree-width. We refer the reader to [2] for the definitions of *tree-width* and *path-width*. A linkage L in a graph G is a *vital linkage* in G, if $V(\bigcup L) = V(G)$ and there is no other linkage $L' \neq L$ in G with the same pattern as L.

Theorem 8 (Robertson and Seymour [20]). *There are functions f and g such that if G has a vital linkage with k components then G has tree-width at most $f(k)$ and path-width at most $g(k)$.*

Recall that the $(n \times n)$ grid has path-width n and tree-width n. Our example yields a lower bound for f and g:

Corollary 2. *Let f and g be as in Theorem 8. Then $2^{k-1}+1 \leq f(k)$ and $2^{k-1}+1 \leq g(k)$.*

Proof. Looking at the graph G and DPP constructed above the solution to the DPP is, due to its uniqueness, a vital linkage for the graph G. G contains a $((2^k+1)\times(2^k+1))$ grid as a minor. The tree-width of such a grid is 2^k+1 , its path-width 2^k+1 [6]. Thus we get lower bounds $2^{k-1}+1 \leq f(k), g(k)$ for the functions f and g. □

Acknowledgment. We thank Ken-ichi Kawarabayashi and Paul Wollan for providing details on the bounds in [10].

References

1. Adler, I., Kolliopoulos, S.G., Thilikos, D.: Planar disjoint paths completion. Submitted for publication (2011)
2. Bodlaender, H.L.: A tourist guide through treewidth. Acta Cybernet. 11(1-2), 1–21 (1993)
3. Dawar, A., Grohe, M., Kreutzer, S.: Locally excluding a minor. In: LICS 2007, pp. 270–279. IEEE Computer Society, Los Alamitos (2007)
4. Dawar, A., Kreutzer, S.: Domination problems in nowhere-dense classes. In: IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2009), pp. 157–168 (2009)
5. Diestel, R.: Graph Theory. Springer, Heidelberg (2005)
6. Ellis, J., Warren, R.: Lower Bounds on the Pathwidth of some Grid-like Graphs. Discrete Applied Mathematics 156(5), 545–555 (2008)
7. Golovach, P.A., Kaminski, M., Paulusma, D., Thilikos, D.M.: Induced packing of odd cycles in a planar graph. In: Dong, Y., Du, D.-Z., Ibarra, O. (eds.) ISAAC 2009. LNCS, vol. 5878, pp. 514–523. Springer, Heidelberg (2009)

8. Kawarabayashi, K.-i., Kobayashi, Y.: The induced disjoint paths problem. In: Lodi, A., Panconesi, A., Rinaldi, G. (eds.) IPCO 2008. LNCS, vol. 5035, pp. 47–61. Springer, Heidelberg (2008)
9. Kawarabayashi, K.-i., Reed, B.: Odd cycle packing. In: Proceedings of the 42nd ACM Symposium on Theory of Computing (STOC 2010), pp. 695–704. ACM, New York (2010)
10. Kawarabayashi, K.-i., Wollan, P.: A shorter proof of the graph minor algorithm: the unique linkage theorem. In: Proc. of the 42nd annual ACM Symposium on Theory of Computing (STOC 2010), pp. 687–694 (2010)
11. Kobayashi, Y., Kawarabayashi, K.-i.: Algorithms for finding an induced cycle in planar graphs and bounded genus graphs. In: Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2009), pp. 1146–1155. ACM-SIAM (2009)
12. Kramer, M.R., van Leeuven, J.: The complexity of wire-routing and finding minimum area layouts for arbitrary VLSI circuits. Advances in Comp. Research 2, 129–146 (1984)
13. Lokshtanov, D., Marx, D., Saurabh, S.: Slightly superexponential parameterized problems. In: 22st ACM–SIAM Symposium on Discrete Algorithms (SODA 2011), pp. 760–776 (2011)
14. Lynch, J.F.: The equivalence of theorem proving and the interconnection problem. ACM SIGDA Newsletter 5, 31–36 (1975)
15. Middendorf, M., Pfeiffer, F.: On the complexity of the disjoint paths problem. Combinatorica 13(1), 97–107 (1993)
16. Reed, B., Robertson, N., Schrijver, A., Seymour, P.D.: Finding dsjoint trees in planar graphs in linear time. In: Robertson, N., Seymour, P.D. (eds.) Graph Structure Theory. Contemporary Mathematics, vol. 147, pp. 295–302. American Mathematical Society, Providence (1991)
17. Robertson, N., Seymour, P.D.: Graph minors. XIII. The disjoint paths problem. J. Combin. Theory Ser. B 63(1), 65–110 (1995)
18. Robertson, N., Seymour, P.: Graph minors. XXI. Graphs with unique linkages. J. Combin. Theory Ser. B 99(3), 583–616 (2009)
19. Robertson, N., Seymour, P.D.: Graph Minors. XXII. Irrelevant vertices in linkage problems. Journal of Combinatorial Theory, Series B (to appear)
20. Robertson, N., Seymour, P.D.: Graph minors. XXI. Graphs with unique linkages. Journal of Combinatorial Theory, Series B 99(3), 583–616 (2009)
21. Schrijver, A.: Finding k disjoint paths in a directed planar graph. SIAM J. Comput. 23(4), 780–788 (1994)
22. Schrijver, A.: Combinatorial optimization. Polyhedra and efficiency, vol. A. Springer, Berlin (2003)
23. Vygen, J.: NP-completeness of some edge-disjoint paths problems. Discrete Appl. Math. 61(1), 83–90 (1995)

A Tighter Insertion-Based Approximation of the Crossing Number

Markus Chimani[1,⋆] and Petr Hliněný[2,⋆⋆]

[1] Algorithm Engineering, Friedrich-Schiller-University Jena, Germany
markus.chimani@uni-jena.de
[2] Faculty of Informatics, Masaryk University Brno, Czech Republic
hlineny@fi.muni.cz

Abstract. Let G be a planar graph and F a set of additional edges not yet in G. The *multiple edge insertion* problem (MEI) asks for a drawing of $G+F$ with the minimum number of pairwise edge crossings, such that the subdrawing of G is plane. As an exact solution to MEI is NP-hard for general F, we present the first approximation algorithm for MEI which achieves an additive approximation factor (depending only on the size of F and the maximum degree of G) in the case of connected G. Our algorithm seems to be the first directly implementable one in that realm, too, next to the single edge insertion.

It is also known that an (even approximate) solution to the MEI problem would approximate the crossing number of the *F-almost-planar graph* $G + F$, while computing the crossing number of $G + F$ exactly is NP-hard already when $|F| = 1$. Hence our algorithm induces new, improved approximation bounds for the crossing number problem of F-almost-planar graphs, achieving constant-factor approximation for the large class of such graphs of bounded degrees and bounded size of F.

1 Introduction

The crossing number $\mathrm{cr}(G)$ of a graph G is the minimum number of pairwise edge crossings in a drawing of G in the plane. The crossing number problem has been vividly investigated for over 60 years, and yet only little is known about it. See [23] for an extensive bibliography. While the problem's approximability is still unknown, several approximation algorithms arose for special graph classes.

The best known polynomial algorithm for the crossing number of general graphs with bounded degree [1, 12] approximates, within a factor of $\log^2 |V(G)|$, the quantity $|V(G)| + \mathrm{cr}(G)$, not directly $\mathrm{cr}(G)$. The known constant factor approximations restrict themselves to graphs following one of two paradigms (see also Section 4): they either assume that the graph is embeddable in some higher surface [13, 18, 20], or they are based on the idea that only a small set of graph elements has to be removed from G to make it planar: removing and

⋆ Markus Chimani was funded by a Carl-Zeiss-Foundation juniorprofessorship.
⋆⋆ Petr Hliněný has been supported by Eurocores grant GIG/11/E023 and Czech Science Foundation grant P202/11/0196.

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 122–134, 2011.

re-inserting them can give strong approximation bounds [3, 8, 19]. In this paper, we follow the latter idea and first concentrate on the following problem:

Definition 1.1. *Given a planar graph G and a set F of k edges (vertex pairs, in fact) not yet in G. The* multiple edge insertion *problem* MEI(G, F) *is to find the minimum number* $\mathrm{ins}(G, F)$ *of crossings necessary to draw $G + F$ (G including the edges F) such that the subdrawing restricted to G is planar. In other words, we ask for some* planar embedding *of G such that we can* insert *the edges F into this embedding making the least possible number of edge crossings.*

For general k, the MEI problem is known to be NP-hard [24], based on a reduction from *fixed linear crossing number*; for fixed $k > 1$ the problem complexity is open.

The case $k = 1$ of MEI is known as the *(single) edge insertion* problem and can be solved in linear time [17], as we will briefly summarize in Section 2.2. Let e be the edge to insert, then denote the resulting number of crossings by $\mathrm{ins}(G, e)$. Let $\Delta(G)$ denote the maximum degree in G. It was shown [3, 19] that $\mathrm{ins}(G, e)$ approximates the crossing number $\mathrm{cr}(G + e)$ —i.e., of the graph containing this edge e—within a multiplicative factor of $\lfloor \frac{1}{2}\Delta(G) \rfloor$ achieved in [3], and this bound is tight. Notice also that already computing $\mathrm{cr}(G + e)$ exactly is NP-hard [4].

Another special case of the MEI problem is when one adds a new node together with its incident edges; this is also polynomially solvable [6] and approximates the crossing number of the resulting *apex graph* [8]. These are the only two types of insertion problems which are currently known to be in P.

Nevertheless, it has been proved in [8] (see Section 4) that a solution (even an approximate one) to MEI(G, F) would directly imply an approximation algorithm for $\mathrm{cr}(G+F)$ with planar G. Just recently, Chuzhoy et al. [10] have shown the first algorithm efficiently computing an approximate solution to the crossing number problem on $G + F$ with a help of a multiple edge insertion solution. Precisely, Chuzhoy et al. [10] achieve a solution with the number of crossings

$$\mathrm{cr}^{\mathrm{aprx}}(G + F) \;\le\; O\big(\Delta(G)^3 \cdot |F| \cdot \mathrm{cr}(G + F) + \Delta(G)^3 \cdot |F|^2\big) \tag{1}$$

(without giving explicit constants). Though not mentioned explicitly in [10], it seems that their results also give an approximation solution to MEI(G, F) with the same ratio, at least in the case of 3-connected $G+F$. A further approximation result, though not directly related to our topic, was given by Chuzhoy in [9].

In this paper, we present an alternative approach to the one proposed in [10]: We directly give an efficient algorithm approximating a solution of MEI(G, F), and then employ the aforementioned result of [8]. On the one hand, our approach is algorithmically and implementationally simpler, virtually only building on top of well-studied and experimentally evaluated sub-algorithms. On the other hand, it gives stronger approximations, cf. (3), as well as better runtime bounds. Our algorithm, in fact, seems to be the first *directly implementable* algorithm in this area, next to the single edge insertion. We are going to show:

Theorem 1.2. *Given a connected planar graph G and an edge set F, $F \cap E(G) = \emptyset$, Algorithm 3.1 described below finds, in $O(|F| \cdot |V(G)| + |F|^2)$ time, a solution to the* MEI(G, F) *problem with* $\mathrm{ins}^{\mathrm{aprx}}(G, F)$ *crossings such that*

$$\mathrm{ins}^{\mathrm{aprx}}(G, F) \;\le\; \mathrm{ins}(G, F) + \left(\lfloor \tfrac{1}{2}\Delta(G)\rfloor + \tfrac{1}{2}\right) \cdot \left(|F|^2 - |F|\right). \tag{2}$$

Consequently, this gives an approximate solution to the crossing number problem

$$\mathrm{cr}^{\mathrm{aprx}}(G + F) \le \lfloor \tfrac{1}{2}\Delta(G)\rfloor \cdot 2|F| \cdot \mathrm{cr}(G + F) + \left(\lfloor \tfrac{1}{2}\Delta(G)\rfloor + \tfrac{1}{2}\right)\left(|F|^2 - |F|\right). \tag{3}$$

Notice the constant-factor approximation ratio when the degree of G and the size of F are bounded. Further consequences of our main result will be discussed below. We, moreover, remark that the assumption of connectivity of G is necessary in the context of Algorithm 3.1 (if G were not connected, then the approximation guarantee (2) of Algorithm 3.1 for $\mathrm{ins}^{\mathrm{aprx}}(G, F)$ would be just the same as for $\mathrm{cr}^{\mathrm{aprx}}(G + F)$ in line (3)).

2 Preliminaries

2.1 Decomposition Trees

We consider multigraphs by default. Our algorithm will use suitable tree-structured decompositions of the given planar graph, according to its connectivity.

Definition 2.1 (BC-tree). *Let G be a connected graph. The* BC-tree $\mathcal{B} = \mathcal{B}(G)$ *of G is a tree that satisfies the following properties:*

i. *$\mathcal{B}$ has two different node types: B- and C-nodes.*
ii. *For every cut vertex in G, $\mathcal{B}$ contains a unique corresponding C-node.*
iii. *For every maximal two-connected subgraph (*block*) in G, $\mathcal{B}$ contains a unique corresponding B-node.*
iv. *No two B-, and no two C-nodes are adjacent. A B-node is adjacent to a C-node iff the corresponding block contains the corresponding cut vertex.*

To further decompose the blocks, we consider SPQR-trees for each non-trivial B-node (i.e., the block containing more than a single edge). This decomposition was first defined in [11], based on prior work of [2, 22]. Even though more complicated than the BC-tree, it requires also only linear size and can be constructed in linear time [15, 21]. We are mainly interested in the property that an SPQR-tree can be used to efficiently represent and enumerate all (potentially exponentially many) planar embeddings of its underlying graph. For conciseness, we call our tree *SPR-tree*, as we do not require nodes of type Q.

Definition 2.2 (SPR-tree, cf. [5]). *Let H be a biconnected graph with at least three vertices. The* SPR-tree $\mathcal{T} = \mathcal{T}(H)$ *of H is the (unique) smallest tree satisfying the following properties:*

i. *Each node ν in $\mathcal{T}$ holds a specific (small) graph $S_\nu = (V_\nu, E_\nu)$, $V_\nu \subseteq V(H)$, called a* skeleton. *Each edge f of E_ν is either a* real *edge $f \in E(H)$, or a* virtual *edge $f = \{u, v\}$ where $\{u, v\}$ forms a 2-cut (a* split pair*) in G.*

ii. *$\mathcal{T}$ has only three different node types with the following skeleton structures:*

S: *The skeleton S_ν is a simple cycle—it represents a* serial *component.*

P: *The skeleton S_ν consists of two vertices and multiple edges between them—it represents a* parallel *component.*

R: *The skeleton S_ν is a simple triconnected graph.*

iii. *For every edge $\{\nu, \mu\}$ in $\mathcal{T}$ the following holds: S_ν contains a specific virtual edge e_μ which "represents" S_μ. Vice versa, $e_\nu \in E(S_\mu)$ represents S_ν. Both edges e_μ and e_ν connect the same vertices.*

iv. *The original graph H can be obtained by recursively applying the following operation: For the edge $\{\nu, \mu\} \in E(\mathcal{T})$, let e_μ, e_ν be the virtual edges as in (iii.) connecting the same vertices u, v. A merged graph $(S_\nu \cup S_\mu) - e_\mu - e_\nu$ is obtained by gluing the skeletons together at u, v and removing e_μ, e_ν.*

We remark that SPQR-trees have also been used in the aforementioned [10], though with a different approach. For our purpose, we are particularly interested in the amalgamated version of both above trees, chiefly denoted by *con-tree*:

Definition 2.3 (Con-tree). *Given a connected, planar graph G, let the* con-tree *$\mathcal{C} = \mathcal{C}(G)$ be formed of the BC-tree $\mathcal{B}(G)$ which holds SPR-trees $\mathcal{T}(H)$ for all non-trivial blocks H of G.*

Clearly, the linear-sized con-tree $\mathcal{C}$ can be obtained from G in linear time.

Observe that, for each cut vertex x (of G) incident with a block $H \subseteq G$, the nodes with skeletons containing x induce a subtree $T_{H,x} \subseteq \mathcal{T}(H)$. In further considerations it will be useful to imagine that the C-node of x in $\mathcal{B}(G)$ is adjacent to these corresponding nodes $V(T_{H,x})$ (over all blocks H incident with x) within $\mathcal{C}$. We will loosely call this view of $\mathcal{C}$ the *extended con-tree*, while understanding that it is not really a tree.

2.2 Single Edge Insertion with Variable Embedding

As noted before, Gutwenger et al. [17] presented an exact linear-time algorithm to solve the single edge insertion problem. Herein, we will only outline some central concepts of this approach. Consider a planar graph G and let v_1, v_2 be the vertices we want to connect by a new edge.

First consider G embedded such that we can deal with its dual graph G^*. We define an *insertion path* to be a path in G^* connecting a face incident to v_1 with a face incident to v_2. The length of this path is then the number of edge crossings necessary to insert the edge $\{v_1, v_2\}$ into embedded G along this path. For a fixed embedding, we can compute the shortest insertion path via a breath-first search (BFS). Now consider G without a fixed embedding. If v_1, v_2 are in two separate connected components, then we can insert the edge $\{v_1, v_2\}$ without any crossings, by choosing sub-embeddings of these components where v_1, v_2 lie on their respective outer faces.

Hence assume G to be connected and compute (in linear time) its con-tree $\mathcal{C}(G)$. Let L be the unique shortest path in $\mathcal{B}(G)$ from a B-node containing v_1 to a B-node containing v_2. The optimal insertion path for $\{v_1, v_2\}$ in G can be obtained by concatenating the optimal insertion paths within the (non-trivial) blocks on this path L; as we can always "flip" (cf. Definition 3.4, C-nodes) the two incident blocks at the common cut vertex to avoid additional crossings. For a block H represented by a B-node on L, let v_i^H, $i = 1, 2$, denote v_i if $v_i \in V(H)$, or the cut vertex in H closest to v_i otherwise. An *insertion path within* H then connects any face incident to v_1^H with any face incident to v_2^H.

To find the optimal insertion path within such a block $H \subseteq G$, let Q_H be the unique shortest path in $\mathcal{T}(H)$ from a skeleton containing v_1^H to a skeleton containing v_2^H. It was shown [17] that only the embeddings of the skeletons within Q_H matter. Roughly speaking, the algorithm walks along these skeletons and fixes a suitable embedding one after another. Finally, an optimal embedding is found and fixed, and one can use a simple BFS algorithm on the dual graph to insert the edge $\{v_1^H, v_2^H\}$ optimally.

Definition 2.4 (Con-chain). *Considering the previous notation, we call a* con-chain *of the edge* $\{v_1, v_2\}$ *the (unique) path* $Q = Q_G(\{v_1, v_2\})$ *resulting from* L *by expanding each B-node* $b \in V(L)$ *into the path* Q_H *where* H *is the block of* G *represented by* b.

Proposition 2.5. *Let* e_1, e_2 *be two insertion edges to the same graph* G. *Then their con-chains* $Q_G(e_1)$ *and* $Q_G(e_2)$ *are either disjoint, or they intersect (within the extended con-tree of* G *in which they lie) in one subpath.*

3 MEI Approximation Algorithm

We can now describe the main algorithm for our Theorem 1.2 (2). In the following, we will always consider the multiple edge insertion problem MEI(G, F) for a planar graph $G = (V, E)$ and an edge set F, $F \cap E = \emptyset$, with $k := |F| > 1$. Let $\Delta := \Delta(G)$ be the maximum degree in G. We will present an algorithm giving a solution to MEI(G, F) that approximates the optimum $\mathrm{ins}(G, F)$ within an *additive factor* (depending only on k, Δ). Afterward, in Section 4, we will discuss how to obtain an approximation algorithm for $\mathrm{cr}(G + F)$ based on this result.

On a high level, our algorithm proceeds as follows:

Algorithm 3.1. Computing an approximate solution to the multiple edge insertion problem MEI(G, F) for connected planar G.

(1) Build the con-tree $\mathcal{C} = \mathcal{C}(G)$.
(2) Using $\mathcal{C}$, compute single-edge insertions (including the con-chains $Q_G(e)$ of Definition 2.4) for each edge $e \in F$ independently, and centrally store their *embedding preferences* (Definition 3.4).
(3) Fix an embedding Γ of G by suitably (see Algorithm 3.7) combining the embedding preferences from step (2).

(4) Independently compute insertion paths for each edge $e \in F$ into the fixed embedding Γ.
(5) Realize all the insertion paths computed in the prior step.
(5)a) If some insertion paths cross multiple times, exchange subpaths, such that in the end all inserted edges cross each other at most once.

By using the aforementioned algorithms for building the decomposition tree and the single edge insertions as black boxes, we can directly perform the steps (1), (2), (4), and (5). We will discuss step (3) in Section 3.2. Yet, we can already informally describe the core idea of why the value $\mathrm{ins}^{\mathrm{Alg}}(G,F)$ of the outcome of Algorithm 3.1 approximates $\mathrm{ins}(G,F)$.

Clearly, $\mathrm{ins}^{\Sigma}(G,F) := \sum_{e\in F} \mathrm{ins}(G,e)$—the sum of the individual insertions without considering interdependency—is a lower bound for $\mathrm{ins}(G,F)$. Moreover, we can compute $\mathrm{ins}^{\Sigma}(G,F)$ exactly in step (2). Hence to give an approximation guarantee for Algorithm 3.1, it is enough to bound $\mathrm{ins}^{\mathrm{Alg}}(G,F)$ in terms of $\mathrm{ins}^{\Sigma}(G,F)$ and a function of k, Δ.

Let $e \in F$ be an inserted edge with the computed con-chain $Q_G(e)$, and $\nu \in V(Q_G(e))$ be a C-,P-, or R-node of $\mathcal{C}$ along it. An embedding preference of e at ν—actually respecting its neighborhood and generally denoted by a node tuple $\mathfrak{c}_\nu$—specifies what the embedding of G should locally "look like at ν" to achieve the (independent) optimum $\mathrm{ins}(G,e)$. Roughly, we call such a pair $(\mathfrak{c}_\nu, e)$ a *dirty pass* if, in the embedding Γ of step (3), the embedding preferences at ν and its neighbors has been fixed incompatibly from those individually chosen by e in the previous step (2), cf. Section 3.1 for details.

Theorem 3.2. *Consider a run of Algorithm 3.1 on G and F, with a particular embedding Γ fixed at step (3). Let $k = |F|$, $\Delta = \Delta(G)$. If r is the total number of dirty passes (over all $e \in F$) determined by this Γ, then the number of crossings in the outcome of the algorithm is*

$$\mathrm{ins}^{\mathrm{Alg}}(G,F) \le \mathrm{ins}^{\Sigma}(G,F) + \left\lfloor \frac{\Delta}{2} \right\rfloor \cdot r + \binom{k}{2}.$$

Sketch of proof. As every node of $\mathcal{C}$ with an embedding preference is associated with a 1- or 2-cut in G, any wrongly fixed preference (a dirty pass) can be "repaired" by rerouting the inserted edge close to this cut, crossing at most $\lfloor \Delta/2 \rfloor$ edges incident with a vertex in the cut. Summing over all dirty passes caused by Γ and taking possible crossings between edges of F into account, we get the bound. See [7] for the full proof. □

The embedding preferences for Γ can be naturally fixed such that, at every node ν of $\mathcal{C}$, not all con-chains of inserted edges disagree with Γ. Consequently, one can give an easy upper bound on r in terms of k as follows: For every dirty pass $(\mathfrak{c}_\nu, e)$ caused by Γ, there is another $f \in F$ such that ν belongs to the con-chain of f and, for a suitable tuple $\mathfrak{c}'_\nu$, $(\mathfrak{c}'_\nu, f)$ is not dirty. So, in particular, the con-chains $Q_G(e)$ and $Q_G(f)$ are not routed through the completely same neighborhood of ν (informally, they "split/merge" at ν), cf. Lemma 3.8 for the

concise statement. Since any two con-chains can split/merge at most twice, a coarse bound of $r = O(k^2)$ on the total number of disagreements with Γ follows.

Stated formally, the above arguments lead to the following conclusion—overall stronger than (1) of [10], with full details in Section 3.2 and 3.3:

Theorem 3.3 (Theorem 1.2 (2)). *Algorithm 3.1 computes a solution to the* MEI (G, F) *problem for connected planar* G *with* $\mathrm{ins}^{\mathrm{Alg}}(G, F)$ *crossings such that*

$$\mathrm{ins}^{\mathrm{Alg}}(G,F) \;\le\; \mathrm{ins}^{\Sigma}(G,F) + \left(2\left\lfloor\frac{\Delta}{2}\right\rfloor + 1\right)\cdot\binom{k}{2}$$

where $k = |F|$, $\Delta = \Delta(G)$, *and (also computed thereby)* $\mathrm{ins}^{\Sigma}(G,F) \le \mathrm{ins}(G,F)$ *is a lower bound on the optimal solution.*

3.1 Embedding Preferences and Estimating Additional Crossings via Dirty Passes

In order to discuss how to obtain an embedding Γ suitable for all edge insertions, we first have to concisely define *embedding preferences* that record the desired local embeddings of G from each independent single edge insertion in step (2).

Definition 3.4 (Embedding preference). *Consider a single edge insertion of an edge* $e \in F$ *into* G. *As argued in Section 2.2, the required embedding of* G *is fixed only for con-tree nodes along the con-chain* $Q = Q_G(e)$. *This requirement is encoded into the C-, P- and R-nodes along* Q; *for every such node* ν *we may store its* embedding preference $\pi_e(\nu)$:

R-nodes: *The skeleton* S_ν *of an R-node* ν *allows only a unique planar embedding and its mirror. We declare one of these two embeddings as the default one. The insertion algorithm then either sets* $\pi_e(\nu) :=$ DEFAULT *or* $\pi_e(\nu) :=$ MIRROR, *depending on which embedding it requires.*

P-nodes: *The skeleton of a P-node* ν *with* p *edges allows* $(p-1)!$ *planar embeddings given by different cyclic orderings of its edges around one of its nodes. When the con-chain* Q *passes through a P-node such that one of its neighbors is a C-node, the order of the edges is irrelevant, denoted by* $\pi_e(\nu) =$ IRRELEVANT. *Otherwise, the insertion path leaves one of the virtual edges of the skeleton* S_ν *and enters another one. Hence, these two edges should be neighbors in cyclic order, and the embedding preference is stored as a pair* $\pi_e(\nu) := (e_1, e_2)$ *which means the skeleton edge* e_1 *is to occur clockwise directly before* e_2.

C-nodes: *Let* $B_1, B_2 \subseteq G$ *be the two blocks neighboring a C-node* ν *on* Q. *The required embedding places (already embedded)* B_2 *into a face* ϕ_1 *of* B_1, *and vice versa* B_1 *into a face* ϕ_2 *of* B_2. *Unfortunately, those faces* ϕ_1, ϕ_2 *do not have standalone definitions within* G. *Let* μ_1, μ_2 *be the nodes adjacent to* ν *along* Q, *and let* x *be the cut vertex of* G *which is represented by* ν. *The insertion path within the skeleton* S_{μ_i}, $i = 1, 2$, *connects* x *to some*

other element (i.e., a vertex or a virtual edge) a_i. So, we set the embedding preference to $\pi_e(\nu) := \{a_1, a_2\}$. This means that we will be able to deduce the faces ϕ_1, ϕ_2 (by looking at the canonically computed local insertion subpaths) whenever S_{μ_1}, S_{μ_2} get fixed, and then embed these two faces into each other.

For all C-, P-, and R-nodes not on the con-chain Q, we do not store any embedding preference. Recall (e.g., from [11, 17]) that S-nodes—representing simple cycles—do not add additional embedding possibilities, and hence the above information is sufficient to determine an embedding of G which allows to insert the edge e with the minimum number of crossings.

Dirty pass. A central ingredient in our approach is the concept of a *dirty pass* $(\mathfrak{c}, e)$ with respect to some embedding Γ, where $\mathfrak{c}$ is a tuple with 1–3 nodes of $\mathcal{C}$ along the con-chain Q. Such structures pinpoint to places (short, possibly overlapping, sub-chains of Q) where no optimal insertion path for e can be realized, due to incompatibilities between Γ and the corresponding embedding preferences for e along the nodes of $\mathfrak{c}$. As Γ itself is encoded in terms of embedding preferences, recognizing a dirty pass boils down to a technical case distinction of preference comparisons; due to space restrictions, we refer to [7].

Theorem 3.2 now immediately follows from repeated application of next Lemma 3.5 to each $e \in F$, and the fact that Algorithm 3.1 computes optimal individual insertion paths within fixed Γ.

Lemma 3.5. *Consider a connected graph G with a plane embedding Γ and maximum degree Δ, and an edge e to insert. If there are altogether r_e dirty passes on the con-chain of e w.r.t. Γ, then e can be inserted into Γ with at most* $\mathrm{ins}(G, e) + r_e \cdot \lfloor \Delta/2 \rfloor$ *crossings.*

3.2 Combining All Embedding Preferences

We now want to find an embedding Γ that satisfies at least one preference per node of $\mathcal{C}$ and has the property that any pair of con-chains disagrees on at most two dirty passes. For each node in $\mathcal{C}$, we will store a *picked* embedding preference π_{pick}. In the end, these picked embedding preferences will be realized, to subsequently obtain the fixed embedding Γ of Algorithm 3.1, step (3).

Consider a chosen processing order $\langle e_1, e_2, \ldots, e_k \rangle$ of the edges of F: Initially, we set $\pi_{\text{pick}}(\nu) = \emptyset$ for all nodes $\nu \in V(\mathcal{C})$. We consider the edges one by one, setting $\pi_{\text{pick}}(\nu)$ for all nodes ν along the corresponding con-chain of the edge, and probably alter other embedding preferences along the previously considered con-chains (see below for details). After the insertion of the i-th edge, we have the property that each node ν along any con-chain of an edge $e_{i'}$ with $i' \leq i$ has a well-defined $\pi_{\text{pick}}(\nu) \neq \emptyset$.

Let $\mathcal{N}^{<i} \subseteq V(\mathcal{C})$ denote the nodes of $\mathcal{C}$ that have embedding preferences from the first $i-1$ inserted edges ($1 \leq i \leq k+1$). The individual trees within the forest induced by any $\mathcal{N}^{<i}$ give rise to a node partition $\dot{\bigcup}_{j=1}^{\ell} N_j^{<i} = \mathcal{N}$ with $\ell < i$. We call any partition set $N_j^{<i}$ an *embedding part*. Generalizing on the flipping of a single con-chain we can observe:

Proposition 3.6. *Let $N_j^{<i}$ be any embedding part with the embedding preferences π_{pick}. When all the nodes of $N_j^{<i}$ are flipped, we obtain new embedding preferences π'_{pick}. Then, an embedding realizing π'_{pick} allows an insertion of the first $i-1$ insertion edges with the same number of crossings as for π_{pick}. In fact, these insertion paths are identical to the former ones up to mirroring.*

We are now ready to give the following method to obtain a merged embedding Γ. Let π_i be the embedding preferences along the con-chain $Q_i = Q_G(e_i)$ stored in step (2) of Algorithm 3.1.

Algorithm 3.7. Combining all embedding preferences to obtain Γ.

A. Let $\pi_{\text{pick}}(\nu) = \emptyset$, $\forall \nu \in V(\mathcal{C})$.
B. For all $1 \le i \le k$:
   (a) Traverse the con-chain Q_i of e_i along its predefined orientation; let $\nu \in Q_i$ be the first node.
   (b) If $\pi_{\text{pick}}(\nu) = \emptyset$, then choose $\pi_{\text{pick}}(\nu) := \pi_i(\nu)$.
   (c) Let μ be the closest non-S-node preceding ν. Skip this step if μ does not exist or ν is a C-node. Otherwise:

   Check if a flip can improve the embedding: The preference π_{pick} is *improvable* if ν can be the last element of a tuple $\mathfrak{c}$ such that $(\mathfrak{c}, e_i)$ is in a dirty pass w.r.t. Γ, while this tuple would not be dirty after flipping $\pi_{\text{pick}}(\nu)$ (which is equivalent to flipping ν's predecessors).

   If the embedding is improvable, let $Q' \subset V(Q_i)$ be the consecutive nodes of Q_i starting from the start node up to (and including) μ. Furthermore, let N' be all embedding parts of $N^{<i}$ that contain at least one node of Q'. Flip π_{pick} for all nodes $Q' \cup N'$.
   (d) If $\pi_{\text{pick}}(\nu)$ has been newly set in (b), let $\nu' := \nu$. Otherwise let $N_j^{<i}$, for some j, be the embedding part to which ν belonged, and set ν' to the last non-S-node in $N_j^{<i}$ that is traversed by Q.

   Set ν to be the closest non-S-node succeeding ν'; if it exists, continue with step (b), otherwise proceed with the next i in step B.

C. Choose a random embedding preference for any node ν with $\pi_{\text{pick}} = \emptyset$, and randomly complete the embedding preference of any P-node to a complete cyclic ordering. Realize all the preferences π_{pick} to obtain Γ.

Consider the dirty passes that arise from Algorithm 3.7. Thanks to Proposition 3.6 it is easy to see that the decision whether a pass $(\mathfrak{c}, e_j)$ would be dirty in the final solution is made by the algorithm exactly at the step when merging π_j into π_{pick}^{j-1}. Note that, due to the tree-property of $\mathcal{C}$ (cf. Proposition 2.5), any two con-chains Q, Q' can have at most one common (connected) sub-chain q. The two non-S-nodes closest to either end of q are the two *split nodes* of (Q, Q'). We say a tuple (ν, j, i), $j < i$, is a *splitter* (w.r.t. i) if ν is a split node w.r.t. (Q_j, Q_i). Observe that multiple splitters may induce the split node property of the same node in $\mathcal{C}$. Our key conclusion here reads:

Lemma 3.8. *The above Algorithm 3.7 guarantees that there is at most one dirty pass for each splitter (over all pairs of con-chains); this dirty pass then also contains the corresponding split node. — Hence, the overall sum of dirty passes in the embedding Γ (obtained by Algorithm 3.7) is at most $2\binom{k}{2}$.*

3.3 Runtime Analysis of Algorithm 3.1

As mentioned in Section 2, we can build the con-tree in linear time $O(|V|)$, based on the linear-time decomposition algorithm [21]. In step (2), we call the $O(|V|)$ insertion algorithm k times. In full [7], we discuss how to implement the merge algorithm (Algorithm 3.7, called in step (3) of the main algorithm) so that it takes at most $O(k|V|)$ time. In step (4), we then run k BFS algorithms, requiring $O(|V|)$ time each. By using suitable tie-breaking, step (5)a) will not be necessary, and since each edge has at most $O(|V| + k)$ crossings in the end, the realization may require up to $O(k|V(G)| + k^2)$ time, which therefore also constitutes the overall runtime bound of the algorithm.

4 Crossing Number Approximations

Our main concept of interest is the crossing number of the graph $G + F$. We can combine our above result with a result of [8], connecting the optimal crossing number with the problem of multiple edge insertion.

Theorem 4.1 (Chimani et al. [8]). *Consider a planar graph G and an edge set F, $F \cap E(G) = \emptyset$. The value* $\mathrm{ins}(G, F)$ *of an optimal solution to* $\mathrm{MEI}(G, F)$ *satisfies*

$$\mathrm{ins}(G, F) \;\leq\; 2|F| \cdot \left\lfloor \frac{\Delta(G)}{2} \right\rfloor \cdot \mathrm{cr}(G + F) + \binom{|F|}{2}$$

where $\mathrm{cr}(G+F)$ *denotes the (optimal) crossing number of the graph G including the edges F, and* $\binom{|F|}{2}$ *thereby accounts for crossings between the edges of F.*

Notice that, when considering the crossing number problem of $G + F$, we may assume G to be connected—otherwise we could "shift" some edges of F to G). Let $k = |F|$, $\Delta = \Delta(G)$. Plugging the estimate of Theorem 4.1 into the place of $\mathrm{ins}^{\Sigma}(G, F) \leq \mathrm{ins}(G, F)$ in Theorem 3.3, and realizing that the $\binom{k}{2}$ term in both estimates stands for the same set of crossings, we immediately obtain

$$\begin{aligned} \mathrm{ins}^{\mathrm{Alg}}(G, F) \;&\leq\; 2k \cdot \left\lfloor \frac{\Delta}{2} \right\rfloor \cdot \mathrm{cr}(G + F) + \binom{k}{2} + 2 \left\lfloor \frac{\Delta}{2} \right\rfloor \cdot \binom{k}{2} \\ &=\; \lfloor \Delta/2 \rfloor \cdot 2k \cdot \mathrm{cr}(G + F) + \big(\lfloor \Delta/2 \rfloor + 1/2\big)(k^2 - k). \end{aligned}$$

Hence we can give the outcome of Algorithm 3.1 as an approximate solution to the crossing number problem on $G + F$, proving:

Theorem 4.2 (Theorem 1.2 (3)). *Given a planar graph G and an edge set F, $F \cap E(G) = \emptyset$, Algorithm 3.1 computes, in $O(|F| \cdot |V(G)| + |F|^2)$ time, a solution to the $\mathrm{cr}(G+F)$ problem with the following number of crossings*

$$\mathrm{cr}^{\mathrm{aprx}}(G+F) \leq \lfloor \Delta(G)/2 \rfloor \cdot 2|F| \cdot \mathrm{cr}(G+F) + \left(\lfloor \Delta(G)/2 \rfloor + \tfrac{1}{2}\right)\left(|F|^2 - |F|\right).$$

In [18], furthermore, an algorithm is presented to approximate the crossing number of graphs embeddable in any fixed higher orientable surface. This algorithm lists the technical requirement that G has a "sufficiently dense" embedding on the surface. Yet, as noted in [18], a result like Theorem 4.2 allows to drop this requirement: If the embedding density is small, then the removal of the offending small set(s) of edges is sufficient to reduce the graph genus, while the removed edges can be later inserted into an intermediate planar subgraph of the algorithm.

5 A Note on the Planarization Heuristic

The currently practically strongest heuristic [16] for the crossing number problem is the *planarization heuristic* which starts with a maximal planar subgraph of the given non-planar graph, and then iteratively performs single edge insertions. The crossings of such an insertion are then replaced by dummy nodes such that each edge is inserted into a planar graph. Due to its practical superior performance, often giving the optimal solution [5, 14], it was an open question if this approach unknowingly guarantees some approximation ratio.

By investigating our strategy and proofs, it becomes clear that this approach as such cannot directly give an approximation guarantee: by routing an edge (in an R-node) through another virtual edge (representing a subgraph S) and replacing the crossings with dummy nodes, you essentially *fix* (most of) the embedding of S. This fix might result in $O(n)$ embedding restrictions for further edge insertions, without having an edge that requires this embedding. Therefore the number of dirty passes can no longer be bounded by a function in k. Yet, an implementation realizing the planarization heuristic already contains all the ingredients to obtain our approximation; one "only" has to compute all insertion paths and fix an accordingly merged embedding (Algorithm 3.7), before running the fixed-embedding edge insertion subalgorithm for all inserted edges.

6 Conclusions

We have presented a new approximation algorithm for the multi-edge insertion problem which is faster and simpler that the only formerly known one [10], while at the same time giving better bounds; in fact, in contrast to the former multiplicative approximation, it is the first one with an additive bound. Our algorithm directly leads also to improved approximations (even constant ratio ones over a large class of inputs) for the crossing number problem of graphs in which a given set of edges can be removed in order to obtain a planar subgraph, and for graphs that can be embedded on a surface of some fixed genus.

We conclude with an interesting open problem. We know that multi-edge insertion is NP-hard when the number of inserted edges is part of the input, and it is linear time solvable for the special case of inserting a single edge. What is the complexity of optimally inserting a *constant* number of edges?

References

1. Arora, S., Rao, S., Vazirani, U.: Expander flows, geometric embeddings and graph partitioning. J. ACM 56, 5:1–5:37 (2009)
2. Bienstock, D., Monma, C.L.: On the complexity of embedding planar graphs to minimize certain distance measures. Algorithmica 5(1), 93–109 (1990)
3. Cabello, S., Mohar, B.: Crossing and weighted crossing number of near-planar graphs. In: Tollis, I.G., Patrignani, M. (eds.) GD 2008. LNCS, vol. 5417, pp. 38–49. Springer, Heidelberg (2009)
4. Cabello, S., Mohar, B.: Adding one edge to planar graphs makes crossing number hard. In: Proc. SoCG 2010, pp. 68–76. ACM, New York (2010)
5. Chimani, M.: Computing Crossing Numbers. PhD thesis, TU Dortmund, Germany (2008), http://www.ae.uni-jena.de/alenmedia/dokumente/ComputingCrossingNumbers_PhDthesis_Chimani_pdf.pdf
6. Chimani, M., Gutwenger, C., Mutzel, P., Wolf, C.: Inserting a vertex into a planar graph. In: Proc. SODA 2009, pp. 375–383 (2009)
7. Chimani, M., Hliněný, P.: A tighter insertion-based approximation of the crossing number. Full version. arXiv:1104.5039 (2011)
8. Chimani, M., Hliněný, P., Mutzel, P.: Approximating the crossing number of apex graphs. In: Tollis, I.G., Patrignani, M. (eds.) GD 2008. LNCS, vol. 5417, pp. 432–434. Springer, Heidelberg (2009)
9. Chuzhoy, J.: An algorithm for the graph crossing number problem. In: Proc. STOC 2011 (to appear, 2011)
10. Chuzhoy, J., Makarychev, Y., Sidiropoulos, A.: On graph crossing number and edge planarization. In: Proc. SODA 2011, pp. 1050–1069. ACM Press, New York (2011)
11. Di Battista, G., Tamassia, R.: On-line planarity testing. SIAM Journal on Computing 25, 956–997 (1996)
12. Even, G., Guha, S., Schieber, B.: Improved approximations of crossings in graph drawings and vlsi layout areas. SIAM J. Comput. 32(1), 231–252 (2002)
13. Gitler, I., Hliněný, P., Leanos, J., Salazar, G.: The crossing number of a projective graph is quadratic in the face-width. Electronic Notes in Discrete Mathematics 29, 219–223 (2007)
14. Gutwenger, C.: Application of SPQR-Trees in the Planarization Approach for Drawing Graphs. PhD thesis, TU Dortmund, Germany (2010)
15. Gutwenger, C., Mutzel, P.: A linear time implementation of SPQR-trees. In: Marks, J. (ed.) GD 2000. LNCS, vol. 1984, pp. 77–90. Springer, Heidelberg (2001)
16. Gutwenger, C., Mutzel, P.: An experimental study of crossing minimization heuristics. In: Liotta, G. (ed.) GD 2003. LNCS, vol. 2912, pp. 13–24. Springer, Heidelberg (2004)
17. Gutwenger, C., Mutzel, P., Weiskircher, R.: Inserting an edge into a planar graph. Algorithmica 41(4), 289–308 (2005)
18. Hliněný, P., Chimani, M.: Approximating the crossing number of graphs embeddable in any orientable surface. In: Proc. SODA 2010, pp. 918–927 (2010)

19. Hliněný, P., Salazar, G.: On the crossing number of almost planar graphs. In: Kaufmann, M., Wagner, D. (eds.) GD 2006. LNCS, vol. 4372, pp. 162–173. Springer, Heidelberg (2007)
20. Hliněný, P., Salazar, G.: Approximating the crossing number of toroidal graphs. In: Tokuyama, T. (ed.) ISAAC 2007. LNCS, vol. 4835, pp. 148–159. Springer, Heidelberg (2007)
21. Hopcroft, J.E., Tarjan, R.E.: Dividing a graph into triconnected components. SIAM Journal on Computing 2(3), 135–158 (1973)
22. Tutte, W.T.: Connectivity in graphs. Mathematical Expositions, vol. 15. University of Toronto Press (1966)
23. Vrt'o, I.: Crossing numbers of graphs: A bibliography (2011), ftp://ftp.ifi.savba.sk/pub/imrich/crobib.pdf
24. Ziegler, T.: Crossing Minimization in Automatic Graph Drawing. PhD thesis, Saarland University, Germany (2001)

Linear-Space Approximate Distance Oracles for Planar, Bounded-Genus and Minor-Free Graphs*

Ken-ichi Kawarabayashi[1,**], Philip N. Klein[2,***], and Christian Sommer[3]

[1] NII, Tokyo, Japan
[2] Brown U, Providence RI
[3] MIT, Cambridge MA

Abstract. A $(1+\epsilon)$–approximate distance oracle for a graph is a data structure that supports approximate point-to-point shortest-path-distance queries. The most relevant measures for a distance-oracle construction are: space, query time, and preprocessing time.

There are strong distance-oracle constructions known for planar graphs (Thorup, JACM'04) and, subsequently, minor-excluded graphs (Abraham and Gavoille, PODC'06). However, these require $\Omega(\epsilon^{-1} n \lg n)$ space for n–node graphs.

In this paper, for planar graphs, bounded-genus graphs, and minor-excluded graphs we give distance-oracle constructions that require only $O(n)$ space. The big O hides only a fixed constant, independent of ϵ and independent of genus or size of an excluded minor. The preprocessing times for our distance oracle are also faster than those for the previously known constructions. For planar graphs, the preprocessing time is $O(n \lg^2 n)$. However, our constructions have slower query times. For planar graphs, the query time is $O(\epsilon^{-2} \lg^2 n)$.

For all our linear-space results, we can in fact ensure, for any $\delta > 0$, that the space required is only $1+\delta$ times the space required just to represent the graph itself.

1 Introduction

A $(1+\epsilon)$–*approximate distance oracle* for a graph is a data structure that supports point-to-point approximate distance queries. A distance-oracle construction for a family of graphs has three complexity measures:

- *preprocessing time:* time to build the data structure,
- *space:* how much space is occupied by the data structure, and
- *query time:* how long does it take for a query to be answered.

* An extended version can be found online [KKS11].

** Research partly supported by Japan Society for the Promotion of Science, Grant-in-Aid for Scientific Research, by C & C Foundation, by Kayamori Foundation and by Inoue Research Award for Young Scientists.

*** Supported in part by National Science Foundation Grant CCF-0964037.

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 135–146, 2011.

Each of these quantities might depend on the *stretch* parameter $1+\epsilon$ (which is defined as the maximum ratio over all pairs of nodes of the query output divided by the length of a shortest path) as well as the size of the graph.

For general graphs, for stretch less than 2, no approximate distance oracle is known that achieves subquadratic space and sublinear query time.

The only known constructions that achieve $(1+\epsilon)$ stretch are for restricted families of graphs: planar graphs [Tho04], minor-excluded graphs [AG06], and graphs of low doubling dimension [Tal04, HPM06, Sli07, BGK+10].

One obstacle to the widespread adoption of this technique may have been the space requirements of known distance oracles. Even the most compact distance oracle of Thorup [Tho04] requires $\Omega(\epsilon^{-1} n \lg n)$ space for n–node graphs. Even though the constant is quite modest, the storage required is rather large [MZ07].

Our Contribution

In this paper, for every family of graphs for which a nontrivial $(1+\epsilon)$–approximate distance oracle is known (planar, bounded genus, H–minor-free, bounded doubling dimension), we give such a distance oracle that in addition requires only linear space[1]. In fact, for any $\delta > 0$, there is such a distance oracle whose space requirement is only $1+\delta$ times the space required just to store the graph itself; thus the overhead due to the distance oracle is in essence negligible.

We achieve this while increasing the query time by a factor that is almost proportional to the decrease in space[2]. For planar graphs, the query time of our oracle is $O(\epsilon^{-2} \lg^2 n)$. Although the query time for our constructions is slower than that for the superlinear-space constructions, the increase may be partly made up for by the decrease in actual time due to better memory performance (because of the memory hierarchy). The space/query-time tradeoff is tunable, so the construction can be adapted to a particular architecture.

For bounded-genus graphs, there was previously no distance-oracle construction known other than that implied by the minor-excluded construction, for which the constant is enormous and the preprocessing time is a high-degree polynomial. We give a more efficient construction tailored to graphs of genus g.

A summary of our results is given in Table 1.

2 Previous Work on Approximate Distance Oracles

General and sparse graphs. Thorup and Zwick [TZ05] gave asymptotically almost optimal trade-offs for distance oracles for general undirected graphs, proving that for any graph and for any integer k there is a $(2k-1)$–approximate distance oracle using space $O(kn^{1+1/k})$ and query time $O(k)$. They also prove that, if stretch strictly less than $2k+1$ is desired, then $\Omega(n^{1+1/k})$ bits of space are necessary. A slightly weaker lower bound holds for sparse graphs: Sommer,

[1] Our results for bounded-doubling-dimension graphs hold only for unit lengths.

[2] The product of space times query time for our oracle is $O(n\epsilon^{-2} \lg^2 n)$ while that same product is $O(n\epsilon^{-2} \lg n)$ for [Tho04].

Table 1. Time complexities of our linear-space $(1+\epsilon)$–approximate distance oracles. N denotes the largest integer weight. Due to space restrictions, some are in [KKS11].

Graph Class	Preprocessing	Query	
Planar Undirected	$O(n \lg^2 n)$	$O(\epsilon^{-2}(\lg n)^2)$	Theorem 1
Planar Directed	$O(n(\lg(nN))(\lg n)^3\epsilon^{-2})$	$O((\epsilon^{-1}(\lg n)(\lg(nN)))^2)$	[KKS11]
Reachability Oracle	$O(n \lg n)$	$O(\lg^2 n)$	[KKS11]
Genus g	$O(n(\lg n)(g^3 + \lg n))$	$O(\epsilon^{-2}(\lg n + g)^2)$	Theorem 2
H–minor-free	$O(poly(n, \epsilon))$	$O(\epsilon^{-2}(\lg n)^2)$	[KKS11]
α–doubling, unit lengths	$\epsilon^{-O(\alpha)}O(poly(n))$	$\epsilon^{-O(\alpha^2)} \cdot (\lg n)^{O(\alpha)}$	[KKS11]

Verbin, and Yu [SVY09] prove that a distance oracle with stretch k and query time t requires space $n^{1+\Omega(1/(kt))}$ (up to poly-logarithmic factors). The oracle with the best stretch factor is by Pătraşcu and Roditty [PR10], who recently gave a 2–approximate distance oracle using space $O(n^{5/3})$ on sparse graphs. Distance oracles with stretch strictly less than 2 have not been achieved for general graphs.

Restricted graph classes. For restricted classes of graphs, better distance oracles are known and stretch $1+\epsilon$ can be achieved.

Thorup [Tho04] presents efficient $(1+\epsilon)$–approximate distance oracles for planar digraphs. (There is a slight improvement [Kle05] to the preprocessing time for one case.) Table 2 lists these results.

There are also many results on exact distance oracles for planar graphs. The best is that of Fakcharoenphol and Rao [FR06] and its subsequent improvements [Kle05, KMW10, MWN10] and variants [MS10, Nus10]. Faster per-query time can be achieved by using more space [Cab06, MS10]. However, all these results require polynomial (but sublinear) query time. There are also results on special cases of planar graphs and special kinds of queries [DPZ00, CX00, KK06].

Abraham and Gavoille [AG06] extend Thorup's result to minor-free graphs. After a polynomial-time preprocessing step, point-to-point queries can be answered in time $O(\epsilon^{-1} \lg n)$ using a data structure of size $O(n\epsilon^{-1} \lg n)$.

Table 2. Time and space complexities of $(1+\epsilon)$–approximate distance oracles for planar graphs on n nodes. In this table, the largest integer weight is assumed to be polynomial in n. The upper part lists results for directed, the lower part lists results for undirected planar graphs.

Preprocessing	Space	Query	Reference
$O(n \lg^4 n\epsilon^{-2})$	$O(n\epsilon^{-1} \lg^2 n)$	$O(\lg \lg(n) + \epsilon^{-1})$	[Tho04, Thm. 3.16]
$O(n \lg^3 n\epsilon^{-1})$	$O(n\epsilon^{-1} \lg^2 n)$	$O(\lg n(\lg \lg n + \epsilon^{-1}))$	[Tho04, Prop. 3.14]
$O(n \lg^2 n(\lg n + \epsilon^{-1}))$	$O(n\epsilon^{-1} \lg^2 n)$	$O(\lg n(\lg \lg n + \epsilon^{-1}))$	[Kle05, Sec. 7]
$O(n(\lg n)^3\epsilon^{-2})$	$O(n\epsilon^{-1} \lg n)$	$O(\epsilon^{-1})$	[Tho04, Thm. 3.19]
$O(n(\lg n)^2\epsilon^{-1})$	$O(n\epsilon^{-1} \lg n)$	$O(\epsilon^{-1} \lg n)$	[Tho04, Implicit]

3 Tunable Approximate Distance Oracle for Planar Graphs

We prove our main theorem. The description of the improved preprocessing algorithm can be found in its own section (Section 4).

Theorem 1. *For any undirected planar graph G with non-negative edge weights there exists a $(1+\epsilon)$–approximate distance oracle with query time $O(\epsilon^{-2}\lg^2 n)$, linear space, and preprocessing time $O(n\lg^2 n)$.*

3.1 Review of Thorup's Distance Oracle

We briefly review a variant of Thorup's distance oracle for undirected graphs (using somewhat different terminology)[3].

There are two core ideas. The first is *approximately representing shortest paths that intersect a shortest path*. Let P be a shortest path in a graph G. A pair (p, v) of nodes where p is in P and v is in G is a *connection for v with respect to P*. A set $\mathcal{C}$ of such connections *covers v in G with respect to P* if, for every node p of P, there is a connection (p', v) in $\mathcal{C}$ such that

$$\text{dist}(p, p') + \text{dist}(p', v) \leq (1+\epsilon)\,\text{dist}(p, v) \tag{1}$$

Let u, v be nodes of the input graph. Let Q be the shortest u-to-v path that intersects P. Suppose $\mathcal{C}$ is a set of connections that covers u and v with respect to P. Then it contains connections $(p, u), (p', v)$ such that

$$\text{dist}(u, p) + \text{dist}(p, p') + \text{dist}(p', v) \leq (1+\epsilon)\text{length}(Q) \tag{2}$$

Thorup gives an algorithm that, given a (mostly) planar graph G and a shortest path P, computes a set $\mathcal{C}$ of connections that covers all nodes of G and that has $O(\epsilon^{-1})$ connections per node v. In Section 4, we give an algorithm that achieves a faster[4] running time by covering only a subset of the nodes of G. The distance oracle involves storing with each node v the connections that cover v with respect to several shortest paths (and the distances associated with these connections). The storage required for v thus has size $O(\epsilon^{-1})$ times the number of such paths.

The second idea is *recursively decomposing a planar graph with shortest-path separators*. This idea is based on a lemma in [LT79] stating that, for any spanning tree T in a planar graph in which every face is a triangle, there is a nontree edge e such that the unique simple cycle in $T \cup \{e\}$ is a balanced separator. The nodes of this separator comprise two paths in T.

[3] This variant does not appear in [Tho04] but is an obvious simplification, analogous to that of [Tho04, Proposition 3.14] (which applies to directed graphs) resulting in slower query times. Since our query time is slower anyway, this simplified variant suffices.

[4] Our algorithm depends on G being wholly planar (as opposed to mostly planar as in [Tho04], which is the case for the variant we address.

The distance-oracle construction uses this lemma with T being a shortest-path tree to recursively decompose the input graph. The recursive decomposition defines a binary *decomposition* tree in which each node x is labeled by (i) a subgraph $G(x)$ of the input graph and (ii) the separator $S(x)$ used to decompose $G(x)$, if x is not a leaf. If x is the root, $G(x)$ is the input graph. If x has children y and z, removing the separator $S(x)$ from $G(x)$ results in two separated subgraphs, $G(y)$ and $G(z)$. If x is a leaf, $G(x)$ consists of one node.

Each input-graph node v is associated with some decomposition-tree node, namely, the leafmost node x whose subgraph includes v. We say that the ancestors of x are *relevant* to v. Thus each input-graph node v has $O(\lg n)$ relevant tree-nodes. The distance oracle assigns a label to v that consists of a set of connections; for each tree-node x relevant to v, for each of the two paths P comprising the separator $S(x)$, the distance oracle stores a set of connections that cover v in $G(x)$ with respect to P. It follows that the label of v has size $O(\epsilon^{-1} \lg n)$.

Next we show that these labels suffice to estimate point-to-point distances. We say that a tree-node x is *relevant* to a path Q if $S(x)$ contains a node of Q, and is the *most relevant* if x is the rootmost relevant tree node. If x is the tree-node most relevant to Q then $G(x)$ contains Q. Let u, v be any pair of input-graph nodes, and let Q be the shortest u-to-v path. Let x be the tree-node most relevant to Q. Then $G(x)$ contains Q, and at least one of the paths comprising the separator $S(x)$, say P, intersects Q. It follows from (2) that the u-to-v distance is approximately

$$\text{dist}(u, p) + \text{dist}(p, p') + \text{dist}(p', v) \tag{3}$$

for two nodes p, p' on P. To estimate the u-to-v distance, therefore, the following procedure suffices: for every tree-node x that is relevant to u and v, compute the minimum of (3) over connections (p, u) and connections (p', v) where p and p' belong to one of the two paths comprising $S(x)$. This takes time proportional to the number of such connections.

3.2 Our Compact Distance Oracle

Our linear-space construction draws on another kind of recursive decomposition using separators. Frederickson [Fre87] introduced the notion of an r–division, which is a partition of the edges into edge-induced subgraphs (called *regions*) such that each region contains $O(r)$ edges and the number of boundary nodes in each region is at most $O(\sqrt{r})$. An r–division can be computed in time $O(n \lg n)$.

Before carrying out the recursive decomposition with shortest-path separators, our preprocessing algorithm computes an r–division for $r = \ell^2$ (where ℓ is a parameter). Subsequently, connections (v, w) are only stored for those nodes v that are boundary nodes of the r–division. Since there are $O(n/\sqrt{r})$ boundary nodes, the connections and associated distances require storage $O((n\epsilon^{-1} \lg n)/\sqrt{r})$. We choose $\ell = \Theta(\epsilon^{-1} \lg n)$ so the total storage is $O(n)$.

An *s-to-t query* is handled as follows. First, compute shortest-path distances from s to all the nodes in s's region R_s. This takes $O(\ell^2)$ time [HKRS97].

At this point, the query algorithm has distances in the subgraph R_s from s to all the boundary nodes of R_s (and to t, if t is in R_s). There are $O(\ell)$ such boundary nodes. Similarly, compute shortest-path distances to t from all the nodes in t's region R_t, obtaining distances in the subgraph R_t to t from all the boundary nodes of R_t.

Let A, B be, respectively, the set of connections for boundary nodes of R_s, R_t. For each separator path P that has connections in A and B, the procedure described in Section 3.2 finds the shortest s-to-t path that enters P via a connection of A and leaves P via a connection of B. The time is linear in the number of such connections (see also [Tho04, Section 3.2.2] and [Tho04, Lemma 3.6]). Since each of the $O(\ell)$ boundary nodes of R_s and R_t has $O(\epsilon^{-1} \lg n)$ connections, the total time for these computations is $O(\ell\epsilon^{-1} \lg n)$.

Finally, return the minimum overall path-length (including the s-to-t distance within R_s, if t belongs to R_s). The total time for handling the query is $O(\ell^2 + \ell\epsilon^{-1} \lg n)$.

We now explain how we find the shortest s-to-t path that enters P via a connection of A and leaves P via a connection of B. The method is a generalization of that in [Tho04, Sections 3.2.1 and 3.2.2].

For each connection (b, p) in A, b is a boundary node of R_s and we have $\text{dist}(s, b)$. For each connection (b, p) in B, b is a boundary node of R_t and we have $\text{dist}(t, b)$. Let C be the sequence of all connections (s, p) and (t, p) in $A \cup P$, sorted according to the position of p on P. We use the following procedure.

initialize $m_s, m_t, d := \infty$
initialize $\hat{p} := p_0$
for each connection (p, b) in C in order,
    $m_s := m_s + \text{dist}(\hat{p}, p)$
    $m_t := m_t + \text{dist}(\hat{p}, p)$
    $\hat{p} := p$
    if b is a boundary node of R_s,
        $m_s := \min\{m_s, \text{dist}(s, b) + \text{dist}(b, p)\}$
    if b is a boundary node of R_t
        $m_t := \min\{m_t, \text{dist}(t, b) + \text{dist}(b, p)\}$
    $d := \min\{d, m_s + m_t\}$
return d

The procedure requires time O(number of connections considered). The procedure maintains the invariant that, after a node $\hat{p}$ of P has been considered in the loop, m_s is the length of the shortest s-to-$\hat{p}$ path of the form that goes via a boundary node b of R_s and a connection (b, p) and then travels along P from p to $\hat{p}$, where p appears before $\hat{p}$ on P. A similar statement holds for m_t. It follows that the value d returned by the procedure is the length of the shortest s-to-t path that travels in R_s to a boundary node b of R_s, then goes to P via a connection for b, then travels along P then leaves P via a connection for a boundary node b' of R_t then travels to t within R_t.

4 Improved Preprocessing Algorithm

Thorup's preprocessing algorithm for his undirected construction takes time $O(n\epsilon^{-2}\lg^3 n)$ (as stated in [Tho04, Theorem 3.19]). We give a preprocessing scheme for our construction that takes time $O(n\lg^2 n)$, independent of ϵ. We give details later, but here we observe that the factor $O(\epsilon^{-2}\lg n)$ speedup has three sources.

First, since we are not aiming for a query time of $O(\epsilon^{-1})$, we can use a simpler preprocessing approach than the one underlying [Tho04, Theorem 3.19]; we use the approach that for directed graphs underlies [Tho04, Proposition 3.14]. The corresponding bound for undirected graphs is listed in Table 2 as "implicit."

Second, we only need to compute connections for a small subset of the nodes (the boundary nodes of the r–division). That in itself does not seem to permit an additional speedup using Thorup's method since his algorithm depends not on the number of connections stored but on the sizes of the graphs searched. Therefore, third, in addition we use another approach to finding connections, one based on the multiple-source shortest-path (MSSP) algorithm of Klein [Kle05] or that of Cabello and Chambers [CC07].

PREPROCESS(G_0)
  let B_0 be the set of boundary nodes of an r–division [Fre87]
  let T be a shortest-path tree
  compute recursive decomposition based on cycle separators $T \cup \{e\}$
  for each nonroot node x of recursive-decomposition tree,
    for each path P_i $(i = 1, 2)$ comprising $S(x)$,
      compute connections for nodes of B_0 in $G(x)$ with respect to P_i

The last step, computing the connections for nodes of B_0 in $G(x)$ with respect to P_i, works on a graph $G'(x)$ obtained from $G(x)$ by cutting along P_i, duplicating the nodes and edges of P_i and creating a new face whose boundary consists of the two copies of P_i. This modification destroys paths that cross P_i but such paths are not needed since P_i is a shortest path. It has the advantage that, for each copy P of P_i, in $G'(x)$ all nodes of P lie on a common face.

For each copy P, there is a computation that selects connections (p, v) for specified nodes v with respect to that copy. The computation uses an algorithm called PATH(G, B, P) that takes time $O((|G|+\text{number of connections})\lg|G|)$ and selects $O(\epsilon^{-1})$ connections per node $v \in B$. Since there are two copies of two paths comprising $S(x)$, the last step of PREPROCESS selects $O(\epsilon^{-1})$ connections per node of B_0 in $G(x)$. Therefore the total number of connections for B_0 is $O(\epsilon^{-1}\lg n)$, and the total time is $O(n\lg^2 n + |B_0|\,\epsilon^{-1}\lg n)$, which is $O(n\lg^2 n)$.

Now we describe PATH(G, B, P). Let the nodes of P be $p_0 \ldots p_s$. First the algorithm computes $i(v) = \text{argmin}_i \text{dist}(p_i, v)$ and $d_v = \min_i \text{dist}(p_i, v)$. These can be computed using a single-source shortest-path computation in the graph obtained by zeroing out the lengths of the edges of P.

For $i = 0, 1, \ldots, s$, let T_i denote the shortest-path tree rooted at p_i. For $i > 0$, let T_i' be the tree obtained from T_{i-1} by removing the parent edge of p_i and

adding the edge $p_i p_{i-1}$, obtaining a p_i–rooted tree (not a shortest-path tree). For $i > 0$, let σ_i denote a sequence of edges whose insertion into T_i' (followed by the ejection of each corresponding parent edge) result in T_i. Klein [Kle05] shows that each edge is inserted at most once, and gives an $O(|G| \lg |G|)$ algorithm (the *multiple-source shortest-path algorithm*) to compute these sequences. For each such inserted edge uv, the algorithm also computes the resulting change Δ_{uv} in the length of the root-to-v path in the tree. Cabello and Chambers [CC07] give a simplification of the multiple-source shortest-path algorithm and generalize it to bounded-genus in $O(g^2 |G| \lg |G|)$ time. The algorithm PATH uses one of these algorithms to compute the sequences σ_i and the corresponding length changes Δ_{uv}.

The remainder of PATH consists of two phases, FORWARD and BACKWARD. A connection (p_i, v) might be added by FORWARD if $i > i(x)$ and by BACKWARD if $i < i(x)$. We describe FORWARD. BACKWARD is symmetric.

The algorithm FORWARD iterates through the nodes $p_0, \ldots, p_s$ of P, maintaining a tree T that is, in turn, $T_0, T_1, \ldots, T_s$. The tree T is represented using a dynamic-tree data structure [ABH+04, AHdLT05, Fre97, ST83, TW05]. A node-labeling is maintained: $\mu(v)$ is a quantity (discussed later) that is used to decide whether v needs a new connection. This labeling is represented implicitly, as is typical in dynamic trees, so as to support bulk updates. In this case (somewhat atypically), an update takes the form "add a quantity Δ to the label of every tree in the subtree rooted at u." Each update takes $O(\lg n)$ amortized time. In addition, searching for a node v that has $\mu(v) \leq 0$ takes $O(\lg n)$ time.

```
FORWARD(G, B, P):
    initialize T := T_0
    for every node v, initialize μ(v) := ∞
    for i = 0, 1, 2, ..., s:
        comment: T is rooted at p_i
⋆       for each node v ∈ B such that either i(v) = i or μ(v) ≤ 0 ,
            create a connection (p_i, v)
            set μ(v) := ε d_v
        if i < s,
            comment: now change the root...
            remove parent edge of p_{i+1} and add edge p_{i+1} p_i
            comment: now make the tree a shortest-path tree
†           for each edge uv in the sequence σ_{i+1},
                remove the current parent edge of v in T, and add uv
‡               for every active node w in the v–rooted subtree of T,
                μ(w) := μ(w) + Δ_uv
```

The overall number of iterations of the loop in Step ⋆ is the number of connections added. The overall number of iterations of the loop in Step † is at most the number of edges, which is $O(|G|)$. Step ‡ can be done using a single bulk update in $O(\lg |G|)$ time. Consequently, the algorithm runs in time $O((|G| + \text{number of connections}) \lg |G|)$.

Now we show that the algorithm selects a covering set of connections (and that the set is small). At each moment in the execution of the algorithm, for each node v such that $\mu(v)$ is finite, let $\text{last}(v)$ denote the node p of P such that (p, v) was the most recently selected connection for v.

The μ *invariant* is: for every node v for which $\mu(v)$ is finite,

$$\mu(v) = \epsilon d_v - (\text{dist}(p_i, \text{last}(v)) + \text{dist}(\text{last}(v), v) - \text{dist}_T(p_i, v)) \tag{4}$$

Note that $\text{dist}(p_i, \text{last}(v)) + \text{dist}(\text{last}(v), v)$ is the length of the path that goes from the current root p_i to v via $\text{last}(v)$. When this length becomes significantly longer than $\text{dist}(p, v)$ (longer by ϵd_v), $\mu(v) \leq 0$ so the node v is included in the loop in Step $\star$, so the connection (p_i, v) is added. This shows that the connections added by FORWARD and BACKWARD cover each node $v \in B$.

To bound the number of connections, we follow Thorup in using the potential function $\Phi_v = \text{dist}(p_s, \text{last}(v)) + \text{dist}(\text{last}(v), v)$. Suppose that, at some execution of Step $\star$, $\mu(v) \leq 0$, so $\text{dist}(p_i, \text{last}(v)) + \text{dist}(\text{last}(v), v) - \text{dist}_T(p_i, v) \geq \epsilon d_v$. When a connection (p_i, v) is then added, $\text{last}(v)$ becomes p_i, so the potential function Φ_v is reduced by at least ϵd_v.

Initially $\Phi_v = \text{dist}(p_s, p_{i(v)}) + \text{dist}(p_{i(v)}, v)$. Throughout the phase, by the triangle inequality , $\Phi_v \geq \text{dist}(p_s, v)$. Again using the triangle inequality (and the fact that the graph is undirected), $\text{dist}(p_s, p_{i(v)}) \leq \text{dist}(p_s, v) + \text{dist}(p_{i(v)}, v)$, so $\Phi_v \geq \text{dist}(p_s, v) \geq \text{dist}(p_s, p_{i(v)}) - \text{dist}(p_s, v)$. Thus the total amount of reduction in Φ_v is at most $2\,\text{dist}(p_s, v)$. Since each reduction is by at least $\epsilon\,\text{dist}(r_{i(v)}, v)$, the total number of reductions (number of connections added by FORWARD after the initial one) is at most $\lceil 2\epsilon^{-1} \rceil$.

5 Approximate Distance Oracles for Genus g Graphs

Theorem 2. *For any undirected graph G embedded in a surface of Euler genus g, there exists a $(1+\epsilon)$–approximate distance oracle with query time $O(\epsilon^{-2}(\lg n + g)^2)$, linear space, and preprocessing time $O(n(\lg n)(g^3 + \lg n))$. The oracle can also be constructed in time $O(n(\lg n)(g/\epsilon + \lg n))$.*

Our distance oracle for genus g graphs is based on separating shortest paths, as for planar graphs (see Section 3.1). Thorup [Tho04] proves that any planar graph can be recursively separated by three shortest paths. In the following, we prove that genus g graphs can be recursively separated using at most $O(g)$ shortest paths. In fact, only the first separator consists of at most $2g$ paths, while lower levels can be separated using 3 paths. These smaller separators allow us to derive approximate oracles and labeling schemes with a dependency on g that is much lower than the corresponding dependency in the more general construction by Abraham and Gavoille [AG06]. More formally, we also prove the following.

Theorem 3 (fast distance queries for genus g graphs). *For any undirected graph G embedded in a surface of Euler genus g, there exists a $(1+\epsilon)$–approximate distance oracle with query time $O(g/\epsilon)$, space $O(n(g+\lg n)/\epsilon)$, and preprocessing time $O(n(\lg n)^3\epsilon^{-2} + n(\lg n)g/\epsilon)$. The oracle can be distributed as a labeling scheme using $O((g + \lg n)/\epsilon)$ bits per node.*

Overview. In the first step, we "cut" the genus g graph into planar subgraphs using the *tree-cotree decomposition* of Eppstein [Epp03], which decomposes a graph of genus g into planar graphs, separated by $2g$ paths from a tree T. We choose T to be a shortest-path tree. At a high level, the theorems follow by combining Eppstein's lemma [Epp03, Proof of Lemma 3.2] with the distance oracles for planar graphs (Thorup [Tho04] and Sections 3 and 4). Within the planar subgraphs, we use the distance oracles for planar graphs. In addition to computing the connections to the separator paths within each planar subgraph, we also need to compute the connections to the $O(g)$ tree-cotree decomposition paths. Note that the latter set of connections consists of paths that may pass through non-planar parts. To compute these, we may use either [CC07] or [Tho04, Lemma 3.12], depending on the values of g and ϵ.

Preprocessing Algorithm. To obtain the preprocessing and space bounds in Theorem 2, we use the preprocessing algorithm described in Section 4 with $r := \ell^2$, where $\ell = O(\epsilon^{-1}(\lg n + g))$. Since the number of connections per node is proportional to ℓ and since a $1/\sqrt{r}$–fraction of the nodes per subgraph lies on the boundary, the overall space consumption is linear. To obtain the preprocessing and space bounds in Theorem 3, we use Thorup's algorithm [Tho04, Thm. 3.19] for $O(1/\epsilon)$ query time.

There are two options to compute connections to the tree-cotree separator: (1) We may use [Tho04, Lemma 3.12] (which internally uses Thorup's $O(m)$ SSSP algorithm [Tho99, Tho00]). The lemma states that, for a path Q, we can compute an ϵ–covering set $\mathcal{C}(v, Q)$ for all nodes v in time $O(\epsilon^{-1} n(\lg n))$. (2) We may use the MSSP data structure for genus g graphs by Cabello and Chambers [CC07], which requires $O(g^2 n \lg n)$ preprocessing and then answers queries in time $O(\lg n)$. See planar preprocessing (Section 4) for details. The time required is $O(g^2 n \lg n + \text{number of connections} \cdot \lg n) = O(g^2 n \lg n)$. We apply either lemma for the at most $2g$ paths of the tree-cotree decomposition. (For Theorem 3, the first option gives faster asymptotic preprocessing time; for Theorem 2, the optimal choice depends on ϵ and g.)

Query Algorithm. At query time, we can essentially use the same algorithm as for the planar case (Section 3.2 and [Tho04, Thm. 3.19]). The only difference to the planar case is that we also need to include the at most $2g$ paths separating the genus graph into planar subgraphs. To obtain the bound on the query time in Theorem 2, note that computing connections through these $\leq 2g$ separating paths can be done in time $O(\ell g/\epsilon)$ and that exploring both regions took time $O(\ell^2)$ (where $\ell = O(\epsilon^{-1}(\lg n + g))$).

References

[ABH+04] Acar, U.A., Blelloch, G.E., Harper, R., Vittes, J.L., Woo, S.L.M.: Dynamizing static algorithms, with applications to dynamic trees and history independence. In: Proceedings of the Fifteenth ACM-SIAM Symposium on Discrete Algorithms, pp. 531–540 (2004)

[AG06] Abraham, I., Gavoille, C.: Object location using path separators. In: Proceedings of the Twenty-Fifth Annual ACM Symposium on Principles of Distributed Computing (PODC), pp. 188–197 (2006); Details in LaBRI Research Report RR-1394-06

[AHdLT05] Alstrup, S., Holm, J., de Lichtenberg, K., Thorup, M.: Maintaining information in fully dynamic trees with top trees. ACM Transactions on Algorithms 1(2), 243–264 (2005)

[BGK+10] Bartal, Y., Gottlieb, L.-A., Kopelowitz, T., Lewenstein, M., Roditty, L.: Fast, precise and dynamic distance queries. CoRR, abs/1008.1480 (2010) (to appear in SODA 2011)

[Cab06] Cabello, S.: Many distances in planar graphs. In: Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 1213–1220 (2006); a preprint of the journal version is available in the University of Ljubljana preprint series, vol. 47, p. 1089 (2009)

[CC07] Cabello, S., Chambers, E.W.: Multiple source shortest paths in a genus g graph. In: Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007, New Orleans, Louisiana, USA, pp. 89–97 (2007)

[CX00] Chen, D.Z., Xu, J.: Shortest path queries in planar graphs. In: Proceedings of the ACM Symposium on Theory of Computing (STOC), pp. 469–478 (2000)

[DPZ00] Djidjev, H., Pantziou, G.E., Zaroliagis, C.D.: Improved algorithms for dynamic shortest paths. Algorithmica 28(4), 367–389 (2000)

[Epp03] Eppstein, D.: Dynamic generators of topologically embedded graphs. In: Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 599–608 (2003)

[FR06] Fakcharoenphol, J., Rao, S.: Planar graphs, negative weight edges, shortest paths, and near linear time. Journal of Computer and System Sciences 72(5), 868–889 (2006), announced at FOCS 2001

[Fre87] Frederickson, G.N.: Fast algorithms for shortest paths in planar graphs, with applications. SIAM Journal on Computing 16(6), 1004–1022 (1987)

[Fre97] Frederickson, G.N.: A data structure for dynamically maintaining rooted trees. Journal of Algorithms 24, 37–65 (1997), announced at SODA 1993

[HKRS97] Henzinger, M.R., Klein, P.N., Rao, S., Subramanian, S.: Faster shortest-path algorithms for planar graphs. Journal of Computer and System Sciences 55(1), 3–23 (1997), announced at STOC 1994

[HPM06] Har-Peled, S., Mendel, M.: Fast construction of nets in low dimensional metrics, and their applications. SIAM J. Comput. 35(5), 1148–1184 (2006), announced at SOCG 2005

[KK06] Kowalik, L., Kurowski, M.: Oracles for bounded-length shortest paths in planar graphs. ACM Transactions on Algorithms 2(3), 335–363 (2006), announced at STOC 2003

[KKS11] Kawarabayashi, K., Klein, P.N., Sommer, C.: Linear-space approximate distance oracles for planar, bounded-genus, and minor-free graphs. CoRR, abs/1104.5214 (2011)

[Kle05] Klein, P.N.: Multiple-source shortest paths in planar graphs. In: Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 146–155 (2005)

[KMW10] Klein, P.N., Mozes, S., Weimann, O.: Shortest paths in directed planar graphs with negative lengths: A linear-space O(nlog^{2}n)-time algorithm. ACM Transactions on Algorithms 6(2) (2010), announced at SODA 2009

[LT79] Lipton, R.J., Tarjan, R.E.: A separator theorem for planar graphs. SIAM Journal on Applied Mathematics 36(2), 177–189 (1979)

[MS10] Mozes, S., Sommer, C.: Exact distance oracles for planar graphs. CoRR, abs/1011.5549 (2010)

[MWN10] Mozes, S., Wulff-Nilsen, C.: Shortest paths in planar graphs with real lengths in $O(n \log^2 n/ \log \log n)$ time. In: de Berg, M., Meyer, U. (eds.) ESA 2010. LNCS, vol. 6347, pp. 206–217. Springer, Heidelberg (2010)

[MZ07] Muller, L.F., Zachariasen, M.: Fast and compact oracles for approximate distances in planar graphs. In: Proceedings of the 15th Annual European Conference on Algorithms, pp. 657–668 (2007)

[Nus10] Nussbaum, Y.: Improved distance queries in planar graphs. CoRR, abs/1012.2825 (2010)

[PR10] Patrascu, M., Roditty, L.: Distance oracles beyond the Thorup–Zwick bound. In: 51st Annual IEEE Symposium on Foundations of Computer Science, FOCS (2010)

[Sli07] Slivkins, A.: Distance estimation and object location via rings of neighbors. Distributed Computing 19(4), 313–333 (2007), announced at PODC 2005

[ST83] Sleator, D.D., Tarjan, R.E.: A data structure for dynamic trees. Journal of Computer and System Sciences 26(3), 362–391 (1983), announced at STOC 1981

[SVY09] Sommer, C., Verbin, E., Yu, W.: Distance oracles for sparse graphs. In: 50th Annual IEEE Symposium on Foundations of Computer Science (FOCS), pp. 703–712 (2009)

[Tal04] Talwar, K.: Bypassing the embedding: algorithms for low dimensional metrics. In: Proceedings of the 36th Annual ACM Symposium on Theory of Computing (STOC), pp. 281–290 (2004)

[Tho99] Thorup, M.: Undirected single-source shortest paths with positive integer weights in linear time. Journal of the ACM 46(3), 362–394 (1999), announced at FOCS 1997

[Tho00] Thorup, M.: Floats, integers, and single source shortest paths. Journal of Algorithms 35(2), 189–201 (2000), announced at STACS 1998

[Tho04] Thorup, M.: Compact oracles for reachability and approximate distances in planar digraphs. Journal of the ACM 51(6), 993–1024 (2004), announced at FOCS 2001

[TW05] Tarjan, R.E., Werneck, R.F.F.: Self-adjusting top trees. In: Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 813–822 (2005)

[TZ05] Thorup, M., Zwick, U.: Approximate distance oracles. Journal of the ACM 52(1), 1–24 (2005), announced at STOC 2001

Stochastic Mean Payoff Games: Smoothed Analysis and Approximation Schemes*

Endre Boros[1], Khaled Elbassioni[2], Mahmoud Fouz[3], Vladimir Gurvich[1], Kazuhisa Makino[4], and Bodo Manthey[5]

[1] RUTCOR, Rutgers University
[2] Max-Planck-Institut für Informatik, Saarbrücken, Germany
[3] Universität des Saarlandes, Fachrichtung Informatik, Germany
[4] University of Tokyo, Graduate School of Information Science and Technology
[5] University of Twente, Department of Applied Mathematics

Abstract. In this paper, we consider two-player zero-sum stochastic mean payoff games with perfect information modeled by a digraph with black, white, and random vertices. These *BWR-games* games are polynomially equivalent with the classical Gillette games, which include many well-known subclasses, such as cyclic games, simple stochastic games, stochastic parity games, and Markov decision processes. They can also be used to model parlor games such as Chess or Backgammon.

It is a long-standing open question if a polynomial algorithm exists that solves BWR-games. In fact, a pseudo-polynomial algorithm for these games with an arbitrary number of random nodes would already imply their polynomial solvability. Currently, only two classes are known to have such a pseudo-polynomial algorithm: BW-games (the case with *no* random nodes) and *ergodic* BWR-games (in which the game's value does not depend on the initial position) with constant number of random nodes. In this paper, we show that the existence of a pseudo-polynomial algorithm for BWR-games with constant number of random vertices implies smoothed polynomial complexity and the existence of absolute and relative polynomial-time approximation schemes. In particular, we obtain smoothed polynomial complexity and derive absolute and relative approximation schemes for BW-games and ergodic BWR-games (assuming a technical requirement about the probabilities at the random nodes).

1 Introduction

The rise of the Internet has led to an explosion in research in game theory: the mathematical modeling of competing agents in strategic situations. The central concept in such models is that of a *Nash equilibrium*, defining a state where

* The first author is grateful for the partial support of the National Science Foundation (CMMI-0856663, "Discrete Moment Problems and Applications"), and the first, second, fourth and fifth authors are thankful to the Mathematisches Forschungsinstitut Oberwolfach for providing a stimulating research environment with an RIP award in March 2010.

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 147–158, 2011.

no agent gains an advantage by changing her current strategy; it serves as a prediction for the outcome of strategic situations in which selfish agents compete.

A fundamental result in game theory shows that if the agents can choose a *mixed strategy* (i.e., probability distributions of deterministic strategies), a Nash equilibrium is guaranteed to exist in finite games. Often, however, already *pure* (i.e., deterministic) strategies already lead to a Nash equilibrium. Still, the existence of Nash equilibria might be irrelevant in practice, since their computation would take too long. Thus, algorithmic aspects of game theory have gained a lot of interest. Following the dogma that only polynomial time algorithms are feasible in practice, it is desirable to show polynomial time complexity for the computation of Nash equilibria. On the other hand, in cases where such an efficient algorithm is not known to exist an approximate notion of Nash equilibria has been suggested, in which no agent can gain a substantial advantage by changing her current strategy. In this paper, we advocate another notion of tractability by considering the *smoothed complexity* of a well-known two-player stochastic game for which the existence of a polynomial algorithm is a long-standing open question. In contrast to the usual worst-case complexity, smoothed complexity analyzes the running time of algorithms on typical instances. By establishing smoothed polynomial complexity, we argue that the computation of a Nash equilibrium is feasible in all, but artificially constructed worst-case instances.

The model that we consider is *mean stochastic payoff games* or *BWR-games*: we are given a directed graph $G = (V, E)$ whose vertex set V is partitioned into three subsets $V = V_B \cup V_W \cup V_R$ that correspond to black, white, and random *positions*, respectively. The arcs stand for *moves*. The black and white vertices are owned by two players: Black – the *minimizer* – owns the black vertices in V_B, and White – the *maximizer* – owns the white vertices in V_W. The vertices in V_R are owned by nature. We have a local reward $r_e \in \mathbb{R}$ for each arc $e \in E$ and a probability p_{vu} for each arc (v, u) going out of $v \in V_R$. Starting from some vertex $v_0 \in V$, a token is moved along one arc e in every round of the game. If the token is on a black vertex, Black selects an outgoing arc e and moves the token along e. If the token is on a white vertex, White selects an outgoing arc e. In a random position $v \in V_R$, a move $e = (v, u)$ is chosen according to the probabilities p_{vu} of the outgoing arcs of v. In all cases, Black pays White the reward r_e on the selected arc e. A strategy of a player is a mapping that assign a move $(u, v) \in E$ to each position u he owns.

Starting from a given initial position $v_0 \in V$, the game produces for a pair of fixed strategies (i.e., one for each player) an infinite walk $\{v_0, v_1, v_2, \ldots\}$ (called a *play*). Let b_i denote the reward $r_{v_i v_{i+1}}$ received by White in step $i \in \{0, 1, \ldots\}$. The *undiscounted limit average effective payoff* is defined as the *Cesàro average* $c = \liminf_{n\to\infty} \frac{\sum_{i=0}^{n} \mathbb{E}[b_i]}{n+1}$. White's objective is to maximize c, while Black's objective is to minimize c. Every such game is known to have a pair of *uniformly optimal strategies* that result in a Nash equilibrium (called a *saddle point*) from *any* initial position [4,8]. All optimal pairs of strategies, from a given initial position v result in a unique payoff $\mu(v)$, called the value of the game at v. An algorithm is said to solve the game if it computes an optimal pair of strategies.

BWR-games are an equivalent formulation [5] of the stochastic games with perfect information and mean payoff that were introduced in 1957 by Gillette [4]. They generalize many important problems. The special case of BWR-games without random vertices ($V_R = \emptyset$) is known as *cyclic* or *mean payoff* games (see, e.g., [5]); we call these *BW-games*. If one of the sets V_B or V_W is empty, we obtain a *Markov decision process* for which polynomial-time algorithms are known [9]. If both are empty ($V_B = V_W = \emptyset$), we get a *weighted Markov chain*.

Besides their many applications, all these games are of interest to complexity theory: Karzanov and Lebedev [7] proved that the decision problem "whether the value of a BW-game is positive" is in the intersection of NP and co-NP. Yet, no polynomial algorithm is known even in this special case, see, e.g., the recent survey by Vorobyov [14]. A similar complexity claim can be shown to hold for BWR-games. On the other hand, there exist algorithms (see, e.g., [5]) that solve BW-games in practice very fast. The situation for these games is thus comparable to linear programming before the seminal discovery of the ellipsoid method, where the problem was also known to lie in the intersection of NP and co-NP and where the simplex algorithm proved to be a fast algorithm in practice. Spielman and Teng [12,13] introduced smoothed analysis to explain the practical performance of the simplex method. We further enforce this analogy by showing a smoothed polynomial complexity for a large class of BWR-games.

While there are numerous pseudo-polynomial algorithms known for the BW-case [5,15], pseudo-polynomiality for BWR-games (with no restriction on the number of random nodes) is in fact equivalent to polynomiality [1]. Recently, a pseudo-polynomial algorithm was given in [3] for BWR-games with a *constant* number of random vertices and polynomial common denominator of transition probabilities, but under the assumption that the game is *ergodic*, i.e., the game value does not depend on the initial position. However, the existence of a similar algorithm for the non-ergodic or non-constant number of random vertices remains open, as the approach in [3] does not seem to generalize to these cases.

1.1 Our Results and Some Related Work

Approximation Schemes. The only result that we are aware of regarding approximation schemes is the observation made by Roth et al. [11] that the values of BW-games can be approximated within an *absolute* error of ε in polynomial-time, if all rewards are in the range $[-1, 1]$. This follows immediately from truncating the rewards and using any of the known pseudo-polynomial algorithms.

In this paper, we generalize this result to BWR-games in two directions. Throughout the paper, we write $\mathbb{G}$ for any class of digraphs $G = (V_B \cup V_W \cup V_R, E)$ that admit a pseudo-polynomial algorithm $\mathbb{A}$, i.e., $\mathbb{A}$ solves any BWR-game $\mathcal{G}$ on $G \in \mathbb{G}$, with integral rewards and rational transition probabilities, in time polynomial in n, D, and R, where $n = n(\mathcal{G})$ is the total number of vertices, $R = R(\mathcal{G})$ is the size of the range of the rewards, and $D = D(\mathcal{G})$ is the common denominator of the transition probabilities. E.g., digraphs without random vertices are known to belong to $\mathbb{G}$. The same holds for digraphs that have a constant number of random nodes and are *structurally ergodic* [6], i.e., for *any*

set of rewards, all positions have the same game value. Note that the dependence on D is inherent in all known pseudo-polynomial algorithms for BWR-games.

Let $p_{\min} = p_{\min}(\mathcal{G})$ be the minimum positive transition probability in the game $\mathcal{G}$. Throughout the paper, we will assume that $k := |V_R|$ is constant.

Theorem 1. *For any $\varepsilon > 0$, there exists for each of the following two cases an algorithm that returns a pair of strategies that approximates the value of any BWR-game on $G \in \mathbb{G}$ from any starting position:*

i. With rewards in the interval $[-1, 1]$, within an absolute *error of ε, in time* $\mathrm{poly}(n, \frac{1}{p_{\min}}, \frac{1}{\varepsilon})$.

ii. With non-negative integral rewards, within a relative *error of ε, in time* $\mathrm{poly}(n, \log R, \frac{1}{p_{\min}}, \frac{1}{\varepsilon})$.

Our reduction in case (i), unlike case (ii), has the property that if the pseudo-polynomial algorithm returns *uniformly* optimal strategies, i.e., that are *independent* of the starting position, then so does the approximation scheme (in an approximate sense). With some more work, we can show that the same is also true in case (ii) of Theorem 1 for BW-games.

In deriving these approximation schemes from a pseudo-polynomial algorithm as defined above, we face two main technical challenges that distinguish the computation of approximate equilibria of BWR-games from similar standard techniques used in optimization: (i) the running time of the pseudo-polynomial algorithm depends polynomially both on the maximum reward and the common denominator D of the transition probabilities; thus to obtain a *fully polynomial-time approximation scheme* (FPTAS) with an absolute guarantee whose running time is independent of D, we need to truncate the probabilities and bound the change in the game value, which is a *non-linear* function of D, (ii) to obtain an FPTAS with a relative guarantee, one usually exploits a (trivial) lower/upper bound on the optimum value; this is not possible in the case of BWR-games, since the game value can be arbitrarily small; the situation becomes even more complicated, if we look for uniformly ε-optimal strategies, since we have to output one pair of strategies which guarantees ε-optimality from any starting position. In order to solve the first issue, we analyze the change in the game values and optimal strategies if the rewards or transition probabilities are changed. The second issue is solved through repeated applications of the pseudo-polynomial algorithm on a truncated game; after each such application we show that either the value of the game has already been approximated within the required accuracy, or the range of the rewards can be shrunk by a constant factor without changing the value of the game (Sections 3.2 and 3.3).Since BW-games and structurally ergodic BWR-games with constant k admit pseudo-polynomial algorithms, we obtain the following results.

Corollary 1. *There is an FPTAS that solves,*

i. within a relative error, in uniformly ε-optimal strategies, any BW-game with non-negative (rational) rewards;

ii. within an absolute error, in uniformly ε-optimal strategies, any structurally ergodic BWR-game with rewards in $[-1, 1]$ and $\frac{1}{p_{\min}} = \text{poly}(n)$;
iii. within a relative error, in uniformly ε-optimal strategies, any structurally ergodic BWR-game with non-negative rational rewards and $\frac{1}{p_{\min}} = \text{poly}(n)$.

Note that (i) strengthens the absolute FPTAS for BW-games [11], and (ii) and (iii) enlarge the class of games for which an FPTAS exists.

Smoothed Analysis for BWR-games. We further show that typical instances of digraphs that admit a pseudo-polynomial algorithm can be solved in polynomial time. Towards this end, we do a smoothed analysis using the one-step model introduced by Beier and Vöcking [2]: an adversary specifies a BWR-game $\mathcal{G}$ and for each arc a density function. These functions are bounded from above by a parameter ϕ. Then the rewards for all arcs are drawn independently according to their respective density functions. We prove that in this setting, independent of the actual choices of the adversary, the resulting game can be solved in polynomial time with high probability; there exists a polynomial $P(n, \phi, 1/\varepsilon)$ such that the probability that the algorithm exceeds a running-time of $P(n, \phi, 1/\varepsilon)$ is at most ε. This shows that such BWR-games with a constant number of random vertices have smoothed polynomial complexity.

Theorem 2. *There is an algorithm solving any BWR-game on any $G \in \mathbb{G}$ with rational transition probabilities and $D = \text{poly}(n)$ in smoothed polynomial time.*

Theorem 2 is similar to the result by Beier and Vöcking [2] who showed that a binary optimization problem defined by linear constraints and a linear objective function has smoothed polynomial complexity if it admits a pseudo-polynomial algorithm. Our proof of Theorem 2 has a similar structure like their analysis. However, in the case of BWR-games, the situation becomes more complicated: First, we have to deal with two conflicting objectives (of the two players). Second, the coefficients of the objective functions are not given explicitly. In consequence, our proof requires a novel isolation lemma that deals with two players who optimize the same objective function in two different directions. Furthermore, our procedure for certifying that the solution found is indeed the optimal solution is considerably more involved and requires careful rounding of the coefficients in order to certify optimality.

Corollary 2. *(i) BW-games and (ii) structurally ergodic BWR-games with $D = \text{poly}(n)$ can be solved in smoothed polynomial time.*

Let us remark finally that removing the assumption that k is constant in the above results remains a challenging open problem.

2 Preliminaries, Notation and Basic Properties

BWR-games and Markov Chains. A BWR-game is defined by a triple $\mathcal{G} = (G, P, r)$, where $G = (V = V_W \cup V_B \cup V_R, E)$ is a digraph that may have loops and

multiple arcs, but no terminal vertices, i.e., vertices of out-degree 0; $P \in [0,1]^E$ is the vector of probability distributions for all $v \in V_R$ specifying the probability p_{vu} of a move from v to u; $r \in \mathbb{R}^E$ is a local reward function. It is assumed that $\sum_{u:(v,u)\in E} p_{vu} = 1$ for all $v \in V_R$ and $p_{v,u} > 0$ whenever $(v,u) \in E$ and $v \in V_R$.

Standardly, we define a strategy $s_W \in S_W$ for WHITE as a mapping that assigns a move $(v,u) \in E$ to each position $v \in V_W$. For simplicity, we may write $s_W(v) = u$ for $s_W(v) = (v,u)$. Strategies $s_B \in S_B$ for BLACK are analogously defined. A pair of strategies $s = (s_W, s_B)$ is called a *situation*. Given a BWR-game $\mathcal{G} = (G, P, r)$ and a situation $s = (s_B, s_W)$, we obtain a weighted Markov chain $\mathcal{G}(s) = (G(s) = (V, E(s)), P(s), r)$ with transition matrix $P(s)$ defined by:

$$p_{vu}(s) = \begin{cases} 1 & \text{if } (v \in V_W \text{ and } u = s_W(v)) \text{ or } (v \in V_B \text{ and } u = s_B(v)); \\ 0 & \text{if } (v \in V_W \text{ and } u \neq s_W(v)) \text{ or } (v \in V_B \text{ and } u \neq s_B(v)); \\ p_{vu} & \text{if } v \in V_R. \end{cases}$$

Here, $E(s) = \{e \in E \mid p_e(s) > 0\}$ is the set of arcs with positive probability. Given an initial position $v_0 \in V$ from which the play starts, we define the limiting (mean) effective payoff $c_{v_0}(s)$ in $\mathcal{G}(s)$ as $c_{v_0}(s) = \rho(s)^T r = \sum_{e\in E} \rho_e(s) r_e$, where $\rho(s) = \rho(s, v_0) \in [0,1]^E$ is the arc-limiting distribution for $\mathcal{G}(s)$ starting from v_0. This means that for $(v,u) \in E$, $\rho_{vu}(s) = \pi_v(s) p_{vu}(s)$, where $\pi \in [0,1]^V$ is the limiting distribution in the Markov chain $\mathcal{G}(s)$ starting from v_0. In what follows, we use $(\mathcal{G}, v_0)$ to denote the game starting from v_0. We write $\rho(s)$ for $\rho(s, v_0)$, when v_0 is clear from the context. For rewards $r : E \to \mathbb{R}$, let $r^- = \min_e r_e$ and $r^+ = \max_e r_e$. Let $[r] = [r^-, r^+]$ be the range of r. Let $R = R(\mathcal{G}) = r^+ - r^-$.

Strategies and Saddle Points. If we consider $c_{v_0}(s)$ for all possible situations, we obtain a matrix game $C_{v_0} : S_W \times S_B \to \mathbb{R}$, with entries $C_{v_0}(s_W, s_B) = c_{v_0}(s_W, s_B)$. Every such game has a Nash equilibrium in pure strategies [4,8]; a corresponding pair of strategies is said to be *optimal*. Moreover, there exists optimal strategies (s_W^*, s_B^*) that do not depend on the starting position v_0; such strategies are called *uniformly optimal*. Although there might be several optimal strategies, it is easy to see that they all lead to the same value. We define this to be the value of the game and write $\mu_{v_0}(\mathcal{G}) := C_{v_0}(s_W^*, s_B^*)$ where (s_W^*, s_B^*) is any pair of optimal strategies. Note that $\mu_{v_0}(\mathcal{G})$ may depend on the starting node v_0.

3 Approximation Schemes

Given a BWR-game $\mathcal{G} = (G = (V,E), P, r)$, a constant $\varepsilon > 0$, and a starting position $v \in V$, an *ε-relative approximation* of the value of the game is determined by a situation (s_W^*, s_B^*) such that

$$\max_{s_W} \mu_v(\mathcal{G}(s_W, s_B^*)) \leq (1+\varepsilon)\mu_v(\mathcal{G}) \quad \text{and} \quad \min_{s_B} \mu_v(\mathcal{G}(s_W^*, s_B)) \geq (1-\varepsilon)\mu_v(\mathcal{G}). \quad (1)$$

An alternative to relative approximations is to look for an approximation with *absolute* error of ε. This is achieved by a situation (s_W^*, s_B^*) such that

$$\max_{s_W} \mu_v(\mathcal{G}(s_W, s_B^*)) \leq \mu_v(\mathcal{G}) + \varepsilon \quad \text{and} \quad \min_{s_B} \mu_v(\mathcal{G}(s_W^*, s_B)) \geq \mu_v(\mathcal{G}) - \varepsilon. \quad (2)$$

A situation (s^*_W, s^*_B) satisfying (1) (resp., (2)) is called relative (resp., absolute) ε-optimal. If the pair (s^*_W, s^*_B) is ε-optimal for any starting position, it is called *uniformly* ε-optimal.

3.1 Absolute Approximation

Let $G = (V, E)$ be a graph in $\mathbb{G}$ and $\mathcal{G} = (G, P, r)$ be a BWR-game on G. In this section, we assume that $r^- = -1$ and $r^+ = 1$, i.e., all rewards are from the interval $[-1, 1]$. We apply the pseudo-polynomial algorithm $\mathbb{A}$ on a truncated game $\tilde{\mathcal{G}} = (G = (V, E), \tilde{P}, \tilde{r})$ defined by rounding the rewards to the nearest integer multiple of $\varepsilon/4$ (denoted $\tilde{r} := \lfloor r \rceil_{\frac{\varepsilon}{4}}$), and truncating the vector of probabilities $(p_{vu} : u \in V)$ for each random node $v \in V_R$ as follows.

Lemma 1. *Let $\alpha \in [0,1]^n$ with $\|\alpha\|_1 = 1$. Let $B \in \mathbb{Z}^+$ be an integer such that $\min_{i:\alpha_i>0}\{\alpha_i\} > 2^{-B}$. Then there exists $\alpha' \in [0,1]^n$ such that (i) $\|\alpha'\|_1 = 1$; (ii) for all $i = 1, \ldots, n$, $\alpha'_i = c_i/2^B$ where $c_i \in \mathbb{Z}^+$ is an integer; (iii) for all $i = 1, \ldots, n$, $\alpha'_i > 0$ if and only $\alpha_i > 0$, and (iv) $\|\alpha - \alpha'\|_\infty \le 2^{-B}$.*

Lemma 2. *Let $\mathbb{A}$ be a pseudo-polynomial algorithm that solves, in (uniformly) optimal strategies, any BWR-game $\mathcal{G} = (G, P, r)$ with $G \in \mathbb{G}$ in time $\tau(n, D, R)$. Then for any $\varepsilon > 0$, there is an algorithm that solves, in (uniformly) absolute ε-optimal strategies, any BWR-game $\mathcal{G} = (G, P, r)$ with $G \in \mathbb{G}$ in time bounded by $\tau(n, \frac{2^{2k+5}n^3k^2}{\varepsilon p_{\min}^{2k}}, \frac{4}{\varepsilon})$, where $p_{\min} = p_{\min}(\mathcal{G})$.*

3.2 Relative Approximation

Let $G = (V, E)$ be a graph in $\mathbb{G}$ and $\mathcal{G} = (G, P, r)$ be a BWR-game on G with non-negative rational rewards (i.e., $r^- = 0$). Without loss of generality, we may assume that the rewards are integral with $\min_{e:r_e>0} r_e = 1$. The algorithm is given as Algorithm 1. The main idea is to truncate the rewards, scaled by a certain factor $1/K$, and use the pseudo-polynomial algorithm on the truncated game $\hat{\mathcal{G}}$. If the value in the truncated game $\mu_w(\hat{\mathcal{G}})$, from the starting node w, is large enough (step 5) then we get a good relative approximation of the original value and we are done. Otherwise, the information that $\mu_w(\hat{\mathcal{G}})$ is small allows us to reduce the maximum reward by a factor of 2 in the original game (step 8). Thus the algorithm terminates in polynomial time (in the bit length of $R(\mathcal{G})$). To remove the dependence on D in the running time, we need also to truncate the transition probabilities. In the algorithm, we denote by $\tilde{P}$ the transition probabilities obtained from P by applying Lemma 1 with $B = \lceil \log 1/\varepsilon' \rceil$, where we select $\varepsilon' = \frac{p_{\min}^{2k}}{2^{2k+3}n^3k^2\theta}$, where $\theta = \theta(\mathcal{G}) := \frac{2(1+\varepsilon)(3+2\varepsilon)n}{\varepsilon p_{\min}^{2k+1}}$, so that $2\delta(\mathcal{G}, \varepsilon') \le \frac{r_+(\mathcal{G})}{\theta(\mathcal{G})} := K(\mathcal{G})$, where $\delta(\mathcal{G}, \varepsilon) := \Big(\frac{\varepsilon}{2}n^2\big(\frac{1}{2}p_{\min}\big)^{-k}\big[\varepsilon nk(k+1)\big(\frac{1}{2}p_{\min}\big)^{-k} + 3k + 1\big] + \varepsilon n\Big)r_*$ with $r_* = r_*(\mathcal{G}) := \max\{|r_+(\mathcal{G})|, |r_-(\mathcal{G})|\}$.

Lemma 3. *Let $\mathbb{A}$ be a pseudo-polynomial algorithm that solves any BWR-game $\mathcal{G} = (G, P, r)$ with $G \in \mathbb{G}$ in time $\tau(n, D, R)$. Then for any $\varepsilon \in (0, 1)$, there*

Algorithm 1. FPTAS-BWR$(\mathcal{G}, w, \varepsilon)$

Input: a BWR-game $\mathcal{G} = (G = (V, E), P, r)$, a starting vertex $w \in V$, an accuracy ε.
Output: an ε-optimal pair $(\tilde{s}_W, \tilde{s}_B)$ for the game $(\mathcal{G}, w)$.

if $r^+(\mathcal{G}) = 1$ **then**
  $\hat{\mathcal{G}} := (G, \tilde{P}, r)$; **return** $\mathbb{A}(\hat{\mathcal{G}}, v)$
$K := \frac{r^+(\mathcal{G})}{\theta(\mathcal{G})}$; $\hat{r}_e = \lfloor \frac{r_e}{K} \rfloor$ for $e \in E$; $\hat{\mathcal{G}} = (G, \tilde{P}, \hat{r})$
$(\tilde{s}_W, \tilde{s}_B) := \mathbb{A}(\hat{\mathcal{G}}, w)$
if $\mu_w(\hat{\mathcal{G}}) \geq \frac{3}{\varepsilon}$ **then**
  return $(\tilde{s}_W, \tilde{s}_B)$
else
  for all $e \in E$, let $\tilde{r}_e = \begin{cases} \lceil \frac{r^+}{2} \rceil & \text{if } r_e > \frac{r^+}{2(1+\varepsilon)} \\ r_e & \text{otherwise} \end{cases}$
  $\tilde{\mathcal{G}} := (G, P, \tilde{r})$; **return** FPTAS-BWR$(\tilde{\mathcal{G}}, w, \varepsilon)$

Algorithm 2. FPTAS-BW$(\mathcal{G}, \varepsilon)$

Input: a BW-game $\mathcal{G} = (G = (V = V_B \cup V_W, E), r)$, and accuracy ε.
Output: a uniformly ε-optimal pair $(\tilde{s}_W, \tilde{s}_B)$ for $\mathcal{G}$.

if $r^+(\mathcal{G}) = 1$ **then**
  return $\mathbb{A}(\mathcal{G})$
$K := \frac{\varepsilon' r^+}{2(1+\varepsilon')^2 n}$; $\hat{r}_e = \lfloor \frac{r_e}{K} \rfloor$ for $e \in E$; $\hat{\mathcal{G}} = (G, \hat{r})$
$(\hat{s}_W, \hat{s}_B) := \mathbb{A}(\hat{\mathcal{G}})$; $U := \{u \in V \mid \mu_u(\hat{\mathcal{G}}) \geq \frac{1}{\varepsilon'}\}$
if $U = V$ **then**
  return $(\tilde{s}_W, \tilde{s}_B) = (\hat{s}_W, \hat{s}_B)$
else
  $\tilde{G} := G[V \setminus U]$
  for all $e \in E(\tilde{G})$, let $\tilde{r}_e = \begin{cases} \lceil \frac{r^+}{2} \rceil & \text{if } r_e > \frac{r^+}{2(1+\varepsilon')} \\ r_e & \text{otherwise} \end{cases}$
  $\tilde{\mathcal{G}} := (\tilde{G}, \tilde{r})$
  $(\tilde{s}_W, \tilde{s}_B) :=$FPTAS-BW$(\tilde{\mathcal{G}}, \varepsilon)$
  $\tilde{s}(w) := \hat{s}(w)$ for all $w \in U$; $\tilde{s} = (\tilde{s}_W, \tilde{s}_B)$

is an algorithm that solves, in relative ε-optimal strategies, any BWR-game $(\mathcal{G} = (G, P, r), w)$ with $G \in \mathbb{G}$, from any given starting position w, in time $\left(\tau\left(n, \frac{4^{k+2} n^4 k^2 (1+\varepsilon)(3+2\varepsilon)}{\varepsilon p_{\min}^{2k}}, \frac{2(1+\varepsilon)(3+2\varepsilon)n}{\varepsilon p_{\min}^{2k+1}}\right) + \text{poly}(n)\right)(\lfloor \log R \rfloor + 1)$.

Remark 1. It is easy to see that, for structurally ergodic BWR-games, one can modify the above procedure to return uniformly ε-optimal strategies.

3.3 Uniformly Relative ε-Approximation for BW-Games

Note that the FPTAS in Lemma 3 does not necessarily return a uniformly ε-optimal situation, even if the pseudo-polynomial algorithm $\mathbb{A}$ provides a uniformly optimal situation. In case of BW-games, we can modify this FPTAS to

return a situation which is ε-optimal for all $v \in V$. The algorithm is given as Algorithm 2. The main difference is that when we recurse on a game with reduced rewards (step 11), we have also to delete all nodes that have large values $\mu(\tilde{\mathcal{G}}, v)$ in the truncated game. This is similar to the approach used to decompose a BW-game into ergodic classes [5]. However, the main technical difficulty is that, with approximate equilibria, WHITE (resp., BLACK) might still have some incentive to move from a higher-value (resp., lower-value) class to a lower-value (resp., higher-value) class, since the values are just estimated approximately. We show that such a move will not be very profitable for WHITE (resp., BLACK). As before, we assume that the rewards are integral with $\min_{e:r_e>0} r_e = 1$.

Lemma 4. *Let $\mathbb{A}$ be a pseudo-polynomial algorithm that solves, in uniformly optimal strategies, any BWR-game $\mathcal{G} = (G, P, r)$ with $G \in \mathbb{G}$ in time $\tau(n, R)$. Then for any $\varepsilon > 0$, there is an algorithm that solves, in uniformly relative ε-optimal strategies, any BW-game $\mathcal{G} = (G, P, r)$ with $G \in \mathbb{G}$, in time $(\tau(n, \frac{2(1+\varepsilon')^2 n}{\varepsilon'}) + \text{poly}(n))h$, where $h = \lfloor \log R \rfloor + 1$, and $\varepsilon' = \frac{\ln(1+\varepsilon)}{4h-2}$.*

4 Smoothed Analysis

We use the following notion of polynomial smoothed complexity introduced by Beier and Vöcking [2]. A problem is said to have smoothed polynomial complexity if and only if there exists an algorithm $\mathcal{A}$ with running-time T and a constant α such that

$$\forall \phi \geq 1, \forall n \in \mathbb{N} : \max_{\boldsymbol{f} \in D_n(\phi)} \mathbb{E}_{X \sim \boldsymbol{f}}(T(X)^\alpha) = O(n\phi) \,. \tag{3}$$

Here, $D_n(\phi)$ denotes all possible vectors of density functions bounded by ϕ for instances of size n, and X is an instance drawn according to $\boldsymbol{f}$. Equivalently, there exists a polynomial $P(n, \phi, 1/\varepsilon)$ such that with probability at most ε, $\mathcal{A}$ exceeds a running-time of $P(n, \phi, 1/\varepsilon)$.

Let $\mathbb{A}$ be a pseudo-polynomial algorithm that solves any BWR-game $\mathcal{G} = (G, P, r)$ with $G = (V, E) \in \mathbb{G}$. In this section, we show that any such game can be solved in smoothed polynomial time. For this, we assume that an adversary specifies a game together with density functions for the rewards (one for each arc), and these density functions are bounded by ϕ, and show a bound as in (3). One (technical) issue is that the perturbed rewards are of course real, non-rational numbers with probability 1. Thus, we cannot really use existing algorithms as sub-routine, and we cannot even compute anything with these numbers on an ordinary RAM. To cope with this problem, we use Beier and Vöcking's [2] approach and assume that the rewards are in $[-1, 1]$ and that we can access the bits of the rewards one-by-one.

To state our results in a bit more general setting, we will assume that $\mathbb{A}$ solves any BWR-game $\mathcal{G}$ in uniformly optimal strategies. If this was not the case, then it is easy to modify the procedure and analysis in this section to solve the game starting from a given vertex.

Algorithm 3. Solve($\mathcal{G}$)

Input: a BWR-game $\mathcal{G} = (G = (V, E), P, r)$.
Output: an optimal pair $(\tilde{s}_W, \tilde{s}_B)$ for the game $\mathcal{G}$.
1: $\ell_0 \leftarrow \log((nD)^{c_0}\phi)$; $i \leftarrow 0$ {c_0 is a constant to be specified later}
2: **repeat**
3: $\quad \ell := \ell_0 + i$; $\varepsilon \leftarrow 2^{-\ell}$; $i := i + 1$
4: $\quad \tilde{r} := \lfloor r \rfloor_\ell$; $\tilde{\mathcal{G}} := (G, P, \tilde{r})$; $\tilde{\mathcal{G}}' := (G, P, 2^\ell \tilde{r})$
5: $\quad (\tilde{s}_W, \tilde{s}_B) := \mathbb{A}(\tilde{\mathcal{G}}')$
6: **until** $\tilde{s}$ is optimal in $\tilde{\mathcal{G}}_{e,\varepsilon}$ for all $e \in E$

Before describing the procedure (Algorithm 3), we need to introduce some notation. Let us write $\lfloor x \rfloor_b$ for the largest integer smaller than or equal to x that has b bits (i.e., we basically cut off all bits after the b-th bit). Let $\gamma = \gamma(\mathcal{G}) := (kn)^{-2}(2D)^{-2(k+2)}$ and $\varepsilon > 0$. Given the game $\mathcal{G} = (G = (V, E), P, r)$, define, for each $e \in E$, the game $\mathcal{G}_{e,\varepsilon} = (G, P, r(e))$, where

$$r_{e'}(e) = \begin{cases} r_e + 2\gamma^{-1}\varepsilon & \text{if } e' = e, \\ r_{e'} & \text{otherwise.} \end{cases} \tag{4}$$

The basic idea behind our smoothed analysis is as follows: We use a certain number of bits for each reward. Then we run the pseudo-polynomial algorithm to solve the resulting game with the rewards rounded down (and scaled to integers) because we do not have more bits at that point (Step 4). This can be done in polynomial-time as long as we have $O(\log n)$ bits. Then we try to certify that the solution obtained is also a solution for the true rewards (Step 6). If this succeeds, then we are done. If this fails, then we use one more bit and repeat the process.

To prove a smoothed polynomial running time, we need to show that with high probability a logarithmic number of bits suffices to compute an equilibrium for the original (untruncated) game. Furthermore, we have to devise a certificate proving that the computed equilibrium is indeed an equilibrium for the original game (we will show that such a certificate is given in Step 6). Both results are based on a sensitivity analysis of the game: we show that by changing the rewards slightly, an optimal strategy remains optimal for the changed game.

A key ingredient for our smoothed analysis is an adaption of the isolation lemma [10] to our setting. An adaption of the isolation lemma has already been used successfully in smoothed analysis of integer programs [2]. It basically says the following: Of course, there are exponentially many alternative strategies for each player. But if a player replaces the optimal strategy with an alternative strategy, the payoff for the respective player gets worse significantly.

Lemma 5 (Isolation Lemma). *Let E be a finite set, and $\mathcal{F} \subset \mathbb{R}_+^E$ be a family of (distinct) vectors, such that for any distinct $\rho, \rho' \in \mathcal{F}$, there exists an $e \in E$ with $|\rho_e - \rho'_e| \geq \gamma$. Let $\{w_e\}_{e \in E}$ be independent continuous random variables with maximum density ϕ. Define $\mathrm{gap}(w) := w^T\rho^* - w^T\rho^{**}$, where $\rho^* = \mathrm{argmax}_{\rho \in \mathcal{F}}\, w^T\rho$ and $\rho^{**} = \mathrm{argmax}_{\rho \in \mathcal{F},\ \rho \neq \rho^*}\, w^T\rho$. Then $\Pr(\mathrm{gap}(w) \leq \varepsilon) \leq |E|\varepsilon\phi\frac{\kappa^2}{\gamma}$, where $\kappa = \max_{e \in E} |\mathcal{F}_e|$, and $\mathcal{F}_e = \{x \mid \rho_e = x \text{ for some } \rho \in \mathcal{F}\}$.*

We use the above lemma with the set $\mathcal{F}$ representing a set of arc-limiting distributions, corresponding to a set of situations in the game starting from a certain vertex. For that we need bounds for κ and γ.

Lemma 6. *Let $\mathcal{G} = (G = (V, E), P, r)$ be a BWR-game, $u \in V$ be any vertex, and s be an arbitrary situation. Then (i) every entry of the arc-limiting distribution $\rho(s)$ for the Markov chain $(\mathcal{G}(s), u)$ can be written as rational numbers of the form $\frac{a}{b}$, where $a, b \in \mathbb{Z}_+$ and $a, b \leq kn(2D)^{k+2}$. Hence, (ii) the number of possible entries in $\rho(s)$ is bounded by $\kappa = (kn)^2(2D)^{2(k+2)}$, and (iii) for any situation s' such that $\rho(s') \neq \rho(s)$, there is an arc e such that $\rho_e(s) - \rho_e(s') \geq \gamma = \gamma(\mathcal{G})$.*

To use the given pseudo-polynomial algorithm, we have to truncate the (perturbed) rewards after a certain number of bits. The following lemma assures that this is possible (with high probability) without changing the optimal strategies, as long as the rounded rewards and the true rewards are close enough. Before we state the lemma, it is useful to observe that, if the rewards are continuous, independently distributed random variables, then, for any two distinct situations s and s', we have $\Pr(\mu_u(\mathcal{G}(s)) = \mu_u(\mathcal{G}(s'))) = 0$ if and only if $\rho(s) \neq \rho(s')$. Thus for the structurally ergodic case, with probability one, two distinct situations result in two distinct values. On the other hand, in the general case, there might be many optimal situations, but all of them lead to the same limiting distribution.

Given a strategy $s_W \in S_W$ of WHITE we call a *uniform best response* (UBR) of BLACK any strategy $s_B^* \in S_B$, such that $\mu_u(\mathcal{G}(s_W, s_B^*)) \leq \mu_u(\mathcal{G}(s_W, s_B))$ for all $s_B \in S_B$. Similarly, a UBR of WHITE is defined. (Note that the existence of such a UBR is an immediate corollary of the existence of uniformly optimal situations in BWR-games.) We denote by $\mathrm{UBR}_{\mathcal{G}}(s_W)$ and $\mathrm{UBR}_{\mathcal{G}}(s_B)$ the sets of uniform best responses in $\mathcal{G}$, corresponding to strategies s_W and s_B, respectively.

Lemma 7. *Let $\mathcal{G} = (G = (V, E), P, r)$, $\mathcal{G}' = (G = (V, E), P, r')$ be two BWR-games such that $r = (r_e)_{e \in E}$ is a vector of independent continuous random variables with maximum density ϕ, and $\|r' - r\|_\infty \leq \varepsilon$, for some given $\varepsilon > 0$. Let $\theta := \frac{2n^3\varepsilon\phi}{\gamma(\mathcal{G})^3}$. Then, the following holds for any situation s:*

i. $\Pr(s$ *is not uniformly optimal in* $\mathcal{G}' \mid s$ *is uniformly optimal in* $\mathcal{G}) \leq 2\theta$;
ii. $\Pr(s$ *is not uniformly optimal in* $\mathcal{G} \mid s$ *is uniformly optimal in* $\mathcal{G}') \leq 2\theta$.

Still, it can happen that rounding results in different optimal strategies. How can we be sure that the solution obtained from the rounded rewards is also optimal for the game with the true rewards? Step 6 in Algorithm 3 is one way to do this. The basic idea is as follows: Let $\tilde{s}$ be a uniformly optimal situation in the rounded game. Lemma 7 says that with high probability $\tilde{s}$ is a uniformly optimal situation in $\mathcal{G}$, and hence it is also uniformly optimal in any game on the same graph and transition matrix, but with rewards lying in a small interval around the rounded rewards. Thus, we create $|E|$ copies of the truncated game; in each copy the reward on a single arc is perturbed by a certain amount within this small interval. If $\tilde{s}$ is uniformly optimal in all these games, then it is also uniformly optimal for all rewards from that small interval. The following lemma justifies the correctness of this certificate.

Lemma 8. *Let $\tilde{\mathcal{G}} = (G = (V, E), P, \tilde{r})$ be a BWR-game and u be an arbitrary vertex. Consider a situation $\tilde{s} = (\tilde{s}_W, \tilde{s}_B)$ such that, for all $e \in E$, $\tilde{s}$ is optimal in the game $(\tilde{\mathcal{G}}_{e,\varepsilon}, u)$ (defined in (4)). Then $\tilde{s}$ is also optimal in $(\mathcal{G} = (G, P, r), u)$, for any r such that $\|r - \tilde{r}\|_\infty \leq \varepsilon$.*

Now, we have all ingredients to prove that Algorithm 3 solves, in uniformly optimal strategies and in smoothed polynomial time, any BWR-game on a graph which admits a pseudo-polynomial algorithm and have a constant number of random vertices. This establishes Theorem 2.

References

1. Andersson, D., Miltersen, P.B.: The complexity of solving stochastic games on graphs. In: Dong, Y., Du, D.-Z., Ibarra, O. (eds.) ISAAC 2009. LNCS, vol. 5878, pp. 112–121. Springer, Heidelberg (2009)
2. Beier, R., Vöcking, B.: Typical properties of winners and losers in discrete optimization. SIAM J. Comput. 35(4), 855–881 (2006)
3. Boros, E., Elbassioni, K.M., Gurvich, V., Makino, K.: A pumping algorithm for ergodic stochastic mean payoff games with perfect information. In: Eisenbrand, F., Shepherd, F.B. (eds.) IPCO 2010. LNCS, vol. 6080, pp. 341–354. Springer, Heidelberg (2010)
4. Gillette, D.: Stochastic games with zero stop probabilities. In: Dresher, M., Tucker, A.W., Wolfe, P. (eds.) Contribution to the Theory of Games III. Annals of Mathematics Studies, vol. 39, pp. 179–187. Princeton University Press, Princeton (1957)
5. Gurvich, V., Karzanov, A., Khachiyan, L.: Cyclic games and an algorithm to find minimax cycle means in directed graphs. USSR Computational Mathematics and Mathematical Physics 28, 85–91 (1988)
6. Hoffman, A.J., Karp, R.M.: On nonterminating stochastic games. Management Science, Series A 12(5), 359–370 (1966)
7. Karzanov, A.V., Lebedev, V.N.: Cyclical games with prohibition. Mathematical Programming 60, 277–293 (1993)
8. Liggett, T.M., Lippman, S.A.: Stochastic games with perfect information and time-average payoff. SIAM Review 4, 604–607 (1969)
9. Mine, H., Osaki, S.: Markovian decision process. Elsevier, Amsterdam (1970)
10. Mulmuley, K., Vazirani, U.V., Vazirani, V.V.: Matching is as easy as matrix inversion. Combinatorica 7(1), 105–113 (1987)
11. Roth, A., Balcan, M.-F., Kalai, A., Mansour, Y.: On the equilibria of alternating move games. In: SODA, pp. 805–816 (2010)
12. Spielman, D.A., Teng, S.-H.: Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. J. ACM 51(3), 385–463 (2004)
13. Spielman, D.A., Teng, S.-H.: Smoothed analysis: An attempt to explain the behavior of algorithms in practice. C. ACM 52(10), 76–84 (2009)
14. Vorobyov, S.: Cyclic games and linear programming. Discrete Appl. Math. 156(11), 2195–2231 (2008)
15. Zwick, U., Paterson, M.: The complexity of mean payoff games on graphs. Theoret. Comput. Sci. 158(1-2), 343–359 (1996)

Pairwise-Interaction Games

Martin Dyer and Velumailum Mohanaraj

School of Computing, University of Leeds
Leeds LS2 9AS, United Kingdom
http://www.comp.leeds.ac.uk

Abstract. We study the complexity of computing Nash equilibria in games where players arranged as the vertices of a graph play a *symmetric* 2*-player game* against their neighbours. We call this a *pairwise-interaction game*. We analyse this game for n players with a fixed number of actions and show that (1) a mixed Nash equilibrium can be computed in constant time for any game, (2) a pure Nash equilibrium can be computed through Nash dynamics in polynomial time for games with a symmetrisable payoff matrix, (3) determining whether a pure Nash equilibrium exists for zero-sum games is NP-complete, and (4) counting pure Nash equilibria is #P-complete even for 2-strategy games. In proving (3), we define a new *defective* graph colouring problem called *Nash colouring*, which is of independent interest, and prove that its decision version is NP-complete. Finally, we show that pairwise-interaction games form a proper subclass of the usual graphical games.

Keywords: Nash equilibrium, graphical game, computational complexity, pairwise interaction.

1 Introduction

1.1 Overview

The *Nash equilibrium* [21] is the central solution concept in game theory. Plausibility of an equilibrium concept like the Nash equilibrium is partly determined by the complexity of computing equilibria [14]. As a result, many recent studies have focused on the complexity of finding Nash equilibria (e.g. [1,4,5,7,11,12,13]). For the complexity problem to be meaningful, however, the game, particularly its payoffs, should allow a compact representation [23].

Many succinctly representable games have been studied in the literature of which *graphical games*, proposed by Kearns *et al.* [18], have received much attention (see [5] and the references therein). In these games, players are arranged as the vertices of a graph and can play the game only with their immediate neighbours. In effect, a vertex k of degree d_k plays a $(d_k + 1)$-player game. If the number of pure strategies available to k is r, payoffs for k can be specified using r^{d_k+1} numbers. Thus, an n-player game with r strategies can be represented by an n-vertex graph and $nr^{\Delta+1}$ numbers, where Δ is the maximum vertex degree.

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 159–170, 2011.

The representation can be further simplified using symmetries in games. A game is *symmetric* if every player has the same payoff matrix and a player's payoff depends only on the player's strategy and the number of other players playing each pure strategy available. Hence, a symmetric graphical game (see [5] for a formal definition) with the neighbourhood size of d can be described by a d-regular graph with n vertices and $r\binom{d+r-1}{r-1}$ numbers. It must be noted that a symmetric graphical game cannot be considered symmetric by definition unless the graph is highly regular like the complete graph. In these games, each player plays the game with a different group of players and largely ignores what happens outside the group, whereas symmetry is defined globally.

The idea of playing games on graphs predates the idea of graphical games. Nearly a decade before graphical games were introduced by [18], Nowak and May [22] empirically studied the impact on the emergence of cooperation of placing players at the vertices of a grid graph. Their investigations stimulated research in this area, and a spate of new work followed, studying the impact of many other types of graph (see [24] and the references therein). In this setting, players are arranged as the vertices of a graph. Each vertex chooses a single pure or mixed strategy from a common strategy space and plays an identical, but independent, symmetric 2-player game with its immediate neighbours using this strategy. The payoff for a vertex is the sum of the payoffs it receives from playing the 2-player game with all its neighbours. This captures the natural tendency of players to treat each interaction as a separate 2-player game when the interactions are pairwise. This game, which we call a *pairwise-interaction game*, is the subject of this paper. Pairwise-interaction games are not necessarily symmetric, due to the reason stated above for symmetric graphical games, even though the 2-player games are assumed to be symmetric. Moreover, in pairwise interaction games, neighbourhoods of players can even be of different sizes.

Predictably, pairwise-interaction games can be represented even more succinctly than symmetric graphical games. More precisely, an n-player pairwise-interaction game with r strategies can be described by an n-vertex graph and a single $r \times r$ matrix. Strategic interactions in political, social, biological and economic situations are often pairwise [3]. Hence, pairwise-interaction games have many applications while admitting an extremely compact representation.

Surprisingly though, to the best of our knowledge, a systematic study of pairwise-interaction games has not been carried out from the computational game theory perspective. That is the main purpose of this paper. Perhaps unsurprisingly, it turns out that the set of pairwise-interaction games is in fact a *proper* subclass of graphical games. What makes our study even more interesting, but surely disappointing from a game-theoretic point of view, is the fact that even for this simple subclass, the problem of deciding whether a pure Nash equilibrium exists is hard for zero-sum games with more than two strategies.

1.2 Our Results

In this paper, we study n-player pairwise-interaction games with a fixed number of strategies r. Clearly, Nash's theorem [21] that there exists a mixed Nash

equilibrium in all finite games holds for pairwise-interaction games. Thus, we have the following easy theorem about mixed strategies. Many detailed proofs including that of Theorem 1 are omitted for brevity.

Theorem 1. *For any pairwise-interaction game with a fixed number of pure strategies, a symmetric mixed Nash equilibrium can be computed in constant time. This strategy corresponds to all players playing the symmetric mixed Nash equilibrium strategy for the 2-player game.*

Although mixed Nash equilibria exist in any game, there is no convincing justification for players deliberately randomising their actions. Hence, the pure Nash equilibrium is considered a better solution concept for games where one exists. This gives rise to two computational problems: (1) does a given game have any pure Nash equilibrium? (2) if it has, can that be computed in polynomial time? We address these questions for pairwise-interaction games. We first prove the following theorem that the *Nash dynamics* converges for games with symmetric matrices. This is the simple dynamics in which, at every iteration, some player switches to the best response to the current strategies of their neighbours. Its convergence implies the existence of a pure Nash equilibrium.

Theorem 2. *For any pairwise-interaction game with r strategies and a symmetric payoff matrix, the Nash dynamics converges in at most $n2^{K/2}(2\Delta+1)^{K+1}$ steps, where $K = r(r+1)/2-2$ and Δ is the maximum vertex degree.*

We show further that adding a constant to any column of the payoff matrix does not affect the Nash equilibria. So, the above result applies to games with payoff matrices that can be symmetrised using this operation. This, in particular, means that the above result applies to all 2-strategy games.

Perhaps more significantly, in Section 4 we prove the following theorem for *zero-sum* games.

Theorem 3. *For all $r \geq 3$, and all antisymmetric $r \times r$ payoff matrices $\mathbf{A}$ such that the 2-player game has a unique mixed strategy which is not a pure strategy, deciding whether there is a pure Nash equilibrium in the pairwise-interaction game with payoff matrix $\mathbf{A}$ on a Δ-regular graph is NP-complete.*

That the mixed strategy cannot be a pure strategy is clear, since otherwise all players playing this strategy would give a pure Nash equilibrium by Theorem 1. The condition of having a unique mixed strategy is made for technical reasons, and we believe the theorem to be true without this assumption. However, we note that having a unique mixed strategy is generic, and we will show this in Lemma 5.

In Section 5 we show that even for some 2-strategy pairwise-interaction games for which the problem of finding a pure Nash equilibrium is in FP, the problem of exactly counting them is #P-hard. Surprisingly, it turns out that even approximately counting them in polynomial time is not possible unless NP=RP. Finally, we have the following theorem that pairwise-interaction games form a (small) proper subset of symmetric graphical games. Thus our hardness results are much stronger than those previously known for graphical games.

Theorem 4. *For given $r > 2$ and $\Delta = \Omega(r)$, pairwise-interaction games form an exponentially small fraction of symmetric graphical games on Δ-regular graphs.*

1.3 Related Work

Two-strategy games. The *parity affiliation game* [12] with all edge weights -1 and the *cut game* [4] with all edge weights $+1$ correspond to the pairwise-interaction game with payoff matrix $\mathbf{P} = \left(\begin{smallmatrix} 0 & 1 \\ 1 & 0 \end{smallmatrix}\right)$. Any pure Nash equilibrium in these games is a STABLE-CONFIGURATION and a MAX-CUT, in the sense of Schäffer and Yannakakis [25]. Thus, finding a pure Nash equilibrium for these games is P-complete [25].

We also note that some 2-strategy games considered here are essentially equivalent to the *defective* 2*-colouring* problem [10]. Similar results to those we give are known for the defective 2-colouring problem.

Our convergence proof in Theorem 2 employs a potential function similar to that used in the convergence of Hopfield's neural networks [16] and other related problems. However, our proof applies to more than two strategies. In addition, due to the simplified nature of pairwise-interaction games, we are able to show that a pure Nash equilibrium can be computed in polynomial time.

Complexity of games. The problem of computing mixed Nash equilibria of n-player normal-form games is PPAD-complete for all $n \geq 2$ [7,11]. When $n \geq 4$, this problem is equivalent to the problem of computing Nash equilibria in graphical games of maximum degree $\Delta \geq 3$, with two strategies per player. Hence, the latter is also PPAD-complete [11]. Some positive results are known when these games are symmetric. A mixed Nash equilibrium of a symmetric n-player normal-form game with r strategies can be computed in polynomial time if $r = O(\log n / \log\log n)$ [23]. Using this result, it is shown in [5] that for the symmetric graphical games with degree Δ, an equilibrium can be computed in polynomial time if $r = O(\log \Delta / \log\log \Delta)$.

For symmetric n-player normal-form games with a constant number of strategies, r, the problem of determining the existence of a pure Nash equilibrium is in AC^0 [2,6]. It is known that these games are guaranteed to have a pure Nash equilibrium when $r = 2$ [8]. For graphical games, the problem of determining whether there exist a pure Nash equilibrium is NP-complete, in general, even if all players have only two strategies and neighbourhoods of size 2 [13].

The problem of counting Nash equilibria is generally hard. Counting the number of (mixed) Nash equilibria is #P-hard even for symmetric 2-player games [9]. For graphical games, counting the number of pure Nash equilibria is #P-hard even for symmetric games with neighbourhood size of 2 [5].

As we will see later, pairwise-interaction games with symmetrisable payoff matrices are in fact *general potential games* [12]. By definition, every potential game has a pure Nash equilibrium. *Congestion games* [12] are a special class of potential games. Computing a pure Nash equilibrium for these games is PLS-complete [12].

2 Preliminaries

2.1 Notations

If all elements of a matrix $\mathbf{A}$ or a vector $\mathbf{n}$ are positive, we write $\mathbf{A} \geq \mathbf{0}$ or $\mathbf{n} \geq \mathbf{0}$ respectively. Here and elsewhere, matrices and column vectors with all 1's and 0's are denoted by $\mathbf{1}$ and $\mathbf{0}$ respectively. By $\mathbf{A}^T$ and $\mathbf{n}^T$, we denote the transpose of $\mathbf{A}$ and $\mathbf{n}$ respectively. We write column vectors as row vectors with the transpose operation, e.g. $(n_0, \ldots, n_{r-1})^T$. An integer set $\{0, \ldots, x-1\}$ is denoted by $[x]$. Interchangeably, we refer to a participant in a pairwise-interaction game as a player or vertex, since each participant is represented by a graph vertex.

2.2 Strategic Games

Definition 1. *A* normal-form game *is given by a set of players $\mathcal{Q}$, and for each player $k \in \mathcal{Q}$ a finite set of pure strategies $\mathcal{S}_k$ and a payoff function $u_k : \times_{k \in \mathcal{Q}} \mathcal{S}_k \to \mathbb{R}$.*

A *pure strategy* of player k is an element of $\mathcal{S}_k$. A *mixed strategy* for player k is a probability distribution Σ_k over $\mathcal{S}_k$, so is a nonnegative vector of length $|\mathcal{S}_k|$. A set of strategies $s = (s_1, \ldots, s_k, \ldots, s_n)$ $(s_k \in \mathcal{S}_k)$ is called a *pure strategy profile*, and $\sigma = (\sigma_1, \ldots, \sigma_k, \ldots, \sigma_n)$ $(\sigma_k \in \Sigma_k)$ is called a *mixed strategy profile*.

A 2-player normal-form game can be conveniently represented by two real matrices $\mathbf{A} = (a_{ij})$ and $\mathbf{B} = (b_{ij})$. The game is *symmetric* if $\mathbf{B} = \mathbf{A}^T$, and is *zero-sum* if $\mathbf{A} + \mathbf{B} = \mathbf{0}$. Hence for a symmetric zero-sum game, $\mathbf{A}$ is *antisymmetric*. Thus, one payoff matrix $\mathbf{A}$ is sufficient to describe any symmetric 2-player game.

Let $G = (V, E)$ be a graph. Let $\mathcal{N}(k) = \{v \in V \mid (k, v) \in E\}$, and $d_k = |\mathcal{N}(k)|$. By s_{-k} and σ_{-k} we denote the pure and mixed strategies of all neighbours of k respectively. Then, the pairwise-interaction game is defined as follows.

Definition 2. *A* pairwise-interaction game $\mathcal{G}$ *is defined by:*

- *An undirected graph $G = (V, E)$, where the vertices $V = [n]$ represent players. Without loss of generality, we may assume G is connected.*
- *A symmetric 2-player game $\langle \mathcal{S}, \mathbf{A} \rangle$, where $\mathcal{S} = [r]$ is the set of pure strategies available to each vertex and $\mathbf{A} = (a_{ij})$ $(i, j \in [r])$ is the payoff matrix.*
- *The payoff for any player k $(k \in [n])$ is*

$$u(\sigma_k; \sigma_{-k}) \;=\; \sum_{p \in \mathcal{N}(k)} \sigma_k^T \mathbf{A} \sigma_p \;. \tag{1}$$

We will regard r as being a constant. We show later that the numbers in $\mathbf{A}$ can be taken as being polynomially bounded in n, so the size of these numbers does not need be included in the input size. Thus, for complexity purposes, the input size is measured only by n.

We shall denote the set of mixed strategies over $\mathcal{S}$ by Σ. To avoid trivialities, we will always assume $r \geq 2$. If the 2-player game is zero-sum, we refer to the pairwise-interaction game as zero-sum.

Let $\mathcal{B}(\sigma_{-k})$ be the set of mixed strategy best responses of vertex k to the neighbour strategies σ_{-k}. Then we have

$$\mathcal{B}(\sigma_{-k}) \;=\; \{\sigma_k \in \Sigma \mid u(\sigma_k;\sigma_{-k}) \geq u(\sigma'_k;\sigma_{-k})\ \forall \sigma'_k \in \Sigma\ \}\ . \tag{2}$$

Definition 3. *A strategy profile* $\sigma^* = \left\{\sigma^*_0, \ldots, \sigma^*_k, \ldots, \sigma^*_{n-1}\right\}$ $(\sigma^*_k \in \Sigma)$ *is a* mixed Nash equilibrium *if* $\sigma^*_k \in \mathcal{B}(\sigma_{-k})\ \forall k \in V$. *We say* σ^* *is a* pure Nash equilibrium *if it is a pure strategy profile.*

Let $n_j^{(k)}$ denote the number of neighbours of k playing strategy $j \in \mathcal{S}$. We shall call a combination of neighbour strategies a *neighbourhood*. Then, instead of using s_{-k} to denote it, for symmetric games, it is convenient to use a column vector of $n_j^{(k)}$ with one entry for each $j \in [r]$, e.g. $\mathbf{n_k} = \left(n_0^{(k)}, \ldots, n_{r-1}^{(k)}\right)^T$ where $\sum_{j=0}^{r-1} n_j^{(k)} = d_k$. Using this notation, for pure strategies, (1) could be rewritten as

$$u\left(s_k; n_0^{(k)}, \ldots, n_{r-1}^{(k)}\right) \;=\; \sum_{j\in\mathcal{S}} n_j^{(k)} a_{s_k j}\ . \tag{3}$$

Similarly, (2) could be written as $\mathcal{B}(n_0^{(k)}, \ldots, n_{r-1}^{(k)})$. We will use this notation in the analysis of pure Nash equilibria, and (1) and (2) in the analysis of mixed Nash equilibria.

Now the following proposition is easy to prove.

Proposition 1. *Adding an arbitrary constant to all entries of any column of* $\mathbf{A}$ *does not affect the Nash equilibria of pairwise-interaction games.*

We next define Nash dynamics and provide a proposition that links its convergence and the existence of a pure Nash equilibrium (e.g. [12]).

Definition 4. Nash Dynamics*: In this dynamics, at every step, some player playing a suboptimal strategy improves their payoff by switching to the best response to the current strategies of their neighbours.*

Proposition 2. *If the Nash dynamics converges, then there is a pure Nash equilibrium.*

We use the following notion of equivalence of games throughout the paper.

Definition 5. *Two games are* equivalent *if they have identical best responses to every combination of opponents' strategies.*

3 Symmetric Payoff Matrices

In this section we prove Theorem 2 about pairwise interaction games with symmetrisable payoff matrix $\mathbf{A}$. The following lemma shows that there always exists a pure Nash equilibrium for these games.

Lemma 1. *An* r*-strategy pairwise-interaction game with symmetric payoff matrix* $\mathbf{A}$ *has a pure Nash equilibrium.*

Proof. We prove this using a potential function $\psi : \mathcal{S}^n \to \mathbb{R}$. Let $s = (s_1, \ldots, s_k, \ldots, s_n) \in \mathcal{S}^n$ be a pure strategy profile. Then $\psi(s)$ is defined as

$$\psi(s) \;=\; \sum_{k \in V} u\big(s_k; n_0^{(k)}, \ldots, n_{r-1}^{(k)}\big) \;=\; \sum_{k \in V} \sum_{j \in S} n_j^{(k)} a_{s_k j} \;. \tag{4}$$

Now suppose, at some point in the Nash dynamics, vertex k switches from its current strategy s_k to its best response $\bar{s_k}$, taking the strategy profile from s to $\bar{s} = (s_1, \ldots, \bar{s_k}, \ldots, s_n)$. Then the payoff of k increases by $\theta_k = \sum_{j \in S} a_{\bar{s_k} j} n_j^{(k)} - \sum_{j \in S} a_{s_k j} n_j^{(k)} > 0$, while the total payoff of the neighbours of k increases by $\theta_{\mathrm{N}} = \sum_{j \in S} a_{j \bar{s_k}} n_j^{(k)} - \sum_{j \in S} a_{j s_k} n_j^{(k)}$. By symmetry of $\mathbf{A}$, we have $\theta_k = \theta_{\mathrm{N}}$. Hence we get $\psi(\bar{s}) - \psi(s) = \theta_k + \theta_{\mathrm{N}} = 2\theta_k > 0$. So, at every step of the Nash dynamics, the potential function increases by a positive value. Thus, the Nash dynamics converges. □

Remark 1. The above proof shows that pairwise-interaction games with a symmetric payoff matrix belong to the class of *weighted potential games* [20]. The weight for each player is $1/2$: when a player improves his payoff by x, the potential function increases by $2x$.

The Nash dynamics converges for these games, but how long does this take? Theorem 2 states that the convergence is fast for the games considered in Lemma 1. To prove this, we need the following lemma.

Lemma 2. *Any $r \times r$ payoff matrix $\mathbf{A}$ can be rescaled such that the minimum difference between any two payoffs is one, and at least one payoff is zero. This can be done without affecting the Nash equilibria or the symmetry of $\mathbf{A}$. The rescaling requires only constant time for fixed r.*

Proof sketch of Theorem 2: We first rescale the payoff matrix using Lemma 2. Then, there is at least one payoff 0 and another 1. We consider the remaining payoffs as variables. Let $\mathbf{v} = (v_1, \ldots, v_K)^T$ denote the vector containing these variables, where K is the total number of variables. As $\mathbf{A}$ is symmetric, we have $K = r(r+1)/2 - 2$. Consider a vertex k with degree d. Let $P(r, d) = \binom{d+r-1}{r-1}$ be the number different neighbourhood configurations for k. Let $\mathbf{a}_i$ $(i \in [r])$ denote the rows of the payoff matrix. For each configuration of the neighbour strategies $\mathbf{n}_p$ $(p \in [P])$, the best response is determined by the ordering of $\mathbf{a}_i \mathbf{n}_p$ $(i \in [r])$. Now, for each neighbourhood $\mathbf{n}_p$ $(p \in [P])$, and each distinct pair of strategies i and j $(0 \le i, j \le r-1)$, we add the inequality $(\mathbf{a}_i - \mathbf{a}_j)^T \mathbf{n}_p \ge 1$ if i yields a higher payoff than j, and we add $(\mathbf{a}_i - \mathbf{a}_j)^T \mathbf{n}_p = 0$ if both strategies yield the same payoff. These inequalities form a convex nonempty polyhedron in K-dimensional space. It is nonempty because the original payoffs satisfy all these inequalities. This polyhedron defines a class of games that are equivalent to $\mathbf{A}$ and have the property that every best response move improves the player's payoff by at least 1. Let $\mathbf{N}\mathbf{v} = \mathbf{b}$ be the set of K inequalities that are tight at a vertex of the polyhedron. Applying Cramer's rule on this, we can find the coordinates of this

vertex in terms of the elements of $\mathbf{n}_p$'s. Then, Hadamard's inequality can be used to bound these coordinates in terms of Δ and r, which actually are the payoffs of an equivalent game. In this context, we may allow exponential dependence on r, since this is assumed to be constant. For the new, equivalent game, $\psi(s)$ is polynomially bounded and its value increases by at least 2 at every step. It can then be shown that the Nash dynamics converges as claimed. □

Note that the computation of an equivalent game in the above proof is valid even if $\mathbf{A}$ is not symmetric, except that K will now be $r^2 - 2$. Hence, the following holds by a similar proof to the one above.

Corollary 1. *For any pairwise-interaction game with a fixed number of strategies, there is an equivalent game with a payoff matrix whose entries are polynomially bounded in n.*

As mentioned before, the payoff matrix of any 2-strategy game can be symmetrised. Let $\mathbf{A} = \left(\begin{smallmatrix} P & T \\ S & R \end{smallmatrix}\right)$ be the original payoff matrix. This can be symmetrised to give $\mathbf{P} = \left(\begin{smallmatrix} \alpha & 0 \\ 0 & \beta \end{smallmatrix}\right)$, where $\alpha = P - S$ and $\beta = R - T$ (see Proposition 1). So, Theorem 2 applies to these games. However, in the following theorem, we get tighter results than that of Theorem 2 for regular graphs by exploiting the unique properties of the game, albeit using essentially similar techniques.

Theorem 5. *For any 2-strategy pairwise-interaction game on a Δ-regular graph $G = (V, E)$ with n vertices and m edges, starting from an arbitrary initial state, the Nash dynamics converges in at most $3n/2$ steps if $\alpha + \beta$ and β are of opposite signs and m steps if they are of the same sign.*

It might be possible to extend the above result to non-regular graphs. For example, consider the unweighted *cut game* where $\alpha = \beta = -1$. In this game, we have $0 \geq \psi(s) \geq \sum_{k \in V} -d_k = -2m$, so it takes only m steps for convergence on any graph. However, we have an alternative proof for general graphs.

Theorem 6. *For any 2-strategy pairwise-interaction game with payoff matrix $\mathbf{A} = \left(\begin{smallmatrix} P & T \\ S & R \end{smallmatrix}\right)$ on a graph with n vertices and m edges, starting from an arbitrary initial state, the Nash dynamics converges in at most (i) $3m - n$ steps if $T > R$ and $S > P$, (ii) $3m$ steps if $T < R$ and $S < P$, (iii) n steps, otherwise.*

Proof (Sketch). The proof is similar to the proofs of Lemma 1 and Theorem 5, but uses an algorithm similar to the graph partitioning algorithm of [15].

4 Zero-Sum Games

In this section, we study pairwise interaction zero-sum games with $r \geq 3$ strategies and prove Theorem 3. The main tool of the proof is the following proposition.

Proposition 3. *In a zero-sum pairwise-interaction game, the best response to any neighbourhood configuration yields a nonnegative payoff. Furthermore, in Nash equilibrium, every player earns zero payoff.*

The neighbourhoods in a Nash equilibrium can be characterised using the proposition above.

Definition 6. *In a zero-sum pairwise-interaction game, a neighbourhood will be called a Nash Equilibrium neighbourhood (NE neighbourhood) if the best response to the neighbourhood yields zero payoff.*

Corollary 2. *If* $\mathbf{n} = (n_0, \ldots, n_{r-1})^T$ *is a NE neighbourhood, then* $\mathbf{An} \leq \mathbf{0}$.

We now show that a highly nontrivial elimination of strategies is possible for zero-sum games, which, in a sense, is much stronger than the usual iterated elimination of dominated strategies. That is, if a strategy earns a negative payoff in any NE neighbourhood, it can be eliminated, implying that *any* surviving strategy is a best response to *any* NE neighbourhood.

Lemma 3. *If a strategy earns a negative payoff when played against a NE neighbourhood, no player will play it in any pure Nash equilibrium.*

We now consider the question of the existence of NE neighbourhoods for rational payoff matrices. But, as we shall see, this does not imply that a pure Nash equilibrium exists in a d-regular graph.

Lemma 4. *If* $\mathbf{A}$ *has rational entries then, for some integer* d*, there exists a NE neighbourhood for a vertex of degree* d*.*

The proof, which is omitted here, reveals a remarkable connection between a NE neighbourhood in a zero-sum pairwise interaction game and the optimal mixed strategy of the 2-player game. This suggests a heuristic approach to pairwise-interaction games: from each player's point of view, their neighbourhood can be viewed as a single opponent playing a mixed strategy. For a general pairwise-interaction game, this approach has no real validity. Since individual players are not able to play mixed strategies, the above view is asymmetric. But, surprisingly, for zero-sum games it is actually valid.

In this context, Lemma 4 can be linked to some well-known mixed strategy results. Consider games with a *unique mixed strategy*, by which we mean the games with a unique NE neighbourhood. These are games for which the surviving strategies are *completely mixed* [17]. (A completely mixed strategy is one for which every pure strategy has a positive probability.) Kaplansky [17] showed that a symmetric two-player zero-sum game can be completely mixed only if the number of strategies is odd. Thus, for games with a unique NE neighbourhood, the number of surviving strategies must be odd. For the remainder of this section we consider only payoff matrices $\mathbf{A}$ which have unique mixed strategies. But we note that there is always a matrix arbitrarily close to $\mathbf{A}$ for which this is true.

Lemma 5. *Let* $\mathbf{A} = (a_{ij})$ *be an antisymmetric payoff matrix. Then, there always exists an antisymmetric payoff matrix* $\mathbf{B} = (b_{ij})$ *that has a unique mixed strategy and satisfies* $|a_{ij} - b_{ij}| \leq 1/M$ $(i, j \in [r])$*, for any* $M > 0$.

Next, let us define an interesting computational problem related to improper vertex colouring that will be used in the proof of Theorem 3.

Definition 7. *NASH-COLOURABLE is a decision problem whose instance is a graph $G = (V, E)$, a set of colours $[r]$ and, for each vertex degree d in G, a set of r nonnegative integers $(c_0^d, c_1^d \ldots, c_{r-1}^d)$ such that $d = \sum_{i=0}^{r-1} c_i^d$. The question is whether there is an improper vertex colouring of G with r colours such that a vertex with degree d has exactly c_i^d neighbours with colour $i \in [r]$.*

If the answer is positive, the graph G will be said to be Nash colourable *and the particular assignment of colours will be called a* Nash colouring *of the graph.*

Definition 8. *$(c_0, c_1, \ldots, c_{r-1})_\Delta$-NASH-COLOURABLE will mean the Nash colouring problem for Δ-regular graphs. In this case we will write $(c_0^\Delta, c_1^\Delta, \ldots, c_{r-1}^\Delta) = (c_0, c_1, \ldots, c_{r-1})$, so $\Delta = \sum_{i=0}^{r-1} c_r$. Since there is only one vertex degree, we will specify $(c_0, c_1, \ldots, c_{r-1})$ in the prefix.*

Proof sketch of Theorem 3: A Nash equilibrium in the zero-sum pairwise-interaction game corresponds to a Nash colouring of the graph. The result then following from Theorem 7 that NASH-COLOURABLE is NP-complete. □

Theorem 7. *If $r \geq 3$, and $c_0, c_1, \ldots, c_{r-1}$ are any positive integers such that $\sum_{i=0}^{r-1} c_i = \Delta$, then $(c_0, c_1, \ldots, c_{r-1})_\Delta$-NASH-COLOURABLE is NP-complete.*

Proof (Sketch). The problem is clearly in NP. To prove that it is NP-hard, we use a gadget-based reduction from CHROMATIC-INDEX of r-regular graphs to $(c_0, c_1, \ldots, c_{r-1})_\Delta$-NASH-COLOURABLE. The hardness then follows from the result of [19] that CHROMATIC-INDEX is NP-complete for Δ-regular graphs with degree $\Delta \geq 3$. □

5 Some Further Results

The following two theorems consider some games whose Nash equilibria correspond to maximal independent sets in the corresponding graphs. For these games, Theorem 8 shows that exactly counting the Nash equilibria is hard while Theorem 9 proves that even counting them approximately is hard.

Theorem 8. *Suppose a 2-strategy pairwise-interaction game with payoff matrix $\mathbf{A} = \left(\begin{smallmatrix} P & T \\ S & R \end{smallmatrix}\right)$ is played on a graph of maximum degree 4. Then, if the payoffs are such that $T > R$, $S > P$ and $0 < \gamma < 1/4$ or $3/4 < \gamma < 1$, where $\gamma = (S - P)/(T + S - R - P)$, the problem of counting the pure Nash equilibria is #P-complete.*

Theorem 9. *For the same game considered in Theorem 8 except that the game is played on a graph of maximum degree 7 and the payoffs are such that $0 < \gamma < 1/7$ or $6/7 < \gamma < 1$, there does not exist a fully polynomial time approximation scheme (FPTAS) to count the pure Nash equilibria unless RP=NP.*

For some 2-strategy pairwise interaction games, computation of a pure Nash equilibrium is inherently sequential, and the following theorem holds.

Theorem 10. *The problem of computing pure Nash equilibria is P-complete for some 2-strategy pairwise-interaction games.*

Thus finding a pure Nash equilibrium almost certainly cannot be done in constant time, in sharp contrast to Theorem 1 for mixed Nash equilibria.

6 Open Problems

We have initiated a systematic study of pairwise-interaction games and presented results for the games with symmetric or antisymmetric payoff matrix. A natural extension of our work is to investigate the remaining case, i.e. the games with asymmetric payoff matrices that are not antisymmetric. We have examples of matrices of this kind for which it is easy to compute a pure Nash equilibrium. Hence, we know that there are easy cases left to study, and we believe there are also hard cases. But, we conjecture that there is a *dichotomy*: the problem of deciding whether a pure Nash equilibrium exists is either in P or is NP-complete. Similarly, we believe that for the problem of counting Nash equilibria there is a dichotomy, thus the problem is in FP or is #P-complete. We leave finding the dichotomy conditions as open problems. For the approximate counting problem, we showed that there does not exist an FPTAS even for some 2-strategy games. Classifying the complexity of approximately counting Nash equilibria for these games we leave as another open question. In addition, considering our hardness results, another topic of interest would be to explore approximate Nash equilibria for these games.

Two-strategy pairwise-interaction games on the complete graph can be modelled as congestion games [8]. In [1], the combinatorial structures in congestion games that ensure that the Nash dynamics converges in polynomial time are studied. It would be interesting to explore if there is any connection between the congestion games with a polynomial time convergence and other pairwise-interaction games with a polynomial time convergence.

References

1. Ackermann, H., Roglin, H., Vocking, B.: On the impact of combinatorial structure on congestion games. In: Proceedings of the 47th IEEE Symposium on Foundations of Computer Science (FOCS 2006), pp. 613–622 (2006)
2. Àlvarez, C., Gabarró, J., Serna, M.: Pure Nash equilibria in games with a large number of actions. In: Jedrzejowicz, J., Szepietowski, A. (eds.) MFCS 2005. LNCS, vol. 3618, pp. 95–106. Springer, Heidelberg (2005)
3. Berninghaus, S.K., Haller, H.: Pairwise interaction on random graphs, Tech. Report 06–16, Sonderforschungsbereich 504, University of Mannheim (February 2007)
4. Bhalgat, A., Chakraborty, T., Khanna, S.: Approximating pure Nash equilibrium in cut, party affiliation, and satisfiability games. In: Proceedings of the 11th ACM Conference on Electronic Commerce (EC 2010), pp. 73–82 (2010)
5. Brandt, F., Fischer, F., Holzer, M.: Equilibria of graphical games with symmetries. In: Papadimitriou, C., Zhang, S. (eds.) WINE 2008. LNCS, vol. 5385, pp. 198–209. Springer, Heidelberg (2008)

6. Brandt, F., Fischer, F., Holzer, M.: Symmetries and the complexity of pure Nash equilibrium. Journal of Computer and System Sciences 75, 163–177 (2009)
7. Chen, X., Deng, X.: Settling the complexity of two-player Nash equilibrium. In: Proceedings of the 47th IEEE Symposium on Foundations of Computer Science (FOCS 2006), pp. 261–272 (2006)
8. Cheng, S., Reeves, D.M., Vorobeychik, Y., Wellman, M.P.: Notes on the equilibria in symmetric games. In: Proceedings of the 6th International Workshop on Game Theoretic and Decision Theoretic Agents (GTDT 2004), pp. 71–78 (2004)
9. Conitzer, V., Sandholm, T.: New complexity results about Nash equilibria. Games and Economic Behavior 63, 621–641 (2008)
10. Cowen, L.J., Goddard, W., Jesurum, C.E.: Defective coloring revisited. J. Graph Theory 24, 205–219 (1995)
11. Daskalakis, C., Goldberg, P.W., Papadimitriou, C.H.: The complexity of computing a Nash equilibrium. Commun. ACM 52, 89–97 (2009)
12. Fabrikant, A., Papadimitriou, C.H., Talwar, K.: The complexity of pure Nash equilibria. In: Proceedings of the 36th ACM Symposium on Theory of Computing (STOC 2004), pp. 604–612 (2004)
13. Fischer, F., Holzer, M., Katzenbeisser, S.: The influence of neighbourhood and choice on the complexity of finding pure Nash equilibria. Information Processing Letters 99, 239–245 (2006)
14. Gilboa, I., Zemel, E.: Nash and correlated equilibria: Some complexity considerations. Games and Economic Behavior 1, 80–93 (1989)
15. Halldórsson, M.M., Lau, H.C.: Low-degree graph partitioning via local search with applications to constraint satisfaction, max cut, and coloring. Journal of Graph Algorithms and Applications 1, 1–13 (1997)
16. Hopfield, J.J.: Neural networks and physical systems with emergent collective computational abilities. Proceedings of the National Academy of Sciences of the United States of America 79, 2554–2558 (1982)
17. Kaplansky, I.: A contribution to von Neumann's theory of games. The Annals of Mathematics 46, 474–479 (1945)
18. Kearns, M.J., Littman, M.L., Singh, S.P.: Graphical models for game theory, Proceedings of the 17th Conference in Uncertainty in Artificial Intelligence (UAI 2001), pp. 253–260 (2001)
19. Leven, D., Galil, Z.: NP-completeness of finding the chromatic index of regular graphs. Journal of Algorithms 4, 35–44 (1983)
20. Monderer, D., Shapley, L.S.: Potential games. Games and Economic Behavior 14, 124–143 (1996)
21. Nash, J.: Non-cooperative games. The Annals of Mathematics 54, 286–295 (1951)
22. Nowak, M.A., May, R.M.: Evolutionary games and spatial chaos. Nature 359, 826–829 (1992)
23. Papadimitriou, C.H., Roughgarden, T.: Computing equilibria in multi-player games. In: Proceedings of the 16th ACM-SIAM Symposium on Discrete Algorithms (SODA 2005), pp. 82–91 (2005)
24. Santos, F.C., Rodrigues, J.F., Pacheco, J.M.: Graph topology plays a determinant role in the evolution of cooperation. Proceedings of the Royal Society B: Biological Sciences 273, 51–55 (2006)
25. Schäffer, A.A., Yannakakis, M.: Simple local search problems that are hard to solve. SIAM J. Comput. 20, 56–87 (1991)

Settling the Complexity of Local Max-Cut (Almost) Completely*

Robert Elsässer and Tobias Tscheuschner

University of Paderborn
Faculty of Computer Science, Electrical Engineering and Mathematics
{elsa,chessy}@upb.de

Abstract. We consider the problem of finding a local optimum for the Max-Cut problem with FLIP-neighborhood, in which exactly one node changes the partition. Schäffer and Yannakakis (SICOMP, 1991) showed $\mathcal{PLS}$-completeness of this problem on graphs with unbounded degree. On the other side, Poljak (SICOMP, 1995) showed that in cubic graphs every FLIP local search takes $O(n^2)$ steps, where n is the number of nodes. Due to the huge gap between degree three and unbounded degree, Ackermann, Röglin, and Vöcking (JACM, 2008) asked for the smallest d such that on graphs with maximum degree d the local Max-Cut problem with FLIP-neighborhood is $\mathcal{PLS}$-complete. In this paper, we prove that the computation of a local optimum on graphs with maximum degree five is $\mathcal{PLS}$-complete. Thus, we solve the problem posed by Ackermann et al. almost completely by showing that d is either four or five (unless $\mathcal{PLS} \subseteq \mathcal{P}$).

On the other side, we also prove that on graphs with degree $O(\log n)$ every FLIP local search has probably polynomial smoothed complexity. Roughly speaking, for any instance, in which the edge weights are perturbated by a (Gaussian) random noise with variance σ^2, every FLIP local search terminates in time polynomial in n and σ^{-1}, with probability $1-n^{-\Omega(1)}$. Putting both results together, we may conclude that although local Max-Cut is likely to be hard on graphs with bounded degree, it can be solved in polynomial time for slightly perturbated instances with high probability.

Keywords: Max-Cut, PLS, graphs, local search, smoothed complexity.

1 Introduction

For an undirected graph $G = (V, E)$ with weighted edges $w : E \to \mathbb{N}$ a cut is a partition of V into two sets V_1, V_2. The weight of the cut is the sum of the weights of the edges connecting nodes between V_1 and V_2. The **Max-Cut** problem asks for a cut of maximum weight. Computing a maximum cut is one of the most famous problems in computer science and is known to be $\mathcal{NP}$-complete even

* Partially supported by the German Research Foundation (DFG) Priority Programme 1307 "Algorithm Engineering".

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 171–182, 2011.

on graphs with maximum degree three [8]. For a survey of MAX-CUT including applications see [16].

A frequently used approach of dealing with hard combinatorial optimization problems is local search. In local search, to every solution there is assigned a set of neighbor solutions, i. e. a neighborhood. The search begins with an initial solution and iteratively moves to better neighbors until no better neighbor can be found. For a survey of local search, we refer to [13]. To encapsulate many local search problems, Johnson et al. [9] introduced the complexity class $\mathcal{PLS}$ (polynomial local search) and initially showed $\mathcal{PLS}$-completeness for the CIRCUIT-FLIP problem. Schäffer and Yannakakis [21] showed $\mathcal{PLS}$-completeness for many popular local search problems including the local MAX-CUT problem with FLIP-neighborhood – albeit their reduction builds graphs with linear degree in the worst case. Moreover, they introduced the notion of so called *tight* $\mathcal{PLS}$-reductions which preserve not only the existence of instances and initial solutions that are exponentially many improving steps away from any local optimum but also the $\mathcal{PSPACE}$-completeness of the computation of a local optimum reachable by improving steps from a given solution.

In a recent paper Monien and Tscheuschner [14] showed the two properties that are preserved by tight $\mathcal{PLS}$-completeness proofs for the local MAX-CUT problem on graphs with maximum degree four. However, their proof did not use a $\mathcal{PLS}$-reduction; they left open whether the local MAX-CUT problem is $\mathcal{PLS}$-complete on graphs with maximum degree four. For cubic graphs, Poljak [15] showed that any FLIP-local search takes $O(n^2)$ improving steps, where Loebl [12] earlier showed that a local optimum can be found in polynomial time using an approach different from local search. Thus, it is unlikely that computing a local optimum is $\mathcal{PLS}$-complete on graphs with maximum degree three.

Due to the huge gap between degree three and unbounded degree, Ackermann et al. [2] asked for the smallest d such that on graphs with maximum degree d the computation of a local optimum is $\mathcal{PLS}$-complete. In this paper, we show that d is either four or five (unless $\mathcal{PLS} \subseteq \mathcal{P}$), and thus solve the above problem almost completely. A related problem has been considered by Krentel [10]. He showed $\mathcal{PLS}$-completeness for a satisfiability problem with trivalent variables, a clause length of at most four, and maximum occurrence of the variables of three.

Our result has impact on many other problems, since the local MAX-CUT has been the basis for many $\mathcal{PLS}$-reductions in the literature. Some of these reductions directly carry over the property of maximum degree five in some sense and result in $\mathcal{PLS}$-completeness of the corresponding problem even for very restricted sets of feasible inputs. In particular, $\mathcal{PLS}$-completeness follows for the MAX-2SAT problem [21] with FLIP-neighborhood, in which exactly one variable changes its value, even if every variable occurs at most ten times. $\mathcal{PLS}$-completeness also follows for the problem of computing a Nash Equilibrium in CONGESTION GAMES (cf. [6], [2]) in which each strategy contains at most five resources. The problem to PARTITION [21] a graph into two equally sized sets of nodes by minimizing or maximizing the weight of the cut, where the maximum degree is six and the neighborhood consists of all solutions in which two nodes of

different partitions are exchanged, is also $\mathcal{PLS}$-complete. Moreover, our $\mathcal{PLS}$-completeness proof was already helpful showing a complexity result in hedonic games [7].

In this paper, we also consider the smoothed complexity of any FLIP local search on graphs in which the degrees are bounded by $O(\log n)$. This performance measure has been introduced by Spielman and Teng in their seminal paper on the smoothed analysis of the Simplex algorithm [19].[1] Since then, a large number of papers deal with the smoothed complexity of different algorithms. In most cases, smoothed analysis is used to explain the speed of certain algorithms in practice, which have an unsatisfactory running time according to their worst case complexity.

The smoothed measure of an algorithm on some input instance is its expected performance over random perturbations of that instance, and the smoothed complexity of an algorithm is the maximum smoothed measure over all input instances. In the case of an LP, the goal is to maximize $z^T x$ subject to $Ax \leq b$, for given vectors z, b, and matrix A, where the entries of A are perturbated by Gaussian random variables with mean 0 and variance σ^2. That is, we add to each entry $a_{i,j}$ some value $\max_{i,j} a_{i,j} \cdot y_{i,j}$, where $y_{i,j}$ is a Gaussian random variable with mean 0 and standard deviation σ. Spielman and Teng showed that an LP, which is perturbated by some random noise as described before, has expected running time polynomial in n, m, and σ. This result has further been improved by Vershynin [22]. The smoothed complexity of other linear programming algorithms has been considered in e.g. [3], and quasi-concave minimization was studied in [11].

Several other algorithms from different areas have been analyzed w. r. t. their smoothed complexity (see [20] for a comprehensive description). Two prominent examples of local search algorithms with polynomial smoothed complexity are 2-opt TSP [5] and k-means [1]. We also mention here the papers of Beier, Röglin, and Vöcking [4,18] on the smoothed analysis of integer linear programming. They showed that if Π is a certain class of integer linear programs, then Π has an algorithm of probably polynomial smoothed complexity[2] iff $\Pi_u \in ZPP$, where Π_u is the unary representation of Π, and ZPP denotes the class of decision problems solvable by a randomized algorithm with polynomial expected running time that always returns the correct answer. The results of [4,18] imply that e.g. 0/1-knapsack, constrained shortest path, and constrained minimum weighted matching have probably polynomial smoothed complexity. Unfortunately, the results of these papers cannot be used to settle the smoothed complexity of local Max-Cut.

Overview. In section 3, we introduce a technique by which we substitute graphs whose nodes of degree greater than five have a certain type – we will call these nodes **comparing** – by graphs of maximum degree five. In particular, we show that certain local optima in the former graphs induce unique local optima in

[1] For this work, Spielman and Teng was awarded the Gödel Prize in 2008.
[2] For the definition of probably polynomial smoothed complexity see Section 5.

the latter ones. In section 4 we show an overview of the proof of the $\mathcal{PLS}$-completeness of computing a local optimum of Max-Cut on graphs with maximum degree five by reducing from the $\mathcal{PLS}$-complete problem CircuitFlip. In a nutshell, we map instances of CircuitFlip to graphs whose nodes of degree greater than five are comparing. Some parts of the graphs are adjustments of subgraphs of the $\mathcal{PLS}$-completeness proof of [21]. Then, using our technique, we show that local optima for these graphs induce local optima in the corresponding instances of CircuitFlip.

In section 5 we show that on graphs with degree $O(\log n)$ local Max-Cut has probably polynomial smoothed complexity. To obtain this result, we basically prove that every improving step w. r. t. the FLIP-neighborhood increases the cut by at least a polynomial value in n and/or σ, with high probability.

2 Preliminaries

A graph G together with a 2-partition P of V is denoted by G_P. We let $c_{G_P} : V \to \{0,1\}$ with $c_{G_P}(u) = 1$ if and only if $u \in V_1$ in G_P. We let $c_{G_P}(u)$ be the **color** of u in G_P, where u is white if $c_{G_P}(u) = 0$ and black otherwise. If the considered graph is clear from the context then we also just write $c_P(v)$ and if even the partition is clear then we omit the whole subscript. For convenience we treat the colors of the nodes also as truth values, i. e. black corresponds to *true* and white to *false*. For a vector v of nodes we let $c(v)$ be the vector of colors induced by c. We say that an edge $\{u, v\}$ is **in the cut** in P if $c_P(u) \neq c_P(v)$. For a node u we say that u **flips** if it changes the partition. A node u is **happy** in G_P if a flip of u does not increase the weight of the cut, and **unhappy** otherwise. Since we consider weighted graphs, we also say that a flip increases the cut if it increases the weight of the cut. A partition P is a **local optimum** if all nodes in G_P are happy.

A local search problem Π consists of a set of instances $\mathcal{I}$, a set of feasible solutions $\mathcal{F}(I)$ for every instance $I \in \mathcal{I}$, and an objective function $f : \mathcal{F}(I) \to \mathbb{Z}$. In addition, every solution $s \in \mathcal{F}(I)$ has a neighborhood $\mathcal{N}(s, I) \subseteq \mathcal{F}(I)$. For an instance $I \in \mathcal{I}$, the problem is to find a solution $s \in \mathcal{F}(I)$ such that for all $s' \in \mathcal{N}(s, I)$ solution s' does not have a greater value than s with respect to f in case of maximization and not a lower value in case of minimization.

A local search problem Π is in the class $\boldsymbol{\mathcal{PLS}}$ [9] if the following three polynomial time algorithms exist: algorithm A computes for every instance $I \in \mathcal{I}$ a feasible solution $s \in \mathcal{F}(I)$, algorithm B computes for every $I \in \mathcal{I}$ and $s \in \mathcal{F}(I)$ the value $f(s)$, and algorithm C returns for every $I \in \mathcal{I}$ and $s \in \mathcal{F}(I)$ a better neighbor solution $s' \in \mathcal{N}(s, I)$ if there is one and "locally optimal" otherwise. A problem $\Pi \in \mathcal{PLS}$ is **$\boldsymbol{\mathcal{PLS}}$-reducible** to a problem $\Pi' \in \mathcal{PLS}$ if there are the following polynomial time computable functions Φ and Ψ. The function Φ maps instances I of Π to instances of Π' and Ψ maps pairs (s, I), where s is a solution of $\Phi(I)$, to solutions of I, such that for all instances I of Π and local optima s^* of $\Phi(I)$ the solution $\Psi(s^*, I)$ is a local optimum of I. Finally, a problem $\Pi \in \mathcal{PLS}$ is **$\boldsymbol{\mathcal{PLS}}$-complete** if every problem in $\mathcal{PLS}$ is $\mathcal{PLS}$-reducible to Π.

In our technique, as well as in the $\mathcal{PLS}$-completeness proof, we make use of a result of Monien and Tscheuschner [14]. They showed a property for a set of graphs containing two certain types of nodes of degree four.

Since we do not need their types in this paper, we omit the restrictions on the nodes and use the following weaker proposition.

Lemma 1 ([14]). *Let C_f be a boolean circuit with N gates which computes a function $f : \{0,1\}^n \to \{0,1\}^m$. Then, using $O(logN)$ space, one can compute a graph $G_f = (V_f, E_f)$ with maximum degree four containing nodes $s_1, \ldots, s_n$, $t_1, \ldots, t_m \in V_f$ of degree one such that for the vectors $s := (s_1, \ldots, s_n), t := (t_1, \ldots, t_n)$ we have $f(c_P(s)) = c_P(t)$ in every local optimum P of G_f.*

Definition 1. *For a polynomial time computable function f we say that $G_f = (V_f, E_f)$ as constructed in Lemma 1 is the graph that **looks** at the input nodes $s_i \in V_f$ and **biases** the output nodes $t_i \in V_f$ to have the colors induced by f.*

Usage of Lemma 1. Notice first that G_f can be constructed in logarithmic space and thus polynomial time for any polynomial time computable function f. In the rest of the paper we use the graph G_f for several functions f and we will scale the weights of its edges. Then, the edges of G_f give incentives of appropriate weight to certain nodes of those graphs to which we add G_f. The incentives bias the nodes to take the colors induced by f. We already point out that for any node v we will introduce at most one subgraph that biases v. Moreover, the unique edge $e = \{u, v\}$ incident to a biased node v that is an edge of the subgraph that biases v will in many cases have the lowest weight among the edges incident to v. In particular, the weight of e will then be chosen small enough such that the color of v, in local optima, depends on the color of u if and only if v is indifferent with respect to the colors of the other nodes adjacent to v. Note that in local optima the node u has the opposite color as the color to which v is biased according to f.

3 Substituting Certain Nodes of Unbounded Degree

Definition 2. *Let $G = (V, E)$ be a graph. A node $v \in V$ is called **comparing** if there is an $m \in \mathbb{N}$ such that*

(i) v is adjacent to exactly $2m+1$ nodes $u_1^1, u_1^2, u_2^1, u_2^2, \ldots, u_m^1, u_m^2, u \in V \setminus \{v\}$ with edge weights $a_1, \ldots, a_m, \delta$, as shown in Figure 1,

(ii) u is a node of a subgraph $G' = (V', E')$ of G that looks at a subset of $V \setminus \{u, v\}$ and biases v,

(iii) $a_i \geq 2a_{i+1}$ for all $1 \leq i < m$ and $a_m \geq 2\delta$.

*The subgraph G' is called the **biaser** of v. For u_i^j with $1 \leq i \leq m$, $1 \leq j \leq 2$ we call the node u_i^k with $1 \leq k \leq 2$ and $k \neq j$ adjacent to v via the unique edge with the same weight as $\{u_i^j, v\}$ the **counterpart** of u_i^j with respect to v.*

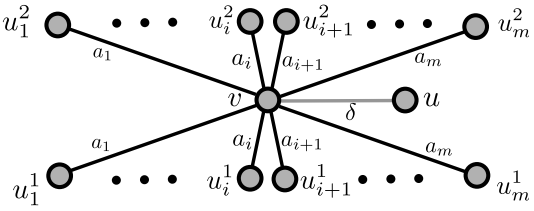


Fig. 1. Node v is a comparing node

The name of the comparing node stems from its behaviour in local optima. If we treat the colors of the neighbors $u_1^1, \ldots, u_m^1$ of v as a binary number a, with u_1^1 being the most significant bit, and the colors of $u_1^2, \ldots, u_m^2$ as the bitwise complement of a binary number b then, in a local optimum, the comparing node v is white if $a > b$, it is black if $a < b$, and if $a = b$ then v has the color to which it is biased by its biaser. In this way, the color of v "compares" a and b in local optima.

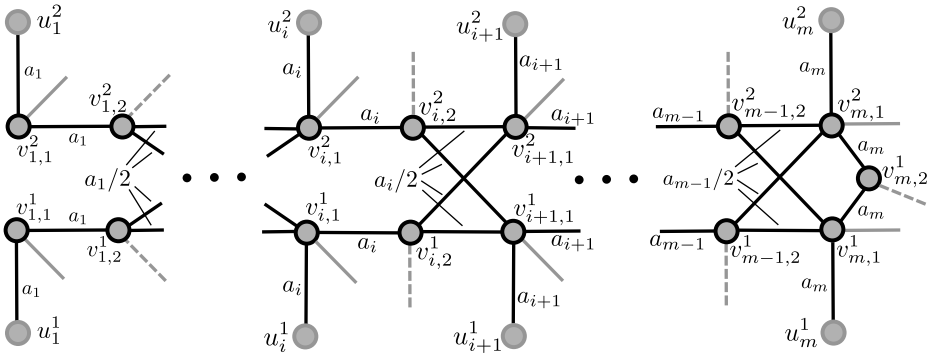


Fig. 2. The gadget that substitutes a comparing node v

In the following, we let $G = (V, E)$ be a graph and $v \in V$ be a comparing node with adjacent nodes and incident edges as in Figure 1. We say that we **degrade** v if we remove v and its incident edges and add the following nodes and edges. We introduce nodes $v_{i,j}^k$ for $1 \leq i < m, 1 \leq j \leq 2, 1 \leq k \leq 2$, nodes $v_{m,1}^k$ for $1 \leq k \leq 2$, and $v_{m,2}^1$ with edges and weights as depicted in Figure 2 – the nodes u_i^j in Figure 2 have gray circumcircles to indicate that they, in contrast to the other nodes, also occur in G. Furthermore, we add a subgraph G'' that looks at u and biases all nodes $v_{i,1}^k$ to the opposite of the color of u (this is illustrated by short gray edges in Figure 2) and the nodes $v_{i,2}^k$ to the color of u (short gray dashed edges). The weights of the edges of G'' are scaled such that each of them is strictly smaller than δ. Note that due to the scaling the color of the unique node of G'' adjacent to u does not affect the happiness of u – node

u is therefore not depicted in Figure 2 anymore. We let $\boldsymbol{G(G,v)}$ be the graph obtained from G by degrading v and we call v **weakly indifferent** in a partition P if $c_P(u_i^1) \neq c_P(u_i^2)$ for all $1 \leq i \leq m$. If v is not weakly indifferent then we call the two nodes u_i^1, u_i^2 adjacent to v via the edges with highest weight for which $c_P(u_i^1) = c_P(u_i^2)$ the **decisive neighbors** of v in P. We let $V_{com} \subseteq V$ be the set of comparing nodes of V, and for a partition P of the nodes of $G(G, v)$ we let $\boldsymbol{col_P} : V_{com} \to \{0, 1\}$ be the partial function defined by

$$col_P(v) = \begin{cases} 0, \text{ if for all } i, j : c_P(v_{i,1}^j) = 0 \text{ and } c_P(v_{i,2}^j) = 1, \\ 1, \text{ if for all } i, j : c_P(v_{i,1}^j) = 1 \text{ and } c_P(v_{i,2}^j) = 0. \end{cases}$$

We say that a comparing node v has the color $\kappa \in \{0, 1\}$ in a partition P if $col_P(v) = \kappa$.

Theorem 1. *Let $G = (V, E)$ be a graph, $v \in V$ a comparing node, its adjacent nodes and incident edges as in Figure 1, P be a local optimum of G such that in P the biaser of v biases v to $c_P(v)$, i. e. $c_P(u) \neq c_P(v)$. Let P' be a partition of the nodes of $G(G, v)$ such that $c_P(w) = c_{P'}(w)$ for all $w \in V \setminus \{v\}$. Then, P' is a local optimum if and only if $c_P(v) = col_{P'}(v)$.*

Note the restriction that in the local optimum P the biaser of v biases v to the color that v in fact has in P and *not* to the opposite. In the $\mathcal{PLS}$-completeness proof in section 4 the biaser of any comparing node v is designed to bias v to the color that v has in a local optimum due the colors of its neighbors. Then, we can use Theorem 1 to argue about the color of v.

Proof. Let $\kappa \in \{0, 1\}$ be the color to which v is biased by its biaser in P, i. e. $\kappa := c_P(v)$. For all i, j we call the color of $v_{i,1}^j$ **correct** if $c_{P'}(v_{i,1}^j) = \kappa$ and we call the color of $v_{i,2}^j$ correct if $c_{P'}(v_{i,2}^j) = \overline{\kappa}$. Moreover, we call $v_{i,j}^k$ correct for any i, j, k if it has its correct color.

"$\Rightarrow$": Let P' be a local optimum. Note that each node $v_{i,j}^k$ is biased by an edge with weight lower than δ to its correct color. Therefore, to show that it is correct in the local optimum P', it suffices to show that it gains at least half of the sum of weights of the incident edges with weight greater than δ if it is correct. We prove the Theorem by means of the following Lemmas which are each proven via straightforward inductive arguments.

Lemma 2. *Let $q \leq m$ and $c_P(u_i^1) = \overline{\kappa}$ for all $i \leq q$. Then, $v_{i,1}^1$ and $v_{i,2}^1$ are correct for all $i \leq q$.*

Lemma 3. *Let $q \leq m$, node $v_{i,1}^1$ and $v_{i,2}^1$ be correct for all $i \leq q$, and $v_{q,1}^2$ be correct. Then, $v_{i,1}^2$ and $v_{i,2}^2$ are correct for all $i < q$.*

Lemma 4. *Let $q \leq m$. If $v_{q,1}^1$ and $v_{q,1}^2$ are correct then $v_{i,j}^k$ is correct for any j, k, and $q \leq i \leq m$.*

We first consider the case that v is weakly indifferent. Then, for each i at least one of the nodes u_i^1 and u_i^2 has the color κ. Due to the symmetry between the

nodes $v^1_{i,j}$ and $v^2_{i,j}$ we may assume w. l. o. g. that $c_P(u^1_i) = \overline{\kappa}$ for all i. Then, Lemma 2 implies that $v^1_{i,1}$ and $v^1_{i,2}$ are correct for all i. Then, the correctness of $v^1_{m,2}$ and $v^1_{m-1,2}$ together imply the correctness of $v^2_{m,1}$. Then, Lemma 3 implies the correctness of $v^2_{i,1}$ and $v^2_{i,2}$ for all $i < m$.

Now assume that v is not weakly indifferent and let u^1_q and u^2_q be the decisive neighbors of v. As in the previous case we assume w. l. o. g. that $c_P(u^1_i) = \overline{\kappa}$ for all $i \leq q$. Then, due to Lemma 2 node $v^1_{i,1}$ and $v^1_{i,2}$ are correct for all $i \leq q$. If $q = 1$ then $c(u^2_1) = \overline{\kappa}$ implies the correctness of $v^2_{1,1}$ – recall that by assumption v is biased to the opposite color of the color of the decisive nodes. On the other hand, if $q > 1$ then the correctness of $v^1_{q-1,2}$ and $c(u^2_q) = \overline{\kappa}$ together imply the correctness of $v^2_{q,1}$. Then, Lemma 3 implies the correctness of $v^2_{i,1}$ and $v^2_{i,2}$ for all $i < q$. Finally, Lemma 4 implies the correctness of $v^k_{i,j}$ for all j, k, and $q \leq i \leq m$.

"$\Leftarrow$": Assume, that every node $v^k_{i,j}$ is correct. As we have seen in "$\Rightarrow$" $v^k_{i,j}$ is happy then. Moreover, each u^j_i is also happy since its neighbors have the same colors as in the local optimum P – recall that if $v^j_{i,1}$ is correct it has the same color in P' as v in P. The colors of the remaining nodes are unchanged. Therefore, P' is a local optimum. This finishes the proof of Theorem 1. □

4 Proof of $\mathcal{PLS}$-Completeness

Our reduction bases on the following $\mathcal{PLS}$-complete problem CircuitFlip (in [9] it is called Flip, which we avoid in this paper since the neighborhood of Max-Cut has the same name).

Definition 3 ([9]). *An instance of* **CircuitFlip** *is a boolean circuit C with n input bits and m output bits. A feasible solution of* CircuitFlip *is a vector $v \in \{0,1\}^n$ of input bits for C and the value of a solution is the output of C treated as a binary number. Two solutions are neighbors if they differ in exactly one bit. The objective is to maximize the output of C.*

Theorem 2. *The problem of computing a local optimum of the* Max-Cut *problem on graphs with maximum degree five is $\mathcal{PLS}$-complete.*

Proof Sketch: Let C be a boolean circuit with n inputs. W. l. o. g. we make the following two assumptions. First, C only consists of NOR-gates $G_N, \ldots, G_1$, where $G_m, \ldots, G_1$ return the output. Second, the gates $G_{m+n}, \ldots, G_{m+1}$ return an improving solution for the given input if there is one and return the input if it is a local optimum. From C we construct a graph G_C consisting of two isomorphic subgraphs G^0_C, G^1_C representing copies of C – the overall structure of our proof is inspired by [10]. For each gate G_i in C there is a subgraph S^κ_i for $\kappa \in \{0,1\}$ in G_C. The subgraphs S^κ_i are taken from [21] and adjusted such that they have maximum degree five without changing local optima. In particular, each S^κ_i contains a comparing node g^κ_i whose color represents the output of G_i. To maintain a maximum degree of five we assume that g^κ_i is degraded in G_C and argue via Theorem 1 about its color in local optima. Then, the colors

of the nodes of S_i^κ, in local optima, either behave as a NOR-gate or have a **reset** state, i. e. a state in which each input node of S_i^κ is indifferent w. r. t. its neighbors in S_i^κ. For each $\kappa \in \{0,1\}$ we have a subgraph T^κ that looks at g_i^κ for $m+1 \leq i \leq m+n$, i. e. at the improving solution, and biases each input node of $G_C^{\overline{\kappa}}$ to the color of its corresponding g_i^κ. Finally, we have a subgraph that looks at the input nodes of G_C^0, G_C^1, decides whose input results in a greater output w. r. t. C – this subgraph is called **winner** as opposed to the **loser** which is the other subgraph – and biases the subgraphs S_i^κ of the winner to behave like NOR-gates and the subgraphs of the loser to take the reset state.

Then, we show that the colors of the subgraphs S_i^κ of the winner in fact reflect the correct outputs w. r. t their inputs and that the input nodes of the loser in fact are indifferent w. r. t. their neighbors in the subgraphs S_i^κ. Then, due to the bias of T^κ, the input nodes of the loser take the colors of the improving neighbor computed by the winner whereafter the loser becomes the new winner. Hence, the improving solutions switch back and forth between the two copies until the colors of the input nodes of both copies are local optima and the copies return their input as improving solution. Then, the colors of the input nodes induce a local optimum of C. □

5 Smoothed Complexity of Local Max-Cut

We consider the smoothed complexity of local Max-Cut for graphs with degree $O(\log n)$. Smoothed analysis, as introduced by Spielman and Teng [19], is motivated by the observation that practical data is often subject to some small random noise. Formally, let $\Omega_{n,m}$ be the set of all weighted graphs with n vertices and m edges, in which each graph has maximum degree $O(\log n)$. In this paper, if A is an algorithm on graphs with maximum degree $O(\log n)$, then the smoothed complexity of A with σ-Gaussian perturbation is

$$Smoothed_A^\sigma(n) = \max_m \max_{G \in \Omega_{n,m}} \mathbb{E}_{\mathrm{x}_m}[T_A(G^{w_{\max} \cdot \mathrm{x}_m})],$$

where $\mathrm{x}_m = (x_1, \ldots, x_m)$ is a vector of length m, in which each entry is an independent Gaussian random variable of standard deviation σ and mean 0. $\mathbb{E}_{\mathrm{x}_m}$ indicates that the expectation is taken over vectors x_m according to the distribution described before, $T_A(G)$ is the running time of A on G, and $G^{w_{\max} \cdot \mathrm{x}_m}$ is the graph obtained from G by adding $w_{\max} \cdot x_i$ to the weight of the i-th edge in G, where $w_{\max}$ is the largest weight in the graph. We assume that the edges are considered according to some arbitrary but fixed ordering.

According to Spielman and Teng, an algorithm A has polynomial smoothed complexity if there exist positive constants c', n_0, σ_0, k_1, and k_2 such that for all $n > n_0$ and $0 \leq \sigma < \sigma_0$ we have

$$Smoothed_A^\sigma(n) < c' n^{k_1} \cdot \sigma^{-k_2}.$$

In this paper, we use a relaxation of polynomial smoothed complexity [20], which builds up on Blum and Dungan [3] (see also Beier and Vöcking [4]). According to

this relaxation, an algorithm A has probably polynomial smoothed complexity if there exist positive constants c', n_0, σ_0, and α such that for all $n > n_0$ and $0 \leq \sigma < \sigma_0$ we have

$$\max_m \max_{G \in \Omega_{n,m}} \mathbb{E}_{\mathrm{X}_m}[T_A^\alpha(G^{w_{\max} \cdot \mathrm{X}_m})] < \frac{c'n}{\sigma}.$$

Theorem 3. *Let A be some FLIP local search algorithm for local* Max-Cut. *Then, A has probably polynomial smoothed complexity on any graph with maximum degree $O(\log n)$.*

Proof. Let $V = \{v_1, \ldots, v_n\}$, and denote by d_i the degree of v_i. Furthermore, let $w_{i,j}$ be the weight of egde (v_i, v_j). Let $m = |E|$, and $\mathrm{x}_m = (x_1, \ldots, x_m)$ a vector of Gaussian random variables of standard deviation σ and mean 0. Alternatively, we denote by $x_{i,j}$ the Gaussian random variable which perturbates edge (v_i, v_j), i. e., $\tilde{w}_{i,j} = w_{i,j} + w_{\max} \cdot x_{i,j}$ represents the weight of (v_i, v_j) in the perturbated graph $G^{w_{\max} \cdot \mathrm{X}_m}$.

In the following, G is an arbitrary graph in $\Omega_{n,m}$ where $m = O(n \log n)$. We show that for any $\delta \in (0, 1)$ there are constants c', n_0, σ_0, k_1, and k_2 such that for all $n > n_0$ and $0 \leq \sigma < \sigma_0$ we obtain

$$Pr_{\mathrm{X}_m}[T_A(G^{w_{\max} \cdot \mathrm{X}_m}) < \delta^{-2} c' n^{k_1} \cdot \sigma^{-k_2}] > 1 - \delta. \tag{1}$$

Then, (1) implies the statement of the theorem (cf. [4]).

In order to show the inequality above, we make use of the fact that the sum of k Gaussian random variables with variance σ^2 and mean 0 is a Gaussian random variable with variance $k\sigma^2$ and mean 0. Let $X_1, \ldots, X_k$ be k Gaussian random variables with variance σ^2 and mean 0. Furthermore, let a be some real number, and $S \subset \{1, \ldots, k\}$. Then, for some large constant c and any $\delta' \in (0, 1)$ we have

$$Pr\left[\left|\sum_{j \in S} X_j - \sum_{j \notin S} X_j - a\right| \leq \frac{\delta' \sigma}{c \cdot 2^k}\right] \leq \delta' \cdot 2^{-k}. \tag{2}$$

In order to show the theorem, we normalize the weights by setting the largest weight to 1, and dividing all other weights by $w_{\max}$. That is, we obtain some graph G' with weights $w'_{i,j} = w_{i,j}/w_{\max}$. The edge weights of G' are perturbated accordingly by Gaussian random variables with variance σ^2 and mean 0. Clearly, $T_A(G') = T_A(G)$ and $Pr[T_A(G'^{\mathrm{X}_m}) < \delta^{-1} c' n^{k_1} \cdot \sigma^{-k_2}] = Pr[T_A(G^{w_{\max} \cdot \mathrm{X}_m}) < \delta^{-1} c' n^{k_1} \cdot \sigma^{-k_2}]$. Therefore, we consider G' instead of G in the rest of the proof.

In the next step, we show that for an arbitrary but fixed partition P of G' and node v_i, flipping v_i increases (or decreases) the cut by $\Omega\left(\frac{\delta\sigma}{n 2^{d_i}}\right)$, with probability $1 - \delta/2 \cdot n^{-1} 2^{-d_i}$. This is easily obtained from Equation (2) in the following way. Define S' to be the set of the neighbors of v_i, which are in the same partition as v_i according to P. Let $e_1, \ldots, e_{d_i}$ be the edges incident to v_i, and denote by $w_1, \ldots, w_{d_i}$ the weights of these edges in G'. We assume w. l. o. g. that $e_1, \ldots, e_{|S'|}$ have both ends in the same partition as v_i, and $S = \{1, \ldots, |S'|\}$.

Furthermore, let $a = \sum_{j \notin S} w_j - \sum_{j \in S} w_j$, $k = d_i$, and $\delta' = \delta/(2n)$. Applying now Equation (2) we obtain the desired result.

For a node v_i, there are at most $\sum_{i=0}^{d_i} \binom{d_i}{i} = 2^{d_i}$ possibilities to partition the edges into two parts, one subset in the same partition as v_i and the other subset in the other partition. Therefore, by applying the union bound we conclude that any flip of an unhappy v_i increases the cut by $\Omega\left(\frac{\delta\sigma}{n2^{d_i}}\right)$, with probability at least $1 - \delta/2 \cdot n^{-1}$. Since there are n nodes in total, we may apply the union bound again, and obtain that every flip (carried out by some unhappy node) increases the cut by $\Omega\left(\frac{\delta\sigma}{n2^{d_i}}\right)$, with probability at least $1 - \delta/2$. Since $d_i = O(\log n)$ and the largest weight in G' is 1, we conclude that the largest cut in G' may have weight $O(n \log n)$. Furthermore, for each i we have $|x_i| \leq l\sqrt{\ln n}$ with probability $1 - O(n^{-l})$ whenever l is large enough (remember that $\sigma < 1$). Let A_1 be the event that there is some x_i with $|x_i| = \omega(\log n)$, and A_2 is the event that there is a node v_i and a partition P such that flipping v_i increases the cut by at most $\tau\left(\frac{\delta\sigma}{n2^{d_i}}\right)$, where τ is a very small constant. We know that $Pr[A_1] = n^{-\omega(1)}$ and $Pr[A_2] < \delta/2$. Thus, as long as $\delta = n^{-O(1)}$, the total number of steps needed by A is at most

$$T_A(G'^{\mathbf{x}_m}) = O\left(\frac{n^2 \log^2 n 2^{d_i}}{\delta\sigma}\right) = \frac{n^{O(1)}}{\delta\sigma}$$

with probability $1 - (Pr[A_1] + Pr[A_2]) > 1 - \delta$. The case $\delta = n^{-\Omega(1)}$ is omitted due to space limitations. □

6 Conclusion and Open Problems

In this paper, we introduced a technique by which we can substitute graphs with certain nodes of unbounded degree, namely so called comparing nodes, by graphs with nodes of maximum degree five such that local optima of the former graphs induce unique local optima of the latter ones. Using this technique, we show that the problem of computing a local optimum of the Max-Cut problem is $\mathcal{PLS}$-complete even on graphs with maximum degree five. We do not show that our $\mathcal{PLS}$-reduction is tight, but the tightness of our reduction would not result in the typical knowledge gain anyway since the properties that come along with the tightness of $\mathcal{PLS}$-reductions, namely the $\mathcal{PSPACE}$-completeness of the standard algorithm problem and the existence of instances that are exponentially many improving steps away from any local optimum, are already known for the maximum degree four [14]. The obvious remaining question is to ask for the complexity of local Max-Cut on graphs with maximum degree four. Is it in $\mathcal{P}$? Is it $\mathcal{PLS}$-complete? Another important question is whether local Max-Cut has in general probably polynomial smoothed complexity. Unfortunately, the methods used so far seem not to be applicable to show that in graphs with super-logarithmic degree the local Max-Cut problem has probably polynomial smoothed complexity (cf. also [17]).

Acknowledgement. We thank Dominic Dumrauf, Martin Gairing, Martina Hüllmann, Burkhard Monien, and Rahul Savani for helpful suggestions.

References

1. Arthur, D., Manthey, B., Röglin, H.: k-Means has polynomial smoothed complexity. In: FOCS 2009, pp. 405–414 (2009)
2. Ackermann, H., Röglin, H., Vöcking, B.: On the impact of combinatorial structure on congestion games. Journal of the ACM (JACM) 55(6), art. 25 (2008)
3. Blum, A., Dunagan, J.: Smoothed analysis of the perceptron algorithm for linear programming. In: SODA, pp. 905–914 (2002)
4. Beier, R., Vöcking, B.: Typical properties of winners and losers in discrete optimization. In: STOC, pp. 343–352 (2004)
5. Englert, M., Röglin, H., Vöcking, B.: Worst case and probabilistic analysis of the 2-Opt algorithm for the TSP. In: SODA, pp. 1295–1304 (2006)
6. Fabrikant, A., Papadimitriou, C.H., Talwar, K.: The complexity of pure Nash Equilibria. In: STOC, pp. 604–612 (2004)
7. Gairing, M., Savani, R.: Computing stable outcomes in hedonic games. In: Kontogiannis, S., Koutsoupias, E., Spirakis, P.G. (eds.) Algorithmic Game Theory. LNCS, vol. 6386, pp. 174–185. Springer, Heidelberg (2010)
8. Garey, M.R., Johnson, D.S.: Computers and intractability, a guide to the theory of NP-completeness. Freeman, New York (1979)
9. Johnson, D.S., Papadimitriou, C.H., Yannakakis, M.: How easy is local search? Journal of Computer and System Sciences 37(1), 79–100 (1988)
10. Krentel, M.W.: Structure in locally optimal solutions. In: FOCS, pp. 216–221 (1989)
11. Kelner, J.A., Nikolova, E.: On the hardness and smoothed complexity of quasi-concave minimization. In: FOCS, pp. 472–482 (2007)
12. Loebl, M.: Efficient maximal cubic graph cuts. In: Leach Albert, J., Monien, B., Rodríguez-Artalejo, M. (eds.) ICALP 1991. LNCS, vol. 510, pp. 351–362. Springer, Heidelberg (1991)
13. Monien, B., Dumrauf, D., Tscheuschner, T.: Local search: Simple, successful, but sometimes sluggish. In: Abramsky, S., Gavoille, C., Kirchner, C., Meyer auf der Heide, F., Spirakis, P.G. (eds.) ICALP 2010. LNCS, vol. 6198, pp. 1–17. Springer, Heidelberg (2010)
14. Monien, B., Tscheuschner, T.: On the power of nodes of degree four in the local max-cut problem. In: Calamoneri, T., Diaz, J. (eds.) CIAC 2010. LNCS, vol. 6078, pp. 264–275. Springer, Heidelberg (2010)
15. Poljak, S.: Integer linear programs and local search for max-cut. SIAM Journal on Computing 21(3), 450–465 (1995)
16. Poljak, S., Tuza, Z.: Maximum cuts and largest bipartite subgraphs. Combinatorial Optimization, pp. 181–244. American Mathematical Society, Providence (1995)
17. Röglin, H.: Personal communication (2010)
18. Röglin, H., Vöcking, B.: Smoothed analysis of integer programming. Math. Program. 110(1), 21–56 (2007)
19. Spielmann, D., Teng, S.-H.: Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. Journal of the ACM (JACM) 51(3), 385–463 (2004)
20. Spielmann, D., Teng, S.-H.: Smoothed analysis: an attempt to explain the behavior of algorithms in practice. Commun. ACM 52(10), 76–84 (2009)
21. Schäffer, A.A., Yannakakis, M.: Simple local search problems that are hard to solve. SIAM Journal on Computing 20(1), 56–87 (1991)
22. Vershynin, R.: Beyond Hirsch Conjecture: Walks on Random Polytopes and Smoothed Complexity of the Simplex Method. In: FOCS, pp. 133–142 (2006)

Clique Clustering Yields a PTAS for max-Coloring Interval Graphs

Tim Nonner

IBM Research - Zurich
tno@zurich.ibm.com

Abstract. We are given an interval graph $G = (V, E)$ where each interval $I \in V$ has a weight $w_I \in \mathbb{R}^+$. The goal is to color the intervals V with an arbitrary number of color classes $C_1, C_2, \ldots, C_k$ such that $\sum_{i=1}^{k} \max_{I \in C_i} w_I$ is minimized. This problem, called max-coloring interval graphs, contains the classical problem of coloring interval graphs as a special case for uniform weights, and it arises in many practical scenarios such as memory management. Pemmaraju, Raman, and Varadarajan showed that max-coloring interval graphs is NP-hard (SODA'04) and presented a 2-approximation algorithm. Closing a gap which has been open for years, we settle the approximation complexity of this problem by giving a polynomial-time approximation scheme (PTAS), that is, we show that there is an $(1+\epsilon)$-approximation algorithm for any $\epsilon > 0$. Besides using standard preprocessing techniques such as geometric rounding and shifting, our main building block is a general technique for trading the overlap structure of an interval graph for accuracy, which we call clique clustering.

1 Introduction

Coloring a given graph $G = (V, E)$ is a classical NP-hard problem in combinatorial optimization. One reason why graph coloring has been studied so extensively is the fact that many practical problems in scheduling and planning can be formulated in such a way. The arguably simplest example is that the nodes V represent tasks which need to be partitioned into *color classes* of pairwise non-conflicting tasks, where a conflict between two tasks is indicated by an edge in E connecting them. All tasks in one color class may share a common resource, and hence minimizing the number of color classes also minimizes the number of needed resources. It is natural to assume that each task requires a resource during a given time interval, and thus two tasks conflict if their time intervals intersect. This results in an *interval graph*, and, in this case, we may think of the nodes in V as intervals, of color classes as sets of pairwise disjoint intervals, and of cliques as sets of intervals with non-empty intersection. It is folklore that an optimal coloring of a given interval graph can be found in polynomial time using the *first-fit strategy*: sort the intervals according to their left endpoints, and then iteratively assign colors according to this ordering. For example, in the sample instance in Figure 1, where I_2 overlaps exactly with I_1 and I_3 and I_3 overlaps

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 183–194, 2011.

exactly with I_2 and I_4, this gives the color classes $\{I_1, I_3\}$ and $\{I_2, I_4\}$. In constrast, finding an optimal coloring of a general graph is hard to approximate even within $n^{1-\epsilon}$ for any $\epsilon > 0$, unless NP $\subseteq$ ZPP [7].

However, coloring interval graphs fails to express non-uniform resource requirements. For instance, a resource might be a buffer and the tasks memory requests of different size that need to be buffered during a given time interval [18]. In this case, a buffer used by different non-conflicting requests needs to be large enough to hold any such request. We can model this extension by assigning a weight $w_I \in \mathbb{R}^+$ to each interval $I \in V$ that represents the size of the corresponding request, and hence the buffer assigned to a color class C of requests needs to have at least size $\max_{I \in C} w_I$. Thus, finding an optimal coloring in this context is called *max-coloring interval graphs*: partition a given interval graph $G = (V, E)$ into an arbitrary number of color classes $C_1, C_2, \ldots, C_k$ such that $\sum_{i=1}^{k} \max_{I \in C_i} w_I$ is minimized. The classical problem of coloring interval graphs is contained as a special case by using uniform weights.

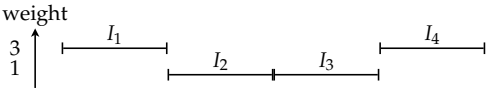


Fig. 1. Sample intervals

Previous work. Unfortunately, the first-fit strategy does not work for max-coloring interval graphs. For example, assume that the y-axis in Figure 1 represents the interval weights. Therefore, $w_{I_1} = w_{I_4} = 3$ and $w_{I_2} = w_{I_3} = 1$. The coloring of the first-fit strategy listed above gives cost $3 + 3 = 6$, whereas an optimal max-coloring with color classes $\{I_1, I_4\}, \{I_2\}$, and $\{I_3\}$ only gives cost $3 + 1 + 1 = 5$. Indeed, it has been shown by Pemmaraju, Raman, and Varadarajan [18] that max-coloring interval graphs is NP-hard. A simplified proof was given by the same authors in [17] by a reduction from coloring circular arc graphs, a superclass of interval graphs. They also conjectured APX-hardness [18], but this conjecture is no longer valid [19]. Finally, they presented an 2-approximation algorithm for max-coloring interval graphs [18], and Pemmaraju and Raman [16] showed later on that any graph class that admits an α-approximation algorithm for coloring also admits a 4α-approximation algorithm for max-coloring. Recall here than an *α-approximation algorithm* yields a solution in polynomial time whose cost is at most α times the cost of an optimal solution. Hence, since it is a well-known fact that perfect graphs, a superclass of interval graphs, can be colored in polynomial time [9], this yields a 4-approximation algorithm for max-coloring perfect graphs. Epstein and Levin [5] improved this factor from 4 to e. On the other hand, after a line of improvements [10,6,11], Kavitha and Mestre [14] presented an algorithm for max-coloring paths, a subclass of interval graphs, which requires only time $\mathcal{O}(n + S(n))$, where $S(n)$ is the time to sort the weights. A *polynomial-time approximation scheme (PTAS)* for trees is

known [2], that is, there is a $(1+\epsilon)$-approximation algorithm for any $\epsilon > 0$, but max-coloring bipartite graphs is APX-hard [4], i.e., there is no PTAS unless P = NP. However, in the context of the original motivation of this problem in buffer management, interval graphs remain the most relevant graph class, since they are easy to describe, but powerful enough to model temporal conflicts.

Contributions and outline. Closing a gap that has been open for years, we settle the approximation complexity of max-coloring interval graphs by presenting a PTAS in Section 3. From the very beginning [13] to more recent celebrated results [1], the arguably most successful scheme to obtain PTASs is to trade the size of the search space for accuracy in order to make it treatable by a dynamic program running in polynomial time. The problem-specific challenge is to do this in a way such that there is still a solution in the restricted search space which is $(1+\epsilon)$-close to an optimal solution. We do this in two steps. In the main step, called *clique clustering*, we argue that we can partition the intervals V into cliques, here called *clusters* to distinguish them from other cliques, such that we are allowed to treat every cluster as a single interval during a dynamic programming procedure. The surprising insight (Lemma 2) is that we are able to find such a partition with the properties that (1) we are only losing an $(1+\epsilon)$-factor in the approximation ratio and (2) the maximum overlap of clusters is logarithmic in the number of intervals n. Note that this is an exponential drop compared to the maximum overlap of intervals, which might be as large as n. Initially, in a minor preprocessing step using the shifting technique of Hochbaum and Maass [12], we trade the number of different interval weights for accuracy such that we may assume that there are only constantly many. Formally, we use the following lemma (proof in full version of this paper). This lemma holds for any graph class, which, to the best of our knowledge, has not been observed before.

Lemma 1. *For any $\epsilon > 0$, by losing an $(1+\epsilon)$-factor in the approximation ratio, we may assume that the number of different interval weights and the ratio between the maximum and the minimum interval weight are both constants.*

Lemma 1 has already been applied by the author [15] to obtain a PTAS for the inverse problem of max-coloring co-interval graphs with *constant capacities*, i.e., there is a constant bound k such that we additionally require that $|C| \leq k$ for any color class C. Note that finding a max-coloring of a co-interval graph G is equivalent to finding a max-clique partition of the co-graph of G, which is again an interval graph. Despite the similarity in name, it is worth mentioning here that max-coloring interval graphs and max-coloring co-interval graphs have a quite different structure. For instance, max-coloring co-interval graphs is polynomially solvable [8,3], and only the capacitated case described above is NP-hard [15]. However, except for this preprocessing step, all arguments in this paper differ completely from [15]. Bambis et al. [2] gave some results for the max-coloring problem with capacities.

2 Preliminaries

Some intervals $I \in V$ might have the same weight w_I. Let then $w_1 < w_2 < \ldots < w_m$ with $m \leq n := |V|$ be an ordering of the different interval weights $\{w_I \mid I \in V\}$. Hence, m is the number of different interval weights and w_m/w_1 is the ratio between the maximum and the minimum interval weight. Recall that Lemma 1 allows us to assume that these parameters are both constant. We also refer to an index $1 \leq i \leq m$ as a *level*. Moreover, we refer to the level i_I with $w_{i_I} = w_I$ as the *level* of an interval $I \in V$, and to $i_C := \max_{I \in C} i_I$ as the *level* of a color class $C \subseteq V$. For simplicity, we write *coloring* for max-coloring throughout this paper, and we use σ to denote a coloring. Hence, the goal is to find a coloring σ that minimizes $\sum_{C \in \sigma} w_{i_C} = \sum_{i=1}^{m} w_i \sigma_i$, where $\sigma_i := |\{C \in \sigma \mid i_C = i\}|$ denotes the number of color classes in σ with level i. Assume that each coloring σ additionally defines a level $i_C(\sigma) \geq i_C$ for each color class $C \in \sigma$. Using this, to simplify the arguments to follow, we slightly relax the problem formulation as follows: find a coloring σ such that $\text{cost}(\sigma) := \sum_{C \in \sigma} w_{i_C(\sigma)}$ is minimized. Note that it holds for an optimal coloring σ^* with $\text{cost}(\sigma^*) = \text{OPT}$ that $i_C(\sigma^*) = i_C$ for each color class $C \in \sigma^*$, which shows that this is indeed a relaxation. Finally, define $i_I(\sigma) := i_C(\sigma)$ for any interval $I \in C$ in a color class $C \in \sigma$.

2.1 Overlap Structure

Assume that the endpoints of all intervals are distinct, which can be easily ensured by rearranging the intervals without modifying the overlap structure. Let then $l_I \in \mathbb{R}$ and $r_I \in \mathbb{R}$ denote the left and right endpoint of an interval $I \in V$, respectively. For two intervals $I, I' \in V$, we write $I < I'$ iff $r_I < l_{I'}$, and $I > I'$ iff $l_I > r_{I'}$. On the other hand, we write $I \prec I'$ iff $l_I < l_{I'}$, and $I \succ I'$ iff $r_I > r_{I'}$. Moreover, for two interval sets $B, B' \subseteq V$, we write $B \prec B'$ iff $I \prec I'$ for any pair $I \in B$ and $I' \in B'$, and $B \succ B'$ iff $I \succ I'$ for any pair $I \in B$ and $I' \in B'$.

For a position $t \in \mathbb{R}$ and a set of intervals $B \subseteq V$, let $B(t) := \{I \in B \mid t \in I\}$ denote the subset of intervals in B that overlap with t. Using this, for a set of cliques P, let $P(t) := \{B \in P \mid B(t) \neq \emptyset\}$ denote the subset of cliques in P that *overlap* with t, and define $\omega_P := \max_{t \in \mathbb{R}} |P(t)|$. Recall that a *clique* $B \subseteq V$ is a set of intervals with $\cap_{I \in B} I \neq \emptyset$, and also recall that a *clique partition* P of V is a set of pairwise disjoint cliques with $\cup_{B \in P} B = V$. Finally, define $\omega := \omega_P$ for the trivial clique partition $P = \{\{I\} \mid I \in V\}$. Hence, ω denotes the maximum overlap of all intervals. Observe that the maximum clique size is exactly ω, and this is also the minimum number of colors needed to color all intervals V. We can find such a coloring σ using the *first-fit strategy*: let $I_1, I_2, \ldots, I_n$ be an ordering of the intervals V with $l_{I_1} < l_{I_2} < \ldots < l_{I_n}$. Then, for $s = 1, \ldots, n$, if there is already a color class $C \in \sigma$ with $I < I_s$ for any interval $I \in C$, add I_s to C, and otherwise, add the new color class $C := \{I_s\}$ that only contains I_s to σ. Note that a (max-)coloring with a minimum number of colors is optimal in case of uniform interval weights.

2.2 Respecting Cliques

We say that a coloring σ *respects* a clique $B \subseteq V$ iff $i_I(\sigma) = i_{I'}(\sigma)$ for any pair $I, I' \in B$. Consequently, we may define $i_B(\sigma) := i_I(\sigma)$ for an arbitrary interval $I \in B$ in such a case. We extend this by saying that a coloring σ *respects* a set of pairwise disjoint cliques P iff σ respects all cliques in P. Furthermore, a *partial coloring* σ is a coloring that only partitions a subset of intervals $V' \subseteq V$ into color classes. For all intervals $I \in V \backslash V'$, we define then $i_I(\sigma) := \infty$ to indicate that I is not contained in a color class. Now note that we can turn any function $f : V' \to \{1, 2, \ldots, m\}$ with $f(I) \geq i_I$ for each interval $I \in V'$ into a partial coloring σ^f as follows: for each level i, use the first-fit strategy to compute a coloring σ^i for the intervals $\{I \in V' \mid f(I) = i\}$ with a minimum number of colors. Finally, combine all these colorings to a coloring $\sigma^f := \cup_{i=1}^{m} \sigma^i$ where, for each level i and color class $C \in \sigma^i$, we set $i_C(\sigma) := i$. We say that such a function f *respects* a set of cliques P iff σ^f respects P.

3 PTAS via Dynamic Programming

Recall the interval ordering $I_1, I_2, \ldots, I_n$ from Section 2 with $l_{I_1} < l_{I_2} < \ldots < l_{I_n}$. Moreover, recall that, for any $1 \leq s \leq n$, $V(l_{I_s})$ denotes the set of all intervals which overlap with the left endpoint of I_s, and observe that $V(l_{I_s}) \subseteq \{I_1, I_2, \ldots, I_s\}$.

3.1 DP Array

Consider a dynamic programming array Π with boolean entries of the form $\Pi(s, f, g)$, where $1 \leq s \leq n$ is an integer, $f : V(l_{I_s}) \to \{1, 2, \ldots, m\}$ with $f(I) \geq i_I$ for each interval $I \in V(l_{I_s})$, and $g : \{1, 2, \ldots, m\} \to \{0, 1, \ldots, n\}$. We want to fill this array such that $\Pi(s, f, g)$ is true if and only if there is a function $\overline{f} : \{I_1, I_2, \ldots, I_s\} \to \{1, 2, \ldots, m\}$ with

(1) $\overline{f}(I) \geq i_I$ for each interval $I \in \{I_1, I_2, \ldots, I_s\}$,
(2) $\overline{f}(I) = f(I)$ for each interval $I \in V(l_{I_s})$,
(3) $\sigma_i^{\overline{f}} = g(i)$ for each level i.

In words, $\overline{f}$ extends f from $V(l_{I_s})$ to $\{I_1, I_2, \ldots, I_s\}$ and, for each level i, g counts the number of color classes in $\sigma^{\overline{f}}$ with level i. As soon as Π is filled correctly, we can find an optimal coloring by enumerating all colorings σ that realize an entry of the form $\Pi(n, f, g)$ in order to find one that minimizes $\text{cost}(\sigma) = \sum_{i=1}^{m} w_i g(i)$. To see this, note that $\text{cost}(\sigma^{\overline{f}}) = \text{OPT}$ for the function $\overline{f}$ with $\overline{f}(I) = i_I(\sigma^*)$ for each interval $I \in \{I_1, I_2, \ldots, I_n\} = V$.

3.2 Recurrence Relation

For some integer s, assume that we have already successfully filled all entries of the form $\Pi(s-1, f, g)$. Now consider an entry of the form $\Pi(s, f, g)$, and let $i' := f(I_s)$. There are exactly two possibilities why $\Pi(s, f, g)$ could be true.

Case 1: There is a true entry $\Pi(s-1, f', g')$ with
(1) $f'(I) = f(I)$ for each interval $I \in V(l_{I_{s-1}}) \cap V(l_{I_s})$,
(2) $g'(i) = g(i)$ for each level i,
(3) $|\{I \in V(l_{I_{s-1}}) \cap V(l_{I_s}) \mid f'(I) = i'\}| < g'(i')$.
Case 2: There is a true entry $\Pi(s-1, f', g')$ with
(1) $f'(I) = f(I)$ for each interval $I \in V(l_{I_{s-1}}) \cap V(l_{I_s})$,
(2) $g'(i') = g(i') - 1$ and $g'(i) = g(i)$ for each level $i \neq i'$,
(3) $|\{I \in V(l_{I_{s-1}}) \cap V(l_{I_s}) \mid f'(I) = i'\}| = g'(i')$.

These two cases correspond to the two cases in the first-fit strategy. Specifically, to see why Case 1 causes $\Pi(s, f, g)$ to be true, consider a partial coloring σ that realizes the true entry $\Pi(s-1, f', g')$ from Case 1. Note that it is possible to add I_s to an already existing color class $C \in \sigma$ with $i_C(\sigma) = i'$ if and only if $g'(i')$, the number of such color classes that already exist, is strictly larger than $|\{I \in V(l_{I_{s-1}}) \cap V(l_{I_s}) \mid f'(I) = i'\}|$, the number of intervals which compete with I_s for a place in such a color class. In this case, we may hence set $g(i') = g'(i')$. On the other hand, Case 2 considers the opposite case where we need to add another color class, and thus we have $g(i') = g'(i') + 1$. Consequently, combining both cases yields a recurrence relation which can be implemented in polynomial time given that the size of Π is polynomial. Observe that it is easy to initially fill all entries of the form $\Pi(1, f, g)$, since then we only need to consider the single interval I_1.

3.3 Approximate DP

By Lemma 1, we may assume that m, the number of levels, and w_m/w_1, the ratio between the maximum and minimal interval weight, are both constants. However, since we only have the upper bound $V(l_{I_s}) \leq \omega$ for each integer $1 \leq s \leq n$, the size of Π could be as large as $n \cdot m^{\omega} \cdot (n+1)^m$, which is not polynomial for $\omega = \Omega(n)$, even for a constant m. Therefore, we need to restrict the set of functions f to consider. To this end, we use the following critical lemma which trades the size of ω_P for accuracy (proof in Section 4).

Lemma 2. *Given that m and w_m/w_1 are constants, for any $\epsilon > 0$, we can compute a clique partition P of V in polynomial time such that there exists a coloring σ with the following properties: (1) σ respects P, (2) $\text{cost}(\sigma) \leq (1 + \mathcal{O}(\epsilon))\text{OPT}$, and (3) $\omega_P = \mathcal{O}(\log(n))$.*

Note that setting $P = \{\{I\} \mid I \in V\}$ does clearly satisfy Property (2) in Lemma 2. However, we do not gain anything in this case, since then still $\omega_P = \omega$. The astonishing fact about Lemma 2 is that by losing an arbitrary small factor in accuracy, we immediately get an exponential drop in ω_P. This allows us to prove our final result.

Theorem 1. *There is a PTAS for max-coloring interval graphs.*

Proof. Let P be a clique partition as proposed by Lemma 2. Note that restricting the search space to functions f that respect P decreases the size of Π to $n \cdot$

$m^{\omega_P} \cdot (n+1)^m$, which is polynomial due to Property (3) in Lemma 2 if m is constant. To see this, note that, for any integer $1 \leq s \leq n$, there are at most m^{ω_P} many functions $f : V(I_s) \to \{1, 2, \ldots, m\}$ that respect P. This restricted dynamic program yields an optimal coloring σ subject to the constraint that σ respects P. However, by Properties (1) and (2), we still obtain the upper bound $\text{cost}(\sigma) \leq (1 + \mathcal{O}(\epsilon))\text{OPT}$. The claim of the theorem follows. □

4 Proof of Lemma 2

We assume throughout this section that m and w_m/w_1 are both constants. Consider then an arbitrary small but fixed $\epsilon > 0$, and assume that $1/\epsilon$ is integral. Assume moreover that $\log(n)$ is integral[1]. In order to construct the claimed clique partition P, we proceed in two steps. First, in Subsection 4.1, we introduce the notion of a hierarchical clique partition P_V which already satisfies Property (3) in Lemma 2. However, this clique partition is not fine-grained enough to also satisfy Property (2). Therefore, in Subsection 4.2, we show how to additionally partition each clique $A \in P_V$, yielding P. We refer to the cliques in P as *clusters*, and these clusters will have size $\kappa := \epsilon^2\omega/\log(n)$. Moreover, define $\hat{\kappa} := \kappa/\epsilon = \epsilon\omega/\log(n)$.

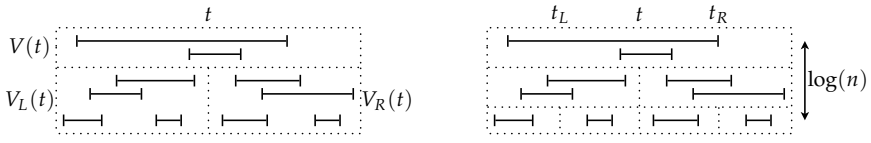
Lemma 3. *We may assume that κ and $\hat{\kappa}$ are both integral.*

Proof. If $\omega < \log(n)/\epsilon^3$, then the claim of Lemma 2 is trivially satisfied, since we can then simply use $P = V$. Therefore, we may assume that $\omega \geq \log(n)/\epsilon^3$. Consequently, it suffices to add at most $\epsilon\omega$ many dummy intervals with the lowest weight w_1 and the form $[-\infty, +\infty]$, i.e., intervals that overlap with all other intervals, in order to increase ω such that this parameter is a multiple of the integer $\log(n)/\epsilon^2$. Because of the trivial lower bound $\text{OPT} \geq w_1\omega$, observe that this increases OPT by at most $\epsilon \cdot \text{OPT}$. Consequently, we may assume that κ is integral, and hence also $\hat{\kappa}$. The claim follows. □

4.1 Hierarchical Clique Partition

Before constructing P_V, we first construct an intermediate clique partition P'_V. To this end, observe that any position $t \in \mathbb{R}$ partitions V into a clique $V(t)$, a left part $V_L(t) := \{I \in V \mid I < t\}$, and a right part $V_R(t) := \{I \in V \mid I > t\}$, as depicted in Subfigure 2(a). Using two positions $t_L, t_R \in \mathbb{R}$ with $t_L < t < t_R$, we can then partition $V_L(t)$ and $V_R(t)$ in the same way, and so on. This gives us a binary tree whose nodes are cliques in V, as depicted in Subfigure 2(b). Now note that there is always a position that splits an interval set into two parts of at most half the size. Therefore, we can choose the positions during this process such that the resulting binary tree has logarithmic depth $\log(n)$, as schematically depicted in Subfigure 2(b). Let P'_V be the clique partition of V containing all cliques constructed during this process, namely $V(t), V(t_L)\backslash V(t), V(t_R)\backslash V(t), \ldots$. For instance, the clique partition depicted in Subfigure 2(b) consists of 7 cliques.

[1] All logarithms have base 2.



(a) the first partition (b) one possible final result

Fig. 2. The intermediate partition P'_V

Note that $|P'_V(t)| \leq \log(n)$ for any position $t \in \mathbb{R}$. To finally obtain P_V, we further partition each cliques $A \in P'_V$ into the cliques $A_1, A_2, \ldots, A_m$, where, for each level i, $A_i := \{I \in A \mid i_I = i\}$ are the intervals in A with level i. Hence, $P_V := \{A_i \mid A \in P'_V, 1 \leq i \leq m\}$. For each clique $A \in P_V$, we may hence define $i_A := i_I$ for an arbitrary interval $I \in A$. We refer to P_V as a *hierarchical clique partition*, and we immediatly obtain the following simple but critical observation.

Observation 2 *For any position $t \in \mathbb{R}$, $|P_V(t)| \leq m \log(n)$.*

The following lemma will simplify some arguments.

Lemma 4. *For each clique $A \in P_V$, we may assume that $\hat{\kappa}$ divides $|A|$.*

Proof. For each clique $A \in P_V$, let $t_A \in \mathbb{R}$ be the position with $t_A \in \cap_{I \in A} I$ from the construction of P'_V, i.e., the position used to construct the clique in P'_V whose partition resulted in A. Hence, there are exactly m cliques $A \in P_V$ with the same position t_A. Observe now that, for each clique $A \in P_V$, we need to add at most $\hat{\kappa}$ many dummy intervals $I = [t_A, t_A]$ with $i_I = i_A$ to A in order to ensure that $\hat{\kappa}$ divides $|A|$. Since these intervals are disjoint for two cliques $A, A' \in P_V$ with $t_A \neq t_{A'}$, we find that we always need at most $m\hat{\kappa}$ additional color classes C with $w_C \leq w_m$ to partition these intervals. Therefore, because of the trivial lower bound $\mathrm{OPT} \geq w_1 \omega$ and the fact that w_m/w_1 is constant, this increases OPT by at most $w_m m\hat{\kappa} = \mathcal{O}(\epsilon)\mathrm{OPT}$, which proves the claim. □

4.2 Clique Clustering for a Single Clique

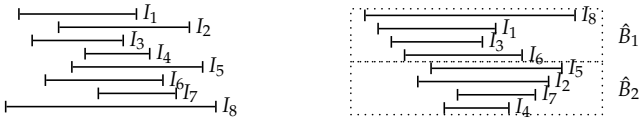
We partition a clique $A \in P_V$ into clusters in two steps.

Step 1: First, we inductively partition A into cliques, called *superclusters*, as follows: let $\hat{B}_1, \hat{B}_2, \ldots, \hat{B}_s$ be the so far generated superclusters of A. Now, if still $\cup_{j=1}^{s} \hat{B}_j \neq A$, then generate another supercluster $\hat{B}_{s+1}$ by distinguishing two cases: if $s+1$ is odd, then $\hat{B}_{s+1}$ contains the $\hat{\kappa}$ intervals in $A \setminus \cup_{j=1}^{s} \hat{B}_j$ with leftmost left endpoints, and otherwise, if $s+1$ is even, then $\hat{B}_{s+1}$ contains the $\hat{\kappa}$ intervals in $A \setminus \cup_{j=1}^{s} \hat{B}_j$ with rightmost right endpoints. Recall here that we assume that all endpoints of intervals are distinct, and hence we never have to

break ties. Moreover, recall the assumption from Lemma 4 that $\hat{\kappa}$ divides $|A|$. Note that $\hat{B}_1 \prec \hat{B}_3 \prec \hat{B}_5 \prec \ldots$ and $\hat{B}_2 \succ \hat{B}_4 \succ \hat{B}_6 \succ \ldots$.

Step 2: Next, we apply a similar scheme to inductively partition each supercluster $\hat{B}_j$ of A into cliques, called *clusters*, as follows: let $B_1, B_2, \ldots, B_s$ be the so far generated clusters of $\hat{B}_j$. If still $\cup_{l=1}^{s} B_l \neq \hat{B}_j$, then generate another cluster B_{s+1} by distinguishing two cases: if j is odd, then B_{s+1} contains the κ intervals in $\hat{B}_j \setminus \cup_{l=1}^{s} B_l$ with rightmost right endpoints, and otherwise, if j is even, then B_{s+1} contains the κ intervals in $\hat{B}_j \setminus \cup_{l=1}^{s} B_l$ with leftmost left endpoints. Note that each supercluster contains exactly $1/\epsilon$ clusters. Moreover, if j is odd, then $B_1 \succ B_2 \succ B_3 \succ \ldots \succ B_{1/\epsilon}$, and if j is even, then $B_1 \prec B_2 \prec B_3 \prec \ldots \prec B_{1/\epsilon}$.

Example. Assume that A consists of the intervals depicted in Subfigure 3(a) and that $\hat{\kappa} = 4$ and $\kappa = 2$. In this case, as depicted in Subfigure 3(b), the first supercluster $\hat{B}_1$ consists of the intervals I_8, I_1, I_3, I_6, since these are the intervals with the leftmost left endpoints. The next supercluster $\hat{B}_2$ then consists of the intervals I_5, I_2, I_7, I_4, since these are the intervals with rightmost right endpoints in $A \setminus \hat{B}_1$. The clusters of $\hat{B}_1$ are then $B_1 = \{I_8, I_6\}$ and $B_2 = \{I_1, I_3\}$, and the clusters of $\hat{B}_2$ are $B_1 = \{I_2, I_5\}$ and $B_2 = \{I_4, I_7\}$.



(a) clique A (b) superclusters $\hat{B}_1, \hat{B}_2$

Fig. 3. Clustering a clique $A \in P_V$

Let then $\hat{P}_A$ and P_A be the sets of all superclusters and clusters generated by the procedure above, respectively. Therefore, $\hat{P}_A$ and P_A are both clique partitions of A, but P_A is more fine-grained. We obtain the following lemma.

Lemma 5. *For each position $t \in \mathbb{R}$, $|\hat{P}_A(t)| \leq 2(1 + |A(t)|/\hat{\kappa})$.*

Proof. Consider an arbitrary $t' \in \cap_{I \in A} I$, and let $\hat{B}_1, \hat{B}_2, \ldots, \hat{B}_s$ be the superclusters in $\hat{P}_A$ in the order they have been created. Moreover, let j be the maximal index such that still $\hat{B}_j \in \hat{P}_A(t)$. We distinguish two cases:

Case $t' \leq t$: Consider an arbitrary supercluster $\hat{B}_{j'}$ with $j' < j$ and j' even. Hence, we have that $\hat{B}_{j'} \succ \hat{B}_j$, since we picked some rightmost intervals with respect to their right endpoints to form $\hat{B}_{j'}$. Consequently, since there is at least one interval in $\hat{B}_j$ that overlaps with t, we find that all intervals in $\hat{B}_{j'}$ must overlap with t, and thus $\hat{B}_{j'} \subseteq A(t)$.

Case $t' > t$: The same arguments as in the last case show that $\hat{B}_{j'} \subseteq A(t)$ for each supercluster $\hat{B}_{j'}$ with $j' < j$ and j' odd.

Combining both cases gives that that there are $\lceil (j-2)/2 \rceil$ many superclusters $\hat{B}_{j'}$ with $j' < j$ and $\hat{B}_{j'} \subseteq A(t)$. Therefore, since each supercluster has size $\hat{\kappa}$, we obtain $|A(t)| \geq \hat{\kappa}(j-2)/2$. On the other hand, since j is maximal, we have that $|\hat{P}_A(t)| \leq j$. The claim follows by combining these facts. □

This gives us the following lemma (proof in full version of this paper).

Lemma 6. *Consider some clique $A \in P_V$. Then, for any partial coloring σ with $i_I(\sigma) < \infty$ for all intervals $I \in A$, there is a partial coloring σ' with the following properties: (1) σ' respects P_A, (2) $\text{cost}(\sigma') \leq \text{cost}(\sigma)$, (3) $i_I(\sigma) = i_I(\sigma')$ for each interval $I \in V \backslash A$, and (4) for each position $t \in \mathbb{R}$, $|\{B \in P_A(t) \mid i_B(\sigma') = \infty\}| \leq 2m(1/\epsilon + 1 + |A(t)|/\hat{\kappa})$.*

In words, Lemma 6 says that we may replace any coloring σ by a coloring σ' that is identical for all intervals not contained in A, respects all clusters in P_A, but is still not 'much more' partial than σ, as quantified in Property (4). Now we are finally ready to define the claimed clique partition P for Lemma 4 as $P := \cup_{A \in P_V} P_A$.

Lemma 7. *For each position $t \in \mathbb{R}$, $|P(t)| = \mathcal{O}(\log(n))$.*

Proof. Observe that

$$\begin{aligned}
|P(t)| &= \sum_{A \in P_V} |P_A(t)| = \sum_{A \in P_V : A(t) \neq \emptyset} |P_A(t)| \leq \frac{1}{\epsilon} \sum_{A \in P_V : A(t) \neq \emptyset} |\hat{P}_A(t)| \\
&\leq \frac{2}{\epsilon} \left(\sum_{A \in P_V : A(t) \neq \emptyset} 1 + \frac{1}{\hat{\kappa}} \sum_{A \in P_V : A(t) \neq \emptyset} |A(t)| \right) \\
&\leq \frac{2}{\epsilon} \left(m \log(n) + \frac{\omega}{\hat{\kappa}} \right) = \frac{2}{\epsilon} \left(m \log(n) + \frac{\log(n)}{\epsilon} \right) = \mathcal{O}(\log(n)),
\end{aligned}$$

which proves the claim. The inequality in the first line is due to the fact that each supercluster contains exactly $1/\epsilon$ clusters. Moreover, the second line is due to Lemma 5 and a simple rearrangement, and the third line is due to Observation 2. □

Lemma 8. *There is a non-partial coloring σ with $\text{cost}(\sigma) \leq (1 + \mathcal{O}(\epsilon))\text{OPT}$ that respects P.*

Proof. Consider some optimal non-partial coloring σ^* for V with $\text{cost}(\sigma^*) = \text{OPT}$. Now, for each clique $A \in P_V$, iteratively apply Lemma 6 in order to transform σ^* into a partial coloring σ that respects P_A for each clique $A \in P_V$. Consequently, σ also respects $P = \cup_{A \in P_V} P_A$. This iterative application is possible because of Property (3) in Lemma 6, since this property ensures that the levels of the intervals outside of A are not modified. Moreover, Property (1) ensures that finally $\text{cost}(\sigma) \leq \text{OPT}$. However, the coloring σ computed in this way is partial, i.e., there are some intervals $I \in V$ with $i_I(\sigma) = \infty$. Let $V^\infty :=$

$\{I \in V \mid i_I(\sigma) = \infty\}$ be the subinstance of these intervals, and let σ^∞ be a coloring for these intervals with a minimum number of colors using the first-fit strategy. We can then turn σ into a non-partial coloring by adding all color classes $C \in \sigma^\infty$ to σ with $i_C(\sigma) := m$. This setting of $i_C(\sigma)$ is feasible, since m is the maximal level, and this does also not affect the fact that σ respects P.

To bound the cost increase of σ due to the additional color classes from σ^∞, note that the number of color classes in σ^∞ is the maximum clique size in V^∞, which is the maximal overlap of intervals $\max_{t\in\mathbb{R}} |V^\infty(t)|$. However, for any position $t \in \mathbb{R}$, we obtain the bound

$$\begin{aligned}
|V^\infty(t)| &= \sum_{A\in P_V} |\{I \in A(t) \mid i_I(\sigma) = \infty\}| = \sum_{A\in P_V : A(t)\neq\emptyset} |\{I \in A(t) \mid i_I(\sigma) = \infty\}| \\
&= \kappa \sum_{A\in P_V : A(t)\neq\emptyset} |\{B \in P_A(t) \mid i_B(\sigma) = \infty\}| \\
&\leq \kappa 2m \left(\sum_{A\in P_V : A(t)\neq 0} \left(\frac{1}{\epsilon} + 1\right) + \frac{1}{\hat{\kappa}} \sum_{A\in P_V : A(t)\neq 0} |A(t)| \right) \\
&\leq \kappa 2m \left(\frac{2m\log(n)}{\epsilon} + \frac{\omega}{\hat{\kappa}} \right) = \epsilon(4m^2 + 2m)\omega.
\end{aligned}$$

The second line is due to the fact that each cluster contains exactly κ intervals. The third line is due to Property (4) in Lemma 6 and a simple rearrangment. Moreover, the inequality in the fourth line is due to Observation 2. Therefore, the cost of the additional color classes from σ^∞ is at most $w_m\epsilon(4m^2 + 2m)\omega = \mathcal{O}(\epsilon)\mathrm{OPT}$, since w_m/w_1 is constant and we have the trivial lower bound $\mathrm{OPT} \geq w_1\omega$. We conclude that $\mathrm{cost}(\sigma) \leq (1 + \mathcal{O}(\epsilon))\mathrm{OPT}$, which proves the claim. □

Proof (Lemma 2). Combine Lemmas 7 and 8. □

References

1. Arora, S.: Polynomial time approximation schemes for euclidean traveling salesman and other geometric problems. Journal of the ACM 45(5), 753–782 (1998)
2. Bampis, E., Kononov, A., Lucarelli, G., Milis, I.: Bounded max-colorings of graphs. In: Cheong, O., Chwa, K.-Y., Park, K. (eds.) ISAAC 2010. LNCS, vol. 6506, pp. 353–365. Springer, Heidelberg (2010)
3. Becchetti, L., Korteweg, P., Marchetti-Spaccamela, A., Skutella, M., Stougie, L., Vitaletti, A.: Latency constrained aggregation in sensor networks. In: Azar, Y., Erlebach, T. (eds.) ESA 2006. LNCS, vol. 4168, pp. 88–99. Springer, Heidelberg (2006)
4. de Werra, D., Demange, M., Escoffier, B., Monnot, J., Paschos, V.T.: Weighted coloring on planar, bipartite and split graphs: Complexity and approximation. Discrete Applied Mathematics 157(4), 819–832 (2009)
5. Epstein, L., Levin, A.: On the max coloring problem. In: Kaklamanis, C., Skutella, M. (eds.) WAOA 2007. LNCS, vol. 4927, pp. 142–155. Springer, Heidelberg (2008)
6. Escoffier, B., Monnot, J., Paschos, V.T.: Weighted coloring: further complexity and approximability results. Inf. Process. Lett. 97(3), 98–103 (2006)

7. Feige, U., Kilian, J.: Zero knowledge and the chromatic number. J. Comput. Syst. Sci. 57(2), 187–199 (1998)
8. Finke, G., Jost, V., Queyranne, M., Sebö, A.: Batch processing with interval graph compatibilities between tasks. Discrete Applied Mathematics 156(5), 556–568 (2008)
9. Gröschel, M., Lovász, L., Schrijver, A.: Geometric Algorithms and Combinatorial Optimization. Springer, Heidelberg (1988)
10. Guan, D.J., Zhu, X.: A coloring problem for weighted graphs. Inf. Process. Lett. 61(2), 77–81 (1997)
11. Halldórsson, M.M., Shachnai, H.: Batch coloring flat graphs and thin. In: Gudmundsson, J. (ed.) SWAT 2008. LNCS, vol. SWAT, pp. 198–209. Springer, Heidelberg (2008)
12. Hochbaum, D.S., Maass, W.: Approximation schemes for covering and packing problems in image processing and VLSI. Journal of the ACM 32(1), 130–136 (1985)
13. Ibarra, O.H., Kim, C.E.: Fast approximation algorithms for the knapsack and sum of subset problems. J. ACM 22, 463–468 (1975)
14. Kavitha, T., Mestre, J.: Max-coloring paths: Tight bounds and extensions. In: Dong, Y., Du, D.-Z., Ibarra, O. (eds.) ISAAC 2009. LNCS, vol. 5878, pp. 87–96. Springer, Heidelberg (2009)
15. Nonner, T.: Capacitated max -batching with interval graph compatibilities. In: Kaplan, H. (ed.) SWAT 2010. LNCS, vol. 6139, pp. 176–187. Springer, Heidelberg (2010)
16. Pemmaraju, S.V., Raman, R.: Approximation algorithms for the max-coloring problem. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 1064–1075. Springer, Heidelberg (2005)
17. Pemmaraju, S.V., Raman, R., Varadarajan, K.: Max-coloring and online coloring with bandwidths on interval graphs. ACM Transactions on Algorithms (accepted for publication)
18. Pemmaraju, S.V., Raman, R., Varadarajan, K.: Buffer minimization using max-coloring. In: Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2004), pp. 562–571 (2004)
19. Raman, R.: Personal communication (2010)

On Variants of File Caching*

Leah Epstein[1], Csanád Imreh[2,**], Asaf Levin[3], and Judit Nagy-György[4]

[1] Department of Mathematics, University of Haifa, 31905 Haifa, Israel
lea@math.haifa.ac.il
[2] Department of Informatics, University of Szeged, 6720 Szeged, Hungary
cimreh@inf.u-szeged.hu
[3] Faculty of Industrial Engineering and Management,
The Technion, 32000 Haifa, Israel
levinas@ie.technion.ac.il
[4] Department of Mathematics, University of Szeged, Aradi Vértanúk tere 1,
H-6720 Szeged, Hungary
Nagy-Gyorgy@math.u-szeged.hu.

Abstract. In the file caching problem, the input is a sequence of requests for files out of a slow memory. A file has two attributes, a retrieval cost and an integer size. It is required to maintain a cache of size k, bringing each file, which is not present in the cache at the time of request, from the slow memory into the cache. This incurs a cost equal to the retrieval cost of the file. Well-known special cases include paging (all costs and sizes are equal to 1), the cost model which is also known as weighted paging (all sizes are equal to 1), the fault model (all costs are equal to 1) and the bit model (the cost of a file is equal to its size).

We study two online variants of the problem, *caching with bypassing* and *caching with rejection*. If bypassing is allowed, a miss for a file still results in an access to this file in the slow memory, but its subsequent insertion into the cache is optional. In the model with rejection, together with each request for a file, the algorithm is informed with a rejection penalty of the request. When a file which is not present in the cache is requested, the algorithm must either bring the file into the cache, paying the retrieval cost of the file, or reject the file, paying the rejection penalty of the request. The goal function is the sum of total rejection penalty and the total retrieval cost.

We design deterministic and randomized algorithms for both problems. The competitive ratios of these randomized algorithms match the best known results for caching. In the deterministic case, it is known that a $(k+1)$-competitive algorithm for caching with bypassing exists, and this is best possible. In contrast, we present a lower bound of $2k+1$ on the competitive ratio of any deterministic algorithm for the variant with rejection, which holds already for paging. We design a $(2k+2)$-competitive algorithm for caching with rejection, and a different $(2k+1)$-competitive algorithm , which is applicable for paging, the bit model and the cost model.

* This research was partially supported by the TÁMOP-4.2.2/08/1/2008-0008 program of the Hungarian National Development Agency.
** Supported by the Bolyai Scholarship of the Hungarian Academy of Sciences.

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 195–206, 2011.

1 Introduction

Paging and caching problems [26,11,20,28] are fundamental optimization problems, with multiple applications such as operating systems and the Internet. Such problems deal with page replacement policies in two-level memory systems, which consist of a small and fast memory, typically referred to as *cache*, and a large and slow main memory. Both memory levels are partitioned into slots of size 1, which can accommodate pages. The cache consists of k such slots. The general problem, which is called *file caching*, is defined as follows. Files are bundles of pages. A file f has two attributes, which are its (integer) size $size(f)$ (number of pages), and its cost $cost(f)$ (the cost of accessing it in the main memory). In most applications, this cost is proportional to the time which is required to access it, while the cost of accessing a file residing in the cache is seen as zero, since this time is negligible compared to the time required to access the main memory.

The input is a sequence of requests for files. On a request for a file which the algorithm stored in the cache, the algorithm does nothing. However, a request for a file f which is not present in the cache requires an access to the main memory, resulting in a cost of $cost(f)$ for the algorithm. This situation is called a *miss*. In the standard model (also called the *forced* model), the algorithm must insert the file into the cache, possibly evicting other files to create sufficient space for the requested file. An additional model was studied, called caching with bypassing, or the *Optional* model [2,13], where the algorithm must read the requested file f from the main memory for a cost of $cost(f)$, but it does not necessarily needs to insert it into the cache, though it may do so.

A number of important special cases were studied. The first study was the *paging* variant. In this case a file is referred to as a *page*. For every page f, $size(f) = cost(f) = 1$. That is, the sequence consists of single pages, and the system is uniform in the sense that reading any page from the main memory results in the same cost. Three common models which generalize paging and are special cases of caching are the *bit model*, the *fault model* and *cost model* (also called weighted paging) (see [2]). In the first two models, files may have arbitrary sizes, while in the third model, all files have size 1 (but arbitrary costs). In the bit model, each file f has $cost(f) = size(f)$, while in the fault model, each file f satisfies $cost(f) = 1$. That is, the fault model assumes that the cost of accessing the main memory multiple times is determined by the number of accesses, while the bit model assumes that what determines the cost is the total size of files read from the main memory.

We study online variants of the problem and use competitive analysis. For caching problems, it is common to use the asymptotic competitive ratio. For an online problem, we let OPT denote an optimal offline algorithm which knows the entire input. For an algorithm $\mathcal{A}$, we let $\mathcal{A}(I)$ (or $\mathcal{A}$, if I is clear from the context) denote the cost of $\mathcal{A}$ on an input I. An online algorithm ALG is $\mathcal{R}$-competitive, if there exists a constant c (independent of I) which satisfies $\text{ALG}(I) \leq \mathcal{R} \cdot \text{OPT}(I) + c$ for any input I. If $\mathcal{R}$ is the smallest value for which this inequality can be satisfied for all inputs and some value c, then $\mathcal{R}$ is the

competitive ratio of ALG. For a randomized algorithm ALG we have a similar definition where $\text{ALG}(I)$ is replaced with its expected value $E(\text{ALG}(I))$.

In this paper, we initiate the study of caching with rejection. In this variant of caching, each miss can be treated in two ways. The first way is the traditional way, where the file is brought from the main memory. The second one is by declining the request, notifying the user that the request cannot be carried out, and paying a rejection penalty. In this variant, if the i-th request is for file f, then this request is a pair (f, r_i), where r_i is the rejection penalty of this request. Note that r_i is not a property of f, and the rejection penalty of one file can differ for different requests for this file. The cost of an algorithm is the sum of the total rejection penalty of rejected requests, and the total cost of served requests. Caching with rejection generalizes caching, since the latter is the special case where the rejection penalty is infinite. Caching with rejection also generalizes caching with bypassing as the later problem is the special case where the rejection penalty for every request (f, r_i) satisfies $r_i = cost(f)$. Similarly, for each one of the specific models, we can define caching (or paging) with rejection in this model, which generalizes caching in this model with or without bypassing.

Previous caching models assume that every request must be fulfilled. This does not describe many of the applications properly, since in practice, requests are often declined. In operating systems, when the system is busy running other processes and as a result a request made by a user to open a file is denied. In web caching, the situation where some requests are not served is often encountered. Browsers deny requests for web pages which are hard to contact at that time.

The offline paging problem is polynomially solvable [8], and caching in the cost model can be solved using network flow methods [11]. We note that these techniques can be extended for the cost model with bypassing or with rejection. The other variants are strongly NP-hard [12] with or without bypassing, implying NP-hardness for the cases with rejection. Though caching problems come from real-time applications, offline approximation algorithms are of interest [20,2,6].

Notation. Throughout the paper, we assume that a cache is empty at the beginning of the input. For a set of files S, let $size(S) = \sum_{g \in S} size(g)$. Let n denote the number of files in the slow memory. An input for caching with bypassing can be represented in two alternative ways which are used interchangeably in the paper. The first option is a sequence of pairs, which are file names and arrival times of the requests. The second option is a sequence of file names, and indices. A pair (f, j) corresponds to the j-th request of file f in the sequence. Similarly, an input for caching with rejection can be represented in two ways, and the only difference is the additional attribute of each request, which is the rejection penalty of this request. For an input I, which consists of a list of requests for one of the variants, we use $\text{OPT}_s(I)$ (or OPT_s) to denote the cost of an optimal offline algorithm which does not use bypassing and similarly $\text{OPT}_b(I)$ (or OPT_b) denotes this cost if bypassing is allowed.

Previous work. Sleator and Tarjan [26] were the first to consider online paging and showed that the two natural paging algorithms, namely, LRU and FIFO have

competitive ratios of at most k, which is best possible for any deterministic online algorithm (see also [22]). A class of algorithms, called MARKING ALGORITHMS, was later shown by Karlin et al. [21] to achieve the same bound.

The study of randomized algorithms for paging was started by Fiat et. al [18], where an algorithm, called *Randomized Marking* was shown to perform much better than the deterministic algorithms, and its competitive ratio is at most $2H_k$, where H_k is the k-th harmonic number (later it was shown [1] that its tight competitive ratio is $2H_k - 1$). A lower bound of H_k on the competitive ratio of any randomized algorithm was shown in [18] as well, which makes the order of growth of the competitive ratio logarithmic in k. Several different algorithms of competitive ratio H_k are known [23,1].

For the variant with bypassing, the results for paging are similar. A simple reduction shows that for paging, $\text{OPT}_b(I) \leq \text{OPT}_s(I) \leq 2 \cdot \text{OPT}_b(I)$. This reduction does not hold for any of the more general variants (see below). Thus, in the deterministic case, the best competitive ratio is $\Theta(k)$ and in the randomized case, the best competitive ratio is $\Theta(\log k)$. In fact, as mentioned in [20], the best deterministic competitive ratio is $k + 1$, and it is achieved by the same algorithms that give the upper bound of k for paging without bypassing.

Once these results were known, it was an intriguing question to find which generalizations of paging allow similar results. In the deterministic variant, the tight bound of k on the competitive ratio holds for all generalizations without bypassing. The result for weighted paging was shown by Chrobak et al. [11] (see also [27]). The result for caching was proved by Young [28] (see also [10]). If bypassing is allowed, Irani [20] mentioned that LRU has a competitive ratio of $k+1$ for the bit and fault models. Cohen and Kaplan [13] adapted the LANDLORD algorithm of Young [28] for the case with bypassing and showed that it is $(k+1)$-competitive. Thus, for both variants, with and without bypassing, the general variant of caching admits the same competitive ratio as the basic paging problem.

On the other hand, for the randomized variant, the progress was slower. Irani [20] designed algorithms of competitive ratio $O(\log^2 k)$ for the bit model and the fault model, which are valid both with and without bypassing. Recently, Bansal, Buchbinder, and Naor, applying a novel usage of the online primal-dual method, designed an algorithm of competitive ratio $O(\log k)$ for weighted paging [4], and algorithms with similar upper bounds on the competitive ratio for the bit model and the fault model [5]. For the most general case, the upper bound of [5] is $O(\log^2 k)$. The papers [4,5] exploit clever techniques, however these techniques fail to work in the case with bypassing (see below).

In *finely competitive paging*, an optimal offline algorithm can either serve a new request by fetching its page into the cache, or rent (reject, in our terminology) it for a cost of $\frac{1}{r}$, where $r \geq 1$ is a given parameter. The online algorithm which is compared to this offline algorithm cannot rent the page and must fetch it into the cache. Blum et al. [9] showed an $O(r + \log k)$-competitive algorithm for this case. Bansal et al. [3] improved this result and showed a $r + O(\log k)$-competitive algorithm for the case where the online algorithm is allowed to rent the page for a unit cost (whereas the offline algorithm can rent it for a cost of $\frac{1}{r}$). Note that

in our model (unlike finely competitive paging), the cost of not serving a request is the same for both the offline algorithm and the online algorithm, and that for $r = 1$ the result of [3] is for paging with bypassing.

Many online problems were studied with rejection. This includes scheduling [7,25,17,24], bin packing [14,15] and coloring [16]. Problems with rejection are frequently studied in offline scenarios, where they are sometimes called prize-collecting problems. The primal-dual method for obtaining offline approximation algorithms for a problem extends naturally to the corresponding prize-collecting problem (the approximation guarantee may degrade). For example, Goemans and Williamson in their seminal work [19] considered the prize-collecting TSP and the prize collecting Steiner tree problem.

Our results. Recall that the best possible competitive ratio for the deterministic variant of caching with bypassing is $k + 1$ [20,13]. We prove a lower bound of $2k + 1$ on the competitive ratio of any deterministic algorithm for paging with rejection and design two deterministic algorithms. The first one is $(2k+2)$-competitive, and works for the most general setting. Using methods related to the Ski-Rental problem [21], we develop a general reduction of the problem with rejection to the problem with bypassing, showing that an α_k-competitive algorithm for the case of bypassing can be converted (using this algorithm as a black box) into a $(4\alpha_k + 1)$-competitive algorithm for the problem with rejection. Our deterministic algorithm for the general problem with rejection uses this reduction. This algorithm Landlord with rejection (LLR) uses the variant of Landlord with bypassing (LLB). In order to get an upper bound of $2k + 2$ rather than the bound $4k + 5$ implied by the reduction, we compare the cost of the algorithm directly to an optimal algorithm for caching with rejection. The second deterministic algorithm is different and it is valid for paging, the bit model and the cost model. This algorithm has an optimal competitive ratio of $2k + 1$, and it generalizes the Flush when Full (FWF) algorithm.

In addition, we consider randomized algorithms for both variants. We design $O(\log k)$-competitive algorithms for paging, the fault model, the cost model, and the bit model and an $O(\log^2 k)$-competitive algorithm for caching. The results for the variant with rejection use the reduction above. To solve the case with bypassing, we build on the methods of [4,5], however, the ratio between the optimal costs without and with bypassing $\frac{\mathrm{OPT}_s(I)}{\mathrm{OPT}_b(I)}$ can be infinite for caching even for large inputs (this gap is smaller for paging, but linear for the bit model and the fault model). The linear programming formulation of [4,5] cannot be used for the variant with bypassing since it requires having each request in the cache. Instead, we use a linear programming formulation for the variant with bypassing (see [13]), and find a fractional solution for it. In order to be able to use the methods of [4,5] of converting a fractional solution into a randomized one, we modify our fractional solution to the problem with bypassing to a solution of the linear programming formulation without bypassing of [4,5]. To do that, our algorithm decides which requests are bypassed based on the fractional solution to our linear program, and the remaining requests which are not bypassed are those

that are seen to be the input of the linear programming formulation without bypassing.

2 Reducing the Problem with Rejection to the Problem with Bypassing

We present a general reduction from the problem with rejection to the problem with bypassing. In this reduction, we convert an input for the problem with rejection into an input for the problem with bypassing. Assume that ALG is a randomized or deterministic α_k-competitive algorithm for caching with bypassing, we present a randomized or deterministic, respectively, $4\alpha_k + 1$-competitive algorithm for caching with rejection. We denote by F(ALG) the new algorithm which we design for caching with rejection.

Algorithm. F(ALG) For every file g, in the slow memory, maintain a non-negative value $counter(g)$ (which satisfies $counter(g) \leq cost(g)$ at all time, except for possibly the case that g had just been requested and F(ALG) is dealing with this request), which is initialized by zero.

Given a new request for a file f, with the rejection penalty p, if $p > cost(f)$, replace the request with $t(f,p) = \lceil \frac{p}{cost(f)} \rceil$ requests for f, each one with the penalty $\frac{p}{t(f,p)}$. Each of these requests is treated as a separate request below. To give an output for the original input, where there is a single request, instead of $t(f,p)$ identical requests, consider first the case where ALG is deterministic. If the file is inserted into the cache by the algorithm for one of these requests, then the file should be inserted into the cache, and otherwise the request is rejected. Now assume that ALG is randomized, then the algorithm moves to a distribution over the cache states which is the distribution at the end of this subsequence of $t(f,p)$ identical requests.

Given a request for a file f, with a rejection penalty $p \leq cost(f)$, do the following:

1. If the probability of the cache states for which f is in the cache is 1 (or in case ALG is deterministic if f is in the cache), then we do nothing. Otherwise we continue as follows.
2. Let $counter(f) = counter(f) + p$.
3. If $counter(f) < cost(f)$, then do not change the (distribution of the) state of the cache.
4. If $counter(f) \geq cost(f)$, then present f as an input to ALG, and set $counter(f) = 0$.
5. If ALG changes the cache state (or the distribution of the cache states), we change the state (distribution) in the same way.

End of Algorithm F(ALG)
Note that F(ALG) rejects a request at any situation that the requested file is not in the cache.

Theorem 1. *Given an online (deterministic or randomized) algorithm* ALG *for caching with bypassing, with a competitive ratio of at most* α_k*, the online algorithm F(*ALG*) for caching with rejection has a competitive ratio of at most* $4\alpha_k + 1$.

3 Deterministic Algorithms

We prove a lower bound, showing that the deterministic bounds of caching with bypassing cannot be achieved already for paging with rejection.

Theorem 2. *The competitive ratio of any deterministic algorithm for paging with rejection is at least* $2k + 1$.

3.1 Caching with Rejection

We design an algorithm for caching with rejection. Our algorithm LANDLORD WITH REJECTION (LLR) is exactly F(LLB). Using Theorem 1, we get an upper bound of $4k + 5$ on the competitive ratio of LLR. We present a more careful analysis of LLR to show the following.

Theorem 3. *The competitive ratio of* LLR *is at most* $2k + 2$.

3.2 An Improved Algorithm for Several Cases

We design the following phase-based algorithm. We define the algorithm for arbitrarily sized files since we use this algorithm not only for paging, but also for caching in the bit model and fault models.

Algorithm. FLUSH WHEN FULL WITH REJECTION (FWFR)

Maintain a non-negative value $counter(g)$ for every file g in the slow memory, which is initialized by zero. Let λ be the total size of the files in the cache, which is initialized by zero.

Given a request for a file f, (f, r), if f is not in the cache, let $counter(f) = counter(f) + r$ and do the following:

1. If $counter(f) \leq 1$ then reject f.
2. If $counter(f) > 1$, and $size(f) + \lambda \leq k$, then insert f into the cache and let $\lambda = \lambda + size(f)$.
3. If $counter(f) > 1$, and $size(f) + \lambda > k$, then empty the cache and set $\lambda = 0$, for each $g \neq f$ let $counter(g) = 0$, and let $counter(f) = counter(f) - 1$. Go to step 1.

End of Algorithm. FWFR

Theorem 4. FWFR *is* $(2k + 1)$*-competitive for paging, and caching in the bit and in the fault models.*

4 Randomized Algorithms

In this section, we use the online primal-dual method and obtain an $O(\log k)$-competitive algorithm for caching with bypassing in the bit model, in the cost model and in the fault model. We also obtain an $O(\log^2 k)$-competitive algorithm for the weighted caching with bypassing. Using Theorem 1, we will obtain algorithms with these competitive ratios for the problems with rejection. We focus on the problem with bypassing and formulate a linear program which guides our algorithm. Our linear program mimics the linear program of [5,4], with the required modifications to allow bypassing requests.

Instead of charging a file for fetching it into the cache we will charge for evicting files from the cache. Let $x(f,j)$ be an indicator variable for the event that file f is evicted from the cache between the j-th time it is requested and the next time it is requested, or the event that in the j-th time file f is requested it is bypassed. For a file f, denote by $t(f,j)$ the index of the request in which it is requested for the j-th time, and by $r(f,t)$ the number of times it is requested in subsequence of the first t requests. For a time t, we denote by $B(t)$ the set of files which were requested until time t (including time t), and by f_t the file which is requested at time t. Let $\mathcal{S}(t) = \{S \subseteq B(t) | size(S) > k,\ f_t \in S\}$. Let T denote the number of requests in the input. Then, the following integer program formulates our (offline) problem, where each constraint represents the property that at each time, the cache cannot contain files with a total size larger than k:

$$\begin{aligned} \min \quad & \sum_{f=1}^{n} \sum_{j=1}^{r(f,T)} cost(f) \cdot x(f,j) \\ s.t. \quad & \sum_{f \in S} size(f) \cdot x(f, r(f,t)) \geq size(S) - k \ \forall 1 \leq t \leq T,\ \forall S \in \mathcal{S}(t) \\ & x(f,j) \in \{0,1\} \qquad \forall f, j. \end{aligned}$$

We first strengthen the formulation of the linear programming relaxation by replacing $size(f)$ with $\min\{size(S)-k, size(f)\}$ in the constraints. We consider the resulting linear programming relaxation denoted by (LP-bypassing). We say that a set of files S is a *small set at time* t if for all $f \in S$ we have $x(f, r(f,t)) < 1$. In order to ensure feasibility of the primal solution x after the new request of time t arrives, only small sets need to be considered at time t. The dual linear program of (LP-bypassing) has a variable $y(t,S)$ for all time t and $S \in \mathcal{S}(t)$. The dual program denoted as (LP-dual) is as follows.

$$\begin{aligned} \max \quad & \sum_{t} \sum_{S \in \mathcal{S}(t)} (size(S) - k) \cdot y(t,S) \\ s.t. \quad & \sum_{t=t(f,j)}^{t(f,j+1)-1} \sum_{S \in \mathcal{S}(t): f \in S} \min\{size(S) - k, size(f)\} \cdot y(t,S) \leq cost(f)\ \forall f, j \\ & y(t,S) \geq 0 \qquad \forall t, S. \end{aligned}$$

The main difference between our linear program and the linear program of [5] is the fact that in our linear program the current request contributes towards

satisfying the constraints corresponding to the current value of t. This is essential since in our problem the algorithm does not necessarily need to insert f_t to the cache.

Denote the input sequence by I. Each item of I is a pair consisting of a time in $\{1, 2, \ldots, T\}$ and a file in the slow memory. In our algorithm we will identify (in an online deterministic fashion) a subsequence of the requests $I' \subseteq I$ for which the contribution of $x(f, r(f,t))$ to satisfying the constraints of time t is crucial. These requests will be either bypassed or cause the algorithm to evict the files from the cache. We use τ to denote the subset of $\{1, 2, \ldots, T\}$, such that the requests of these times of the subset are not in I'. That is, we do not re-index the requests, but allow the times of the requests to be not necessarily consecutive. The set of the remaining requests $I'' = I \setminus I'$ need to be inserted into the cache, so it will be an input sequence for the caching problem (without bypassing) for which [5] considered the following linear programming formulation denoted as (LP-caching).

$$\min \sum_{f \in I''} \sum_{j=1}^{r_{I''}(f,T)} cost(f) \cdot x(f,j) \qquad s.t.$$

$$\sum_{f \in S \setminus \{f_t\}} \min\{size(S) - k, size(f)\} \cdot x(f, r_{I''}(f,t)) \geq size(S) - k \quad \forall t \in \tau, \; \forall S \in \mathcal{S}_{I''}(t)$$

$$x(f,j) \geq 0 \qquad \forall f, j,$$

where $r_{I''}(f,t)$ and $\mathcal{S}_{I''}(t)$ are the values of $r(f,t)$ and $\mathcal{S}(t)$, respectively, in the instance I''. That is, we consider only the subset of requests $\{1, 2, \ldots, t\} \cap \tau$ in the computation of $r_{I''}(f,t)$ and $B_{I''}(t)$, and we define $\mathcal{S}_{I''}(t) = \{S \subseteq B_{I''}(t) | size(S) > k, \; f_t \in S\}$. Note that we define $r_{I''}(f,t)$ and $\mathcal{S}_{I''}(t)$ only for $t \in \tau$, so in the definition of $\mathcal{S}_{I''}(t)$, $f_t \in B_{I''}(t)$ always holds.

In Section 4 of [5], it is shown that in order to obtain a randomized algorithm whose competitive ratio for the bit and fault models is $O(\log k)$ and for the general case the competitive ratio is $O(\log^2 k)$, it suffices to construct a solution X to (LP-caching) whose cost is bounded by $O(\log k)$ times the cost of a feasible solution to problem (LP-dual). Similarly, by [4], such a solution X will imply an online randomized $O(\log k)$-competitive algorithm for the cost model. Specifically, Bansal, Buchbinder and Naor [4,5] showed that presenting an algorithm which returns a fractional solution which is an $O(\log k)$-competitive is sufficient in order to get the desired competitive ratio for the randomized algorithm (which is fractional with respect to cache states and not specific files).

Our online algorithm constructs a feasible solution x to (LP-bypassing), and a dual feasible solution y to (LP-dual). Moreover, we will identify the set I'' and a feasible solution X to (LP-caching) whose cost is at most twice the cost of x. This will be done in an online fashion. That is, at each time t, a new set of constraints of (LP-bypassing) is revealed to the algorithm. We will increase both dual variables and primal variables until all primal constraints are satisfied. At this stage, if $x(f_t, r(f_t,t)) \geq \frac{1}{2}$ then we reset its value to 1 and decide that

this request for f_t belongs to I' and hence not to I''. In this case the online algorithm decides to either evict the file from its cache or to bypass it (i.e., with probability 1, it will not have f_t in its cache). Otherwise (if $x(f_t, r(f_t, t)) < \frac{1}{2}$), we set $x(f_t, r(f_t, t)) = 0$ and in this case this request will belong to I''. In addition to the partition, a subset of I'', called A is maintained. Consider a request for a file f which is bypassed. If the previous request for f in I is not bypassed, then after this request, the cache cannot contain f anymore (and the previous request for f which belongs to I'' is added to A). Therefore, the algorithm should be forced to remove f from the cache, and the values of the corresponding variables are set accordingly.

We note that all the primal variables increase as we consider the new request. The unique step of the algorithm in which a primal variable is decreased, is when we reset the value of $x(f_t, r(f_t, t))$ to 0. However, this variable correspond to the current request, and hence it was set to 0 when the request is revealed. Therefore, when we consider the change to the primal variables between consecutive requests, this variable does not decrease. Clearly, the dual variables are only increased throughout the iterations of the algorithm. We define Algorithm Fractional Caching with Bypassing (fcb).

Lemma 1. *The solution X is a feasible solution to (LP-caching) whose cost is at most twice the cost of x (for (LP-bypassing)).*

Lemma 2. *The dual solution y violates the constraints of (LP-dual) by a factor of at most $1 + \ln k$. That is, the vector $\frac{y}{1+\ln k}$ is a feasible solution to (LP-dual).*

Lemma 3. *The cost of x is at most four times the objective function value of y.*

We summarize the above three lemmas by concluding that X is a feasible solution to (LP-caching) whose cost is at most $8(1 + \ln k)$ times the cost of a feasible solution to (LP-dual), and by weak-duality of linear programming this is at most $8(1 + \ln k) \cdot \text{OPT}_b(I)$. The cost of bypassing the requests which do not belong to I'' is at most the cost of x, and hence this cost is at most $4(1 + \ln k) \cdot \text{OPT}_b(I)$. We change the distribution of the states of the cache only in iterations which belong to I'', and this incurs a cost which is either a constant times the cost of X (in the bit model, fault model, and the cost model) or a factor of $O(\log k)$ times the cost of X in the general case. Thus, we established the following.

Theorem 5. *There is a randomized $O(\log k)$-competitive algorithm for caching with bypassing in the cost model, in the bit model and in the fault model, and a randomized $O(\log^2 k)$-competitive algorithm for the weighted caching with bypassing. Moreover, there is a randomized $O(\log k)$-competitive algorithm for caching with rejection in the cost model, in the bit model and in the fault model, and a randomized $O(\log^2 k)$-competitive algorithm for the weighted caching with rejection.*

Algorithm. Fractional Caching with Bypassing (fcb)

Maintain a set of requests $A \subseteq I'' \subseteq I$. Initially $I'' = A = \emptyset$.

The solution X to (LP-caching) is always defined to be $X(f, r_{I''}(f_t, t)) = 1$ if $(f, r_{I''}(f_t, t)) \in A$, and otherwise $X(f, r_{I''}(f_t, t)) = 2x(f, r(f_t, t))$.

Given a request for a file f_t at time t do the following:

1. Set $x(f_t, r(f_t, t)) = 0$. Implicitly set $y(t, S) = 0$ for all $S \in \mathcal{S}(t)$.
2. Until all primal constraints of (LP-bypassing) corresponding to time t are satisfied do the following.
   (a) Pick a small set S whose primal constraint (for time t) is not satisfied by the current solution x.
   (b) Increase the dual variable $y(t, S)$ continuously and for each primal variable $x(f, j)$ such that $f \in S$ do:
   (c) If $x(f, j) = 1$ then remove f from S and continue (letting $S = S \setminus \{f\}$).
   (d) If $x(f, j) = 0$ and $\sum_{t=t(f,j)}^{t(f,j+1)-1} \sum_{S \in \mathcal{S}(t): f \in S} \min\{size(S) - k, size(f)\} \cdot y(t, S) = cost(f)$ then increase $x(f, j)$ to be $\frac{1}{k}$.
   (e) Otherwise (that is if $\frac{1}{k} \leq x(f, j) < 1$) increase $x(f, j)$ to be equal to

$$\frac{1}{k} \cdot e^{\frac{1}{cost(f)} \cdot \left(\left[\sum_{t=t(f,j)}^{t(f,j+1)-1} \sum_{S \in \mathcal{S}(t): f \in S} \min\{size(S)-k, size(f)\} \cdot y(t,S) \right] - cost(f) \right)}.$$

3. If $x(f_t, r(f_t, t)) < \frac{1}{2}$, then set $x(f_t, r(f_t, t)) = 0$ and add $(f_t, r(f_t, t))$ to I''.
4. Otherwise, set $x(f_t, r(f_t, t)) = 1$, and add $(f_t, r_{I''}(f_t, t))$ to A.

End of Algorithm. fcb

References

1. Achlioptas, D., Chrobak, M., Noga, J.: Competitive analysis of randomized paging algorithms. Theoretical Computer Science 234(1-2), 203–218 (2000)
2. Albers, S., Arora, S., Khanna, S.: Page replacement for general caching problems. In: Proc. of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 1999), pp. 31–40 (1999)
3. Bansal, N., Buchbinder, N., Naor, J.: Towards the randomized k-server conjecture: A primal-dual approach. In: Proc. of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2010), pp. 40–55 (2010)
4. Bansal, N., Buchbinder, N., Naor, S.: A primal-dual randomized algorithm for weighted paging. In: Proc. of the 48th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2007), pp. 507–517 (2007)
5. Bansal, N., Buchbinder, N., Naor, S.: Randomized competitive algorithms for generalized caching. In: Proc. of the 40th Annual ACM Symposium on Theory of Computing, STOC 2008 (2008)
6. Bar-Noy, A., Bar-Yehuda, R., Freund, A., Naor, J., Schieber, B.: A unified approach to approximating resource allocation and scheduling. Journal of the ACM 48(5), 1069–1090 (2001)

7. Bartal, Y., Leonardi, S., Marchetti-Spaccamela, A., Sgall, J., Stougie, L.: Multiprocessor scheduling with rejection. SIAM Journal on Discrete Mathematics 13(1), 64–78 (2000)
8. Belady, L.A.: A study of replacement algorithms for a virtual-storage computer. IBM Systems Journal 5(2), 78–101 (1966)
9. Blum, A., Burch, C., Kalai, A.: Finely-competitive paging. In: Proc. of the 40th Annual IEEE Symposium on Foundations of Computer Science (FOCS 1999), pp. 450–458 (1999)
10. Cao, P., Irani, S.: Cost-aware www proxy caching algorithms. In: Proc. of the USENIX Symposium on Internet Technologies and Systems, pp. 193–206 (1997)
11. Chrobak, M., Karloff, H.J., Payne, T.H., Vishwanathan, S.: New results on server problems. SIAM Journal on Discrete Mathematics 4(2), 172–181 (1991)
12. Chrobak, M., Woeginger, G.J., Makino, K., Xu, H.: Caching is hard – even in the fault model. In: de Berg, M., Meyer, U. (eds.) ESA 2010. LNCS, vol. 6346, pp. 195–206. Springer, Heidelberg (2010)
13. Cohen, E., Kaplan, H.: Caching documents with variable sizes and fetching costs: An LP-based spproach. Algorithmica 32(3), 459–466 (2002)
14. Dósa, G., He, Y.: Bin packing problems with rejection penalties and their dual problems. Information and Computation 204(5), 795–815 (2006)
15. Epstein, L.: Bin packing with rejection revisited. Algorithmica 56(4), 505–528 (2010)
16. Epstein, L., Levin, A., Woeginger, G.J.: Graph coloring with rejection. Journal of Computer and System Sciences 77(2), 439–447 (2011)
17. Epstein, L., Noga, J., Woeginger, G.J.: On-line scheduling of unit time jobs with rejection: minimizing the total completion time. Operetions Research Letters 30(6), 415–420 (2002)
18. Fiat, A., Karp, R.M., Luby, M., McGeoch, L.A., Sleator, D.D., Young, N.: Competitive paging algorithms. Journal of Algorithms 12, 685–699 (1991)
19. Goemans, M.X., Williamson, D.P.: A general approximation technique for constrained forest problems. SIAM Journal on Computing 24(2), 296–317 (1995)
20. Irani, S.: Page replacement with multi-size pages and applications to web caching. Algorithmica 33(3), 384–409 (2002)
21. Karlin, A., Manasse, M., Rudolph, L., Sleator, D.D.: Competitive snoopy caching. Algorithmica 3, 79–119 (1988)
22. Manasse, M.S., McGeoch, L.A., Sleator, D.D.: Competitive algorithms for server problems. Journal of Algorithms 11(2), 208–230 (1990)
23. McGeoch, L.A., Sleator, D.D.: A strongly competitive randomized paging algorithm. Algorithmica 6(6), 816–825 (1991)
24. Nagy-György, J., Imreh, C.: Online scheduling with machine cost and rejection. Discrete Applied Mathematics 155(18), 2546–2554 (2007)
25. Seiden, S.S.: Preemptive multiprocessor scheduling with rejection. Theoretical Computer Science 262(1), 437–458 (2001)
26. Sleator, D.D., Tarjan, R.E.: Amoritzed efficiency of list update and paging rules. Communications of the ACM 28, 202–208 (1985)
27. Young, N.E.: The k-server dual and loose competitiveness for paging. Algorithmica 11(6), 525–541 (1994)
28. Young, N.E.: On-line file caching. Algorithmica 33(3), 371–383 (2002)

On the Advice Complexity of the k-Server Problem*

Hans-Joachim Böckenhauer[1], Dennis Komm[1], Rastislav Královič[2], and Richard Královič[1]

[1] Department of Computer Science, ETH Zurich, Universitätstrasse 6, 8092 Zurich, Switzerland
{hjb,dennis.komm,richard.kralovic}@inf.ethz.ch
[2] Department of Computer Science, Comenius University, Mlynská dolina, 84248 Bratislava, Slovakia
kralovic@dcs.fmph.uniba.sk

Abstract. Competitive analysis is the established tool for measuring the output quality of algorithms that work in an online environment. Recently, the model of *advice complexity* has been introduced as an alternative measurement which allows for a more fine-grained analysis of the hardness of online problems. In this model, one tries to measure the amount of information an online algorithm is lacking about the future parts of the input. This concept was investigated for a number of well-known online problems including the k-server problem.

In this paper, we first extend the analysis of the k-server problem by giving both a lower bound on the advice needed to obtain an optimal solution, and upper bounds on algorithms for the general k-server problem on metric graphs and the special case of dealing with the Euclidean plane. In the general case, we improve the previously known results by an exponential factor, in the Euclidean case we design an algorithm which achieves a constant competitive ratio for a very small (i. e., constant) number of advice bits per request.

Furthermore, we investigate the relation between advice complexity and randomized online computations by showing how lower bounds on the advice complexity can be used for proving lower bounds for the competitive ratio of randomized online algorithms.

1 Introduction

Although the concept of information processing lies at the heart of computer science, we lack a satisfactory precise definition of the term "information". Both the concepts of entropy due to Shannon [11] and of Kolmogorov complexity [4,9], while being extremely valuable research instruments, cannot be used to estimate the information content of particular objects.

Computing a solution for an instance of a computing problem can be seen as extracting some desired information about this instance that is somehow hidden

* This work was partially supported by ETH grant TH 18 07-3, SNF grant 200020-120073, and VEGA grant 1/0671/11.

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 207–218, 2011.

in its description. In other words, information processing aims for extracting some part of information that is contained in the given input. In this way, computing means to transform one representation of information into another. This suggests that understanding what information really is could be crucial for a better understanding of the nature of computing. One possible way to capture the meaning of the notion information is the concept of the information content of computing problems as proposed by Hromkovič et al. [8].

Here, we apply this the concept of information content to online problems. In an online setting, we deal with an environment where algorithms are needed which have to permanently produce chunks of output depending on an input they read continuously. These so-called *online algorithms* aim at having a high output quality without knowing the whole input at specific time steps, i. e., they may merely base their computations on the input they have read so far.

At first, we formally define the terms of an online algorithm and its competitive ratio which is usually used to measure the algorithm's output quality on some input.

Definition 1. *Consider an input sequence* $I = (x_1, \ldots, x_n)$ *for some minimization problem* U. *An* online algorithm $\mathtt{A}$ *computes the output sequence* $\mathtt{A}(I) = (y_1, \ldots, y_n)$, *where* $y_i = f_i(x_1, \ldots, x_i)$ *for some sequence of functions* $f_1, \ldots, f_n$. *The* cost *of the solution is given by* $\mathrm{cost}(\mathtt{A}(I))$. *By* $\mathrm{Sol}_{\mathtt{A}} = \mathtt{A}(I)$ *we denote a solution computed by* $\mathtt{A}$ *on* I. *An algorithm* $\mathtt{A}$ *is* c-competitive, *for some* $c \geq 1$, *if there exists a constant* α *such that, for every input sequence* I, $\mathrm{cost}(\mathtt{A}(I)) \leq c \cdot \mathrm{cost}(\mathtt{Opt}(I)) + \alpha$, *where* $\mathtt{Opt}$ *is an optimal offline algorithm for the problem. If* $\alpha = 0$, *then* $\mathtt{A}$ *is called* strictly c-competitive. *Finally,* $\mathtt{A}$ *is* optimal *if it is strictly* 1*-competitive.*

Note that Definition 1 can be easily adapted to include the case of maximization problems. The concept of the *competitive analysis* was introduced in [12] by Sleator and Tarjan. For a more detailed introduction to it and online algorithms in general, we refer to the standard literature, e. g., [3,7].

However, comparing online algorithms to optimal solutions, which may only be constructed if the whole input is known in advance, does not seem very realistic because, by the nature of many real-world online scenarios, optimal results can never be reached. We want to get a deeper understanding of what online algorithms really lack. We do this by analyzing the information content of the given online problem [8], i. e., we investigate what additional information we need to supply these algorithms with to increase their performance. The idea of *online algorithms with advice* was proposed in [5]. However, the original model used advice over a two letter alphabet $\{0, 1\}$ and an additional delimiter \$. In [2], a new model was introduced to prevent the hidden encoding of information by implicitly using the delimiter symbol. In this new model, such online algorithms are considered that are allowed to access an *advice tape* ϕ, which has an infinite number of bits written on it. These *advice bits* are calculated by an oracle which has access to the whole input before the online algorithm gets the first part of it. At any time step, an online algorithm $\mathtt{A}$ may then read, together with the

chunk of input it gets, as many advice bits as needed. We use the same model in this paper.

Definition 2. *An* online algorithm $\mathtt{A}$ with advice *computes the output sequence* $\mathrm{Sol}_{\mathtt{A}}^{\phi} = \mathtt{A}^{\phi}(I) = (y_1, \ldots, y_n)$ *such that* y_i *is computed from* $\phi, x_1, \ldots, x_i$*, where* ϕ *is the content of the advice tape, i. e., an infinite binary sequence.* $\mathtt{A}$ *is* c-competitive with advice complexity $s(n)$ *if there exists a constant* α *such that, for every* n *and for each input sequence* I *of length at most* n*, there exists some* ϕ *such that* $\mathrm{cost}(\mathtt{A}^{\phi}(I)) \leq c \cdot \mathrm{cost}(\mathtt{Opt}(I)) + \alpha$ *and at most the first* $s(n)$ *bits of* ϕ *have been accessed during the computation of* $\mathtt{A}^{\phi}(I)$*.*

Emek et al. [6] proposed a different way to fix the model from [5] by restricting the online algorithm to read a fixed number of advice bits in every time step. With such an approach, it is not possible to analyze sublinear advice complexity, what is a serious issue for many online problems [2]. It is easy to simulate the model from [6] with our model, which is more general in this sense, and all lower bounds in our model directly carry over to the model from [6]. On the other hand, the upper bounds carry over from the model from [6] to our model. Furthermore, the model presented in [6] captures the idea that the advice is not available from the start, but comes gradually as more and more of the input is revealed. Although it seems that for concrete problems some bootstrapping techniques may be used to transform upper bounds from our model to the model from [6], the two models seem to be incomparable in general. Our interest focuses on the number of advice bits $s(n)$ that are necessary [sufficient] to achieve optimality or a specific competitive ratio.

In this paper, we consider the problem k-SERVER in which k agents (so-called *servers*) move in a metric space satisfying requests which appear in an online fashion. More formally, we look at the following problem.

Definition 3. *Let* $G = (V, E, d)$ *be a complete undirected weighted graph, where* V *is a (not necessarily finite) set of vertices,* $E = \{\{v, w\} \mid v, w \in V, v \neq w\}$ *is the set of edges, and* $d : E \to \mathbb{R}$ *is a* metric *cost function, that is,* d *satisfies the triangle inequality* $d(\{v, u\}) \leq d(\{v, w\}) + d(\{w, u\})$ *for every* $u, v, w \in V$*. Furthermore, we are given a set of* k *servers, residing in some vertices of the graph. Let* $C_i \subseteq V$ *be the multiset of vertices occupied by servers at time step* i*: A vertex occupied by* j *servers occurs* j*-times in* C_i*. We also call* C_i *the* configuration *at time step* i*. Then, a vertex* v_i *is requested and some servers may be moved yielding a new configuration* C_{i+1}*. The* request v_i *is* satisfied *if, after this movement of servers, some server resides in* v_i*, i. e., if* $v_i \in C_{i+1}$*. The* distance *between two configurations* C_1 *and* C_2 *is given by the unique cost of a minimum-weight matching between* C_1 *and* C_2*.*

The k-SERVER *problem is the problem to satisfy all requests while minimizing the sum of the distances of all consecutive configurations.*

Although Definition 3 allows to place several servers to the same point, it is easy to see that this is not necessary; it is easy to modify any algorithm for k-SERVER such that it never places more than one server to one vertex and such that this modification do not increase costs of any solution produced by the algorithm.

The solution of k-SERVER is a sequence of configurations, and between two configurations, arbitrary number of servers can be moved. However, sometimes it is convenient to restrict ourselves to so-called *lazy algorithms*, which move exactly one server in response to each yet uncovered request. This can be done without loss of generality, as any algorithm for k-SERVER can be transformed to a lazy one without any increase in the costs of produced solutions [3]. It is easy to see that, for the case of lazy algorithms, the produced solutions can be uniquely described as a sequence of servers used to satisfy individual requests.

This paper is organized as follows: In Section 2, we give a lower bound on the number of advice bits needed to achieve optimality. Section 3 is devoted to designing an algorithm for the special case where the servers move on the Euclidean plane. The main result is presented in Section 4 where we give an algorithm for the general case which performs very well using only a constant number of bits per request. This algorithm is also valid in the restricted model from [6] and exponentially improves over the k-SERVER algorithm presented there. In Section 5, we relate the advice complexity to the model of randomized online algorithms and show how the advice complexity can be used to prove lower bounds on the competitive ratio achievable by randomized online algorithms. Some proofs are omitted due to space restrictions and can be found in the technical report [1].

For the ease of presentation, throughout this paper, we denote $\log_2$ by log.

2 A Lower Bound on the Optimality

In this section, we focus on the number of bits needed to obtain an optimal solution. Hence, we show that, if an online algorithm with advice A is optimal, there exist instances in which A needs to read large advice together with every request. First, we give a bound for inputs of fixed length k which we extend afterwards to instances of arbitrary length. Note that any graph with a cost function d that maps edges from $V \times V$ to values of 1 and 2 only, trivially respects the triangle inequality and is therefore a metric.

Lemma 1. *For any $k \in \mathbb{N}$, there exists an instance of the k-SERVER problem with k requests in total, for which any online algorithm A with advice needs at least $k(\log k - c)$ bits of advice to be optimal for some constant $c < 1.443$.*

Proof. Let $k \in \mathbb{N}$ and let $G = (U \cup W, E)$ be a complete bipartite graph with a metric cost function $d : E \to \{1, 2\}$, where $U = \{u_1, u_2, \ldots, u_k\}$ and $W = \{w_1, w_2, \ldots, w_{2^k}\}$. Since $|W| = 2^k$, we can define a bijective mapping $\mathrm{Set}\colon W \to 2^U$ which maps every vertex from W to a unique subset of vertices in U. We define the edge costs in G as follows: for $v \in U$ and $w \in W$, let

$$d(\{v, w\}) = \begin{cases} 2, & \text{if } v \in \mathrm{Set}(w) \\ 1, & \text{otherwise.} \end{cases}$$

Additionally, since formally the instance for the k-SERVER problem has to contain a complete weighted graph, we define the cost for all edges from $(U \times U) \cup (W \times W)$

to be 2. A schematic view of the constructed graph for $k = 4$ is shown in Figure 1. Let $G_i \subseteq W$ denote the vertices from W corresponding to subsets of U with exactly i elements, i. e., $G_i = \{w \in W \mid |\mathrm{Set}(w)| = i\}$. Clearly,

$$|G_i| = \binom{k}{i}.$$

We construct a class of instances $\mathcal{I}'$ in the following way. An instance $I \in \mathcal{I}'$ consists of a graph G as above, where each vertex of U is covered by a single server, and a sequence $(x_0, x_1, x_2, \ldots, x_{k-1})$ of requests such that

1. $x_j \in G_j$ and
2. $\mathrm{Set}(x_j) \subseteq \mathrm{Set}(x_{j+1})$.

Intuitively, the first requested vertex is the unique vertex from W with only cheap edges to U. Every following request has exactly one more expensive edge, chosen in such a way that the set of expensively connected vertices from U is extended by one vertex in each time step.

Clearly, we may represent I as a permutation π_I of $\{1, 2, \ldots, k\}$ in the following way:

$$\pi_I(j) = \mathrm{Set}(x_j) \setminus \mathrm{Set}(x_{j-1}) \text{ for } j = 1, 2, \ldots, k-1, \text{ i. e., } \pi_I(j) = y_j \in U,$$
$$\pi_I(k) = U \setminus \{\pi_I(j) \mid j = 1, 2, \ldots, k-1\}.$$

In other words, $\pi_I(j)$ denotes the index of that vertex from U which is connected to the requested vertex via an expensive edge from request x_j on. The unique optimal solution Sol_I for I with cost of exactly k can also be described by π_I in the following way. For every j, Sol_I satisfies the j-th request x_{j-1} by moving one server from some vertex from U, in particular the one in vertex $\pi_I(j)$, to the requested vertex from W, over a cheap edge. It is easy to see that there is no solution with cost of less than k, since all the servers start in U, all requests are unique vertices from W, and every edge has a cost of at least 1. To see why Sol_I is indeed the unique optimal solution, consider an offline environment where an optimal offline algorithm `Opt` receives the whole input at once and may satisfy the requests in an arbitrary order. It does so in the opposite order the requests are made: the last vertex requested is from the group G_{k-1} and there is one unique vertex $v_{\pi_I(k)}$ connected to it with a cheap edge. The vertex that was requested before is from G_{k-2}. Due to the construction, it also has a cheap edge to $v_{\pi_I(k)}$ and to a second vertex $v_{\pi_I(k-1)}$, so that `Opt` now uses this second edge. Following this strategy, it is immediately clear that `Opt` uses exactly k edges of cost 1 and that its strategy is the only one being no more expensive than k.

Since we may represent every instance from $\mathcal{I}'$ by a unique permutation of $\{1, 2, \ldots, k\}$, we need to distinguish $k!$ different cases. It remains to show that we also need a unique advice string for every input to be solved optimally by any online algorithm `A` with advice. Towards contradiction, let I_1 and I_2 be two different inputs from $\mathcal{I}'$. Suppose that `A` is optimal for both of these inputs. However, for the same advice string ϕ, the algorithm `A` behaves deterministically.

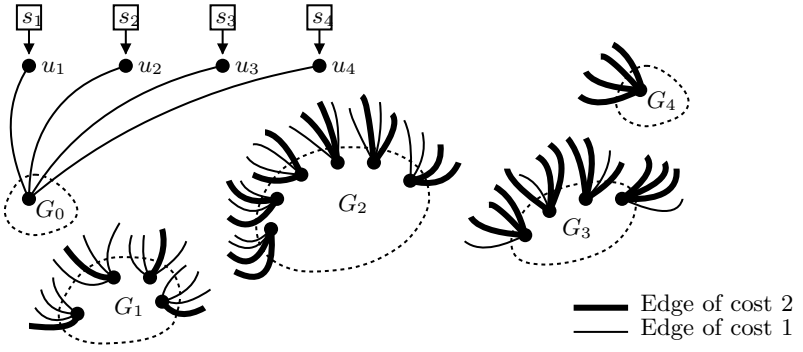


Fig. 1. A sample instance for 4-server as used in the proof of Lemma 1. Note that, for the ease of presentation, not all edges are shown completely.

Let us take the algorithm's point of view: In time step 1, the only vertex from G_0 is requested and A uses some server to satisfy this request. Then, in time step 2 it is revealed whether this was a good choice, i. e., if the server at $\pi_I(1)$ was used to serve the first request. After that, the algorithm chooses a second server to move and again, in time step 3, it is revealed whether this was a good choice, and so on. Suppose that the corresponding permutations of I_1 and I_2 differ at position j for the first time. This means that, in time step $j-1$, the algorithm has to make two different choices for the different inputs. But since it reads the same prefix of the input up to this point and furthermore uses the same advice string, it has to act the same way. But this directly implies that A cannot be optimal for both I_1 and I_2. We conclude that we need a different advice string for every instance and therefore $\log(k!)$ advice bits. Using Stirling's Formula, we get

$$\log(k!) \geq \log\left(\sqrt{2\pi k}\left(\frac{k}{\mathrm{e}}\right)^k\right) \geq k(\log k - c),$$

where $c = \log \mathrm{e} < 1.443$, which concludes our proof. □

We generalize this statement in the following theorem to an arbitrary number of requests. The idea is to use two instances as in the proof of Lemma 1, which are connected together. The optimal solution is forced to perform optimally within these two instances in an alternating way and any wrong step within any instance cannot be compensated later. Due to space restrictions the proof is omitted.

Theorem 1. *The number of bits necessary to enable* A *to be optimal can be bounded from beneath by* $n\left(\frac{1}{2}\log k - \frac{1}{2}c\right)$ *for some* $c < 1.443$*, where* n *is the number of requests.*

Note that, if we allow the graph to have unbounded size, it is easy to construct a lower bound of $n(\log k - \log \mathrm{e})$ by branching the graph infinitely often.

Note that all lower bounds presented in this chapter directly carry over to the model of [6]. Furthermore, [6] presents a very simple upper bound of $\log k$ bits per request. Hence, the lower bound in Theorem 1 is tight up to a factor of 2.

3 An Upper Bound for the Euclidean Case

In this section, we restrict the k-SERVER problem to the case where the underlying metric space is the two-dimensional Euclidean plane. For this case, we propose a simple algorithm that achieves a constant competitive ratio with an advice of linear size (in particular, the algorithm reads a constant number of bits with every request).

Let us fix a parameter b such that the algorithm uses b bits of advice per request. The algorithm A works as follows: if the requested point is $r = (r_x, r_y)$, it divides the plane into 2^b disjoint segments $S_1, \ldots, S_{2^b}$ with their origin in r, and angle $\frac{2\pi}{2^b}$ each. Then it reads b bits of advice that identify a segment S_i, and serves the request greedily with the closest server from S_i. We show that A has a constant competitive ratio with linear advice.

Theorem 2. *For a fixed $b \geq 3$, A uses b bits of advice per request, and achieves a competitive ratio of at most $1/\left(1 - 2\sin\left(\frac{\pi}{2^b}\right)\right)$.*

Before proving Theorem 2, let us first show the following technical lemma which we need in the analysis.

Lemma 2. *For $0 < \alpha \leq \frac{\pi}{4}$, let $\mathcal{C}(\alpha, R)$ be the region of the Euclidean plane $\mathcal{C}(\alpha, R) = \{[r\cos\varphi, r\sin\varphi] \mid 0 < \varphi \leq \alpha, 0 < r \leq R\}$. For any point $p \in \mathcal{C}(\alpha, R)$ let*

$$f(p) = \frac{\|p\|}{R - \|[R, 0] - p\|},$$

where $\|v\|$ is the Euclidean length of v. Then $f(p)$ is maximized over region $\mathcal{C}(\alpha, R)$ for $p_{max} = R \cdot [\cos\alpha, \sin\alpha]$ and $f(p_{max}) = \frac{1}{1-2\sin\left(\frac{\alpha}{2}\right)}$.

The situation described by the Lemma is illustrated in Figure 2(a). The proof of the Lemma is straightforward and omitted due to space restrictions.

Now we are ready to analyze the competitive ratio of A. Let us adopt the following notation: a configuration C is a multiset of k points that are occupied by the servers. A configuration $C_{p_1 \mapsto p_2}$ is obtained from C by moving a server from $p_1 \in C$ to p_2. An instance of the problem is a configuration, and a sequence of requests (points); the length of the instance is the number of requests. We restrict ourselves to lazy algorithms, hence, as already noted, a solution of an instance is a sequence of servers. To describe a server used to satisfy certain request, it is sufficient to specify the point occupied by this server, hence the solution can be described by a sequence of points as well. We prove the following theorem:

Theorem 3. *For any instance $I = (C, r_1, r_2, \ldots, r_n)$, and a solution Sol_I, the cost of A on I is at most $q \cdot \mathrm{cost}(\mathrm{Sol}_I)$ where $q = \frac{1}{1-2\sin\left(\frac{\pi}{2^b}\right)}$.*

Proof. Let Sol_I serve the i-th request r_i by a server located at s_i, with a cost of $d_i = \|s_i - r_i\|$. In the first request, A uses b bits to specify the segment of angle $\alpha = \frac{2\pi}{2^b}$ around r_1 in which s_1 is located, and moves the closest server in

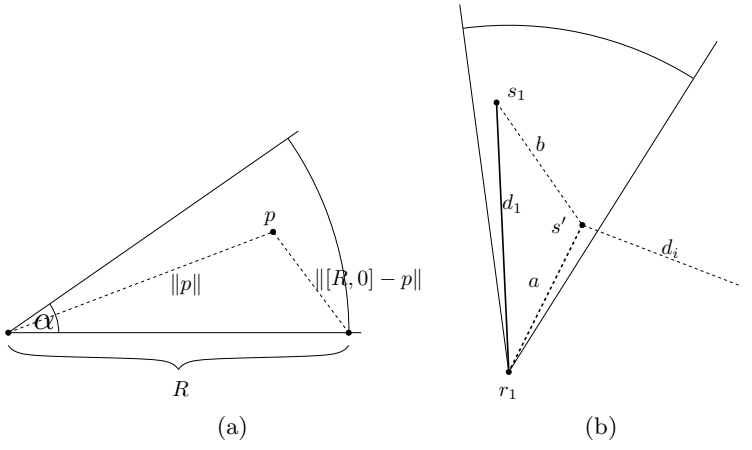


Fig. 2. (a) Illustration of Lemma 2. (b) Illustration of the algorithm.

this segment, s' to r_1 incurring distance $a \leq d_1$. Hence, after the first request, Sol_I lead to configuration $C_{s_1 \mapsto r_1}$, whereas A is in configuration $C_{s' \mapsto r_1}$. This situation is illustrated in Figure 2(b).

The proof is done by induction on n. If $n = 1$, the cost of A is $a \leq d_1 = \mathrm{cost}(\mathrm{Sol}_I)$. Let $n > 1$, and let r_i be the first request that is served by s' in Sol_I. Consider the instance $I' = (C_{s' \mapsto r_1}, r_2, \ldots, r_n)$; the sequence $(s_2, \ldots, s_{i-1}, s_1, s_{i+1}, \ldots, s_n)$ is a solution of I' with cost at most $b + \sum_{i=2}^{n} d_i$. By induction, the cost of A on I' is at most $q \cdot (b + \sum_{i=2}^{n} d_i)$, hence the cost of A on I is at most

$$a + q \cdot \left(b + \sum_{i=2}^{n} d_i \right).$$

Due to Lemma 2, $a \leq q(d_1 - b)$ and the result follows. □

Theorem 3 immediately implies that A reaches competitive ratio q using b advice bits per request. For 3, 4, and 5 bits per request, the corresponding ratios are 4.261972632, 1.639829878, and 1.243834129. With $b \mapsto \infty$, the competitive ratio converges to 1.

4 An Upper Bound for the General Case

We now focus on the trade-off between the advice size and the competitive ratio achievable in general metric spaces.

Theorem 4. *For every $b \geq 2$, there is an online algorithm solving the k-*SERVER *problem that uses $b \cdot n$ advice bits for inputs with n requests and achieves competitive ratio $2 \left\lceil \frac{\lceil \log k \rceil}{b-1} \right\rceil \leq 4 + \frac{2}{b-1} \log k$.*

Proof (idea). At first, we describe the algorithm A. With each request, A reads one bit of advice called a *control bit*. If this bit is zero, A satisfies the request greedily with the nearest server. Afterwards, the used server is returned to its original position before the next request. If the control bit is one, A reads the next $\log k$ bits. These bits specify which server should be used to satisfy the request. After the request is satisfied, the server is left at its new position.

It is possible to prove that, for every input instance, there exists an advice such that A has a competitive ratio as claimed and such that A uses at most bi bits of advice while processing the first i requests.

The intuition behind our algorithm is as follows: In the first step, a greedy move (obviously) has cost smaller than or equal to the move of the optimal solution. Hence, doing the greedy move and returning the server back incurs at most twice the cost of the first step of the optimal solution. On the other hand, the configuration of the server stays the same. Thus, we have not disrupted the configuration too much, i. e., if we follow the optimal solution in the future, the price for doing the first step in a (non-optimal) greedy way is not too high. Indeed, assume that the greedy solution satisfies the first request r_1 by using a server located at r_0, incurring cost $2d(\{r_0, r_1\})$ and that the optimal algorithm should continue from this configuration. If there is a request satisfied by a server not located at r_1, there is no change in comparison to the original optimal solution. If a server from r_1 is used in the original optimal solution, the optimal algorithm could use the server located at r_0 instead, incurring an extra cost of $d(\{r_0, r_1\})$. Thus, the greedy move of the first step caused an extra cost $2d(\{r_0, r_1\})$ immediately and an extra cost $d(\{r_0, r_1\})$ that propagates further.

These arguments can be generalized for a greedy step in any configuration: The greedy step incurs some extra cost immediately and some extra cost that is propagated. The algorithm A can, however, receive a full specification of the optimal move after some cost has been propagated $q := \left\lceil \frac{\lceil \log k \rceil}{b-1} \right\rceil$ times. To do that, it is sufficient to set the control bit to 0 for q times, and then to 1, thus stopping the error propagation. Doing so, we amortize the $\lceil \log k \rceil$ non-control bits of advice over q steps, giving at most b amortized bits of advice per request. Because every cost contributing to the original optimal solution is propagated at most q times and, for every propagation, contributes to the constructed solution at most twice, the competitive ratio of A is at most $2q$.

The full proof can be found in the technical report. □

We have proven Theorem 4 in the model of an advice tape. Nevertheless, the result of this theorem can be easily adapted to the model with fixed number of advice bits received with each request, as used in [6]. In this case, consider the advice tape created in the proof of Theorem 4. Separate this tape into two bit sequences, one containing only control bits, the other containing non-control bits only. Afterwards, interleave these bits, always taking $b - 1$ non-control bits followed by a single control bit to create a single b-bit advice message. Algorithm A then reads $b - 1$ non-control bits with every message and stores them into a FIFO data structure. Afterwards, it reads the control bit; if this bit is 1, $\lceil \log k \rceil$

bits are extracted from the FIFO. The proof of Theorem 4 ensures that there are always enough bits stored in the FIFO when a control bit 1 arrives.

Note that Theorem 4 improves the result of [6] exponentially: With b bits per request, a feasibility of competitive ratio of $k^{\Theta(\frac{1}{b})}$ was proven in [6], while Theorem 4 proves competitive ratio of $\log\left(k^{\Theta(\frac{1}{b})}\right)$.

Furthermore, note that, while we did not impose any restriction on the way the advice string is generated in our definition, the advice strings used for our upper bounds are always efficiently computable from an optimal offline solution.

5 Relation between Randomization and Advice

One intriguing aspect of the advice-based analysis of online problems is the relationship of advice and randomization. Specifically, one can ask if there is some connection between the existence of an efficient randomized algorithm and the existence of an efficient algorithm with small advice. One can view the randomized algorithm as randomly selecting one witness object from a universe of possible witnesses. To achieve a good performance, the universe must be constructed in such a way that, for any input, there are many reasonably good witnesses. On the other hand, the advice can be interpreted as selecting a particular witness from a given universe. Hence, for algorithms with advice it is necessary to have a small universe such that for any input at least one good witness exists.

From Yao's minimax principle [13] it follows that, if there is a randomized algorithm with a given expected cost, then there is a small universe of witnesses such that for each input there is a reasonably good witness, as stated in the following theorem.

Theorem 5. *Consider a minimization online problem P such that there are $|\mathcal{I}(n)| = I(n)$ possible inputs of length n, where $\mathcal{I}(n)$ is the set of all possible inputs of length n. Furthermore, let us suppose that there is a randomized algorithm* R *with worst-case expected competitive ratio at most $E(n)$. Then, for any fixed $\varepsilon > 0$, it is possible to construct a deterministic algorithm* A *with advice $\log n + 2\log\log n + \log\left(\frac{\log I(n)}{\log(1+\varepsilon)}\right)$ with competitive ratio $E(n)(1+\varepsilon)$.*

Proof. Let us suppose that R uses $r(n)$ random bits, so there are $R(n) = 2^{r(n)}$ possibilities for a random string, which is the only source of randomness used by R. Let us construct a matrix A with $I(n)$ rows and $R(n)$ columns such that $A_{x,q}$ is the competitive ratio of R on input x with random string q. We show how to construct a set of strings $\mathcal{W}$ containing $\frac{\log I(n)}{\log(1+\varepsilon)}$ elements such that, for each input x, there is a string $w \in \mathcal{W}$ such that the competitive ratio of R on input x with random string w is at most $(1+\varepsilon)E(n)$. The first part of the advice used by A is the number of requests n encoded in a self-delimited form using $\log n + 2\log\log n$ bits. Such an encoding can be achieved by writing the value of $\log\log n$ in unary base, extending it by the value of $\log n$ in binary base, which uses $\log\log n$ digits, and finally extending it by the value of n in binary base,

which uses $\log n$ digits. Algorithm A knows a-priori the set $\mathcal{W}$, since it depends on n only. The second part of the advice for a given input is the index of the particular string $w \in \mathcal{W}$.

Since the expected competitive ratio of R for any input is at most $E(n)$, we have $\forall x \in \mathcal{I}(n)\colon \frac{1}{R(n)} \sum_q A_{x,q} \leq E(n)$. Hence, the overall sum in the matrix is at most $R(n) \cdot I(n) \cdot E(n)$, and

$$\exists w\colon \sum_{x \in \mathcal{I}} A_{x,w} \leq I(n) \cdot E(n). \tag{1}$$

This w will be included in $\mathcal{W}$, and will be used as advice for any input x where $A_{x,w} \leq (1+\varepsilon)E(n)$; let us call such x a *hit* for w. We argue that w has many hits: for a given number s of hits we have

$$\sum_{x \in \mathcal{I}} A_{x,w} > (I(n) - s)\,(1+\varepsilon)E(n). \tag{2}$$

From (1) and (2) we get $s > I(n)\frac{\varepsilon}{1+\varepsilon}$. Now we have covered $I(n)\frac{\varepsilon}{1+\varepsilon}$ inputs by one string w. Consider the sub-matrix of A obtained by removing the column w and the rows corresponding to the already covered inputs. Since, for all remaining $I(n)\frac{1}{1+\varepsilon}$ inputs, the string w contributed more than $E(n)$ to the row-sum, the average of any row in the sub-matrix is still at most $E(n)$.

We can repeat the above argument, every time covering a fraction of $\frac{\varepsilon}{\varepsilon+1}$ of the inputs. Hence, after $\log I(n)/\log(1+\varepsilon)$ steps, all possible inputs are covered by some $w \in \mathcal{W}$. □

Please note that the algorithm constructed in the previous example is not polynomial: in order to decode the advice, the algorithm has to construct the dictionary $\mathcal{W}$ by simulating the randomized algorithm on all inputs of length n.

In general, this bound cannot be extended to the case $\varepsilon = 0$, as can be seen by the following (rather artificial) "winner-takes-all" problem:

Definition 4 (Winner-takes-all problem). *The input consists of a sequence of requests $x_1 = 0, x_2, \ldots, x_{n+1}$, where $x_i \in \{0,1\}$. The solution is a sequence $y_i \in \{0,1\}$. The cost of a given solution is* 1*, if $y_i = x_{i+1}$ for all $i \in \{1,\ldots,n\}$, and* 2 *otherwise. Hence, the optimal solution has cost* 1 *and all other solutions pay an extra penalty of* 1*.*

It is easy to show that the best possible randomized algorithm is to guess each bit with equal probability, so the expected cost is $2 - \frac{1}{2^n}$. Obviously, any algorithm with advice less than n bits has worst-case performance 2, so n bits are needed to be on par with randomization.

Theorem 5 relates the advice and randomization in such a way that lower bounds on advice complexity can be used to deliver lower bounds on randomization. Specifically, for the k-server problem, the famous "randomized k-server" conjecture (RKSC) (for a survey see, e. g., [10]) states that, for any metrical space, there is a randomized online algorithm with competitive ratio $\mathcal{O}(\log k)$. This would imply that for a (discrete) metric space with M elements, there is an

algorithm with advice $\mathcal{O}(\log n + \log\log\log M)$ that is $\mathcal{O}(\log k)$-competitive. Our results, however, grant the $\mathcal{O}(\log k)$ competitive ratio only by using $\mathcal{O}(n)$ advice bits. Hence, if one believes the RKSC, the upper bound is still exponentially far from the optimum. On the other hand, any lower bound of the form $\omega(\log n)$ would disprove the RKSC. Compare this to the specific case of k-server [2] in a space where all distances are 1 (i. e., the paging problem) where it is known that linear advice is needed for optimality, and constant (i. e., independent of n) advice is sufficient for being $\mathcal{O}(\log k)$-competitive.

References

1. Böckenhauer, H.-J., Komm, D., Královič, R., Královič, R.: On the advice complexity of the k-server problem. Technical Report 703, ETH Zürich (2010)
2. Böckenhauer, H.-J., Komm, D., Královič, R., Královič, R., Mömke, T.: On the advice complexity of online problems. In: Dong, Y., Du, D.-Z., Ibarra, O. (eds.) ISAAC 2009. LNCS, vol. 5878, pp. 331–340. Springer, Heidelberg (2009)
3. Borodin, A., El-Yaniv, R.: Online Computation and Competitive Analysis. Cambridge University Press, Cambridge (1998)
4. Chaitin, G.J.: On the length of programs for computing finite binary sequences. Journal of the ACM 13(4), 547–569 (1966)
5. Dobrev, S., Královič, R., Pardubská, D.: How much information about the future is needed? In: Geffert, V., Karhumäki, J., Bertoni, A., Preneel, B., Návrat, P., Bieliková, M. (eds.) SOFSEM 2008. LNCS, vol. 4910, pp. 247–258. Springer, Heidelberg (2008)
6. Emek, Y., Fraigniaud, P., Korman, A., Rosén, A.: Online computation with advice. In: Albers, S., Marchetti-Spaccamela, A., Matias, Y., Nikoletseas, S., Thomas, W. (eds.) ICALP 2009. LNCS, vol. 5555, pp. 427–438. Springer, Heidelberg (2009)
7. Hromkovič, J.: Design and Analysis of Randomized Algorithms: Introduction to Design Paradigms. Texts in Theoretical Computer Science. An EATCS Series. Springer, New York (2005)
8. Hromkovič, J., Královič, R., Královič, R.: Information complexity of online problems. In: Hliněný, P., Kučera, A. (eds.) MFCS 2010. LNCS, vol. 6281, pp. 24–36. Springer, Heidelberg (2010)
9. Kolmogorov, A.N.: Three approaches to the definition of the concept "quantity of information". Problemy Peredachi Informatsii 1, 3–11 (1965)
10. Koutsoupias, E.: The k-server problem. Computer Science Review 3(2), 105–118 (2009)
11. Shannon, C.E.: A mathematical theory of communication. Mobile Computing and Communications Review 5(1), 3–55 (2001)
12. Sleator, D.D., Tarjan, R.E.: Amortized efficiency of list update and paging rules. Communications of the ACM 28(2), 202–208 (1985)
13. Yao, A.C.-C.: Probabilistic computations: Toward a unified measure of complexity (extended abstract). In: FOCS, pp. 222–227. IEEE, Los Alamitos (1977)

Sleep Management on Multiple Machines for Energy and Flow Time

Sze-Hang Chan[1], Tak-Wah Lam[1], Lap-Kei Lee[2], Chi-Man Liu[1], and Hing-Fung Ting[1]

[1] Department of Computer Science, University of Hong Kong, Hong Kong
[2] MADALGO Center for Massive Data Algorithmics, Aarhus University, Denmark

Abstract. In large data centers, determining the right number of operating machines is often non-trivial, especially when the workload is unpredictable. Using too many machines would waste energy, while using too few would affect the performance. This paper extends the traditional study of online flow-time scheduling on multiple machines to take sleep management and energy into consideration. Specifically, we study online algorithms that can determine dynamically when and which subset of machines should wake up (or sleep), and how jobs are dispatched and scheduled. We consider schedules whose objective is to minimize the sum of flow time and energy, and obtain $O(1)$-competitive algorithms for two settings: one assumes machines running at a fixed speed, and the other allows dynamic speed scaling to further optimize energy usage.

Like the previous work on the tradeoff between flow time and energy, the analysis of our algorithms is based on potential functions. What is new here is that the online and offline algorithms would use different subsets of machines at different times, and we need a more general potential analysis that can consider different match-up of machines.

1 Introduction

Energy consumption is a major concern for large-scale data centers. It has been reported that the energy consumption of the data centers in US costs more than \$4.5 billion a year and accounts for more than 1.5% of the total electricity usage in US [15]. When a machine (or server) is on, the power consumption is divided into *dynamic power* and *static power*; the former is consumed only when the machine is processing a job, while the latter is consumed constantly (due to leakage current) even when the machine is idle (e.g., an Intel Xeon E5320 server requires 150W when idling and 240W when working [7]). The static power consumption is cut off only when a machine is sleeping. From the energy viewpoint, a data center should let machines sleep whenever they are idle, yet waking up a machine later requires extra energy. It is energy inefficient to frequently switch a machine on and off. It is challenging to determine dynamically the appropriate number of working machines so as to strike a balance between energy usage and quality of service (QoS), especially when the workload is unpredictable. This paper initiates a theoretical study of online job scheduling on a pool of identical

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 219–231, 2011.

machines that takes sleep management, energy and QoS into consideration. We consider both models where machines are running at a fixed speed and machines can each scale their speed to control their power, respectively.

Flow time scheduling. A well-studied QoS measurement for job scheduling is the total flow time. The flow time (or simply the flow) of a job is the time elapsed since the job is released until it is completed. Without energy concern, there is already considerable amount of research on minimizing total flow time alone (see the survey [14]). It is well-known that the online algorithm SRPT (shortest remaining processing time) minimizes total flow time on a single machine. For scheduling on $m > 1$ identical machines, no online algorithm can be $O(1)$-competitive even if job migration is allowed; nevertheless, Chekuri et al. [6] showed that the non-migratory algorithm IMD [4] is $O(1 + \frac{1}{\epsilon})$-competitive for any $\epsilon > 0$ when using $(1 + \epsilon)$ times faster machines.

Sleep management and energy. The above results assume that all machines are always on. This paper extends the study of multi-machine scheduling to consider sleep management and the tradeoff between flow time and energy. When a job arrives, we need to determine whether to wake up a sleeping machine to process it or assign it to an awake machine. We also have to decide when to put machines to sleep to save static power. Waking up machines too aggressively wastes a lot of energy, while an over-conservative wake-up policy accumulates excessive flow time. The objective is to minimize the flow plus energy. Lam et al. [11] studied this problem in the single-machine setting and gave an $O(1)$-competitive algorithm for flow time plus energy. In the multi-machine setting, sleep management gets complicated and cannot be considered separately on each machine; for instance, a scheduler may overload some machines to put others to sleep earlier. No online algorithm has been known for the multi-machine setting. A relevant work is by Khuller et al. [10] who considered an offline problem of minimizing makespan subject to a wake-up budget for jobs all released at time 0.

Speed scaling. All the results above assume that machines run at a fixed speed. Another energy saving technology is *dynamic speed scaling*, which allows a processor to scale its speed dynamically. Running a job slower can reduce energy usage, but it leads to longer flow time. There are several online results that take speed scaling into consideration and attempts to minimize the total flow plus energy (see the survey [1]). Most results are based on a model in which the processor, when running at speed $s \in [0, \infty)$, consumes dynamic power s^α, where $\alpha > 1$ and is typically 3 [16]. Here a scheduler needs a speed scaling policy to determine the speed of each processor. When there is only one processor which is assumed to be always on, the best algorithm uses the job selection algorithm SRPT and the speed scaling algorithm AJC, which sets its speed based on the number of active jobs [13]. This algorithm is 2-competitive for flow plus energy [5,3]. Lam et al. [11] have also studied single-processor scheduling that exploits both sleep management and speed scaling, and gave an $O(\frac{\alpha}{\log \alpha})$-competitive algorithm for flow plus energy.

Existing online results on speed scaling for multiple machines do not consider sleep management [12,9,8]. The best result in the non-migratory model is by Gupta et al. [8]. They assume each machine runs the best single-machine speed scaling algorithm as mentioned above, and consider a job dispatching algorithm which assigns each new job to the machine that would result in the least increase in the future flow plus energy. The algorithm is $O(\alpha)$-competitive for flow plus energy. It remains open whether there exists a competitive algorithm when machines can exploit both speed scaling and sleep management.

Note that speed scaling and sleep management might work in opposite directions; the former prefers load balancing and working slowly, while the latter could overload some machines (and make them run faster) so as to let other machines sleep. Furthermore, speed scaling can be optimized separately within individual machine, but sleep management often requires considering all machines together.

Our contribution. This paper gives the first sleep management algorithm on multiple machines for minimizing total flow plus energy. Our results cover both the fixed speed and speed scaling models. We only consider non-migratory schedules since migration requires overheads in practice. In the fixed speed model, it is easy to show that any online algorithm is $\Omega(m)$-competitive even if all jobs have unit size. In view of the lower bound, we give the online algorithm $(1+\epsilon)$-speedup processors, which run $(1+\epsilon)$ times faster but does not consume more power. We show an online algorithm POOL that is $O(1+\frac{1}{\epsilon})$-competitive for total flow plus energy when using $(1+\epsilon)$-speedup processors. (Our result remains valid even if a machine when running at speed $(1+\epsilon)$ is charged for more power, say, $(1+\lambda)$ times of the original power. The competitive ratio would increase by a factor of $(1+\lambda)$.) We can adapt POOL to the speed scaling model using AJC; it is $O(\alpha)$-competitive for flow plus energy. Our new algorithm has comparable performance with the best speed scaling results on multiple machines that assume all machines are always on [12,9,8].

The core idea of our algorithm is to keep a pool of "dispatchable" machines, which are either all asleep or all awake. A new job is dispatched only to a machine currently in the pool. We separate the management of the pool from the job dispatching policy; the former depends on the history of the workload while the latter depends on the size of the current job. We exploit three simple ideas to manage the pool: (1) uses the total flow to trigger all machines in the pool to wake up; (2) uses the total working flow plus energy to decide when to include more machines; and (3) uses the total idling energy to decide when to remove a machine from the pool. Like many online algorithms, POOL is conservative in committing resources (i.e., waking up machines). Yet POOL can keep its flow under control even if there are many occasions when it uses too few machines.

We also face new challenge when analyzing the competitiveness. As in previous work, we need to discover the "right" potential function to account for the difference in the progress of the online algorithm and the optimal offline algoithm OPT. With different sleep management, POOL and OPT may indeed operate with a different subset of machines, and it is possible that POOL makes machine i heavily loaded while OPT puts machine i to sleep. It makes no sense

to compare their progress of machine i in POOL and OPT. We should match the machines of POOL and OPT with the same state and measure the progress of each pair. This paper gives a new potential analysis that allows us to dynamically change the matching of the machines so as to minimize the number of state mismatch. One might think that changing the matching might require us to "restart" the potential analysis. Yet we observe that such change cannot increase the potential. Another important observation is that we can restrict our attention to those offline algorithms that at any time have at most one machine that is sleeping and has unfinished jobs.

Definitions and notations. The input is a sequence of jobs arriving online, to be scheduled on $m \geq 1$ machines. The job's size is arbitrary and only known at release time. Jobs are sequential in nature (i.e., each job can be processed by one machine at a time). We consider only non-migratory schedules, in which each job is dispatched to one machine and is run entirely on that machine. Preemption is allowed, and a preempted job can resume at the point of preemption.

Sleep, awake, and static power. At any time, a machine is in either the *awake* state or the *sleep* state. It can process a job only when it is awake, and the energy is consumed at the rate $\mu > 0$, which includes both static and dynamic power. An awake machine can be idle (i.e., speed $= 0$) and only requires the static power σ ($0 < \sigma < \mu$). It can enter the sleep state to further reduce the power to zero. Initially, all machines are in the sleep state. Following the literature, we assume that state transition is immediate but requires energy. A *wake-up* from the sleep state to the awake state requires an amount ω of energy, and the reverse takes zero energy. It is useful to differentiate two types of awake state, namely, with zero speed and with positive speed, which are referred to respectively as the *awake-idle* and *awake-working* state, or simply the *idle* and *working* state. Furthermore, we call a sleeping machine in the *procrastinating* state if the machine has jobs not yet finished.

Fixed-speed and speed scaling models. In the fixed-speed model, a working machine runs at speed 1 (i.e., processing x units of work in x units of time) and the power required is μ. In the speed scaling model, a working machine can run at any speed $s > 0$ at any time but at different power. We assume that the *power function* or the rate of energy usage is $P(s) = s^\alpha + \sigma$, where $\alpha > 1$ is a constant.

Flow and energy. Consider a schedule of jobs. At any time t, a job j is said to be *active* if j has been released but not yet completed. Its *flow time* or simply *flow* $F(j)$ is the time elapsed since it arrives and until it is completed. The total flow is $F = \sum_j F(j)$. Let n_t be the number of active jobs at time t. We can also view the total flow as a quantity incurring at a rate equal to the number of active jobs, i.e., $F = \int_0^\infty n_t \, \mathrm{d}t$. Energy is consumed by awake machines continuously over time and by discrete wake-up transitions. Denote the total energy as E. The cost G of a schedule is defined to be $F + E$. And the objective is to find a schedule that minimizes G. Furthermore, suppose that each job is dispatched to some machine immediately at release time. Then we can consider the flow and energy machine by machine. For each machine, we are particularly interested in the flow and energy incurred over the time when it is in the working state.

We define the *working cost* G_{W} of a schedule to be the sum over all machines of the flow and energy incurred when the machine is in the working state. Obviously, $G_{\mathrm{W}} < G$ as the latter includes the flow incurred when a machine is sleeping, the energy when idle, and the wake-up energy.

2 Sleep Management for Fixed-Speed Machines

This section focuses on fixed-speed machines and gives our algorithm POOL that handles sleep management, job dispatching and scheduling in an integrated way. POOL is using $m > 1$ machines that are $(1+\epsilon)$ times faster than the offline optimal algorithm OPT, while using the same power. (Recall that without extra speed, any online algorithm is $\Omega(m)$-competitive.) Following is our main result.

Theorem 1. *For any $\epsilon > 0$,* POOL *is $O(1+\frac{1}{\epsilon})$-competitive for flow plus energy when using $(1+\epsilon)$-speedup machines.*

Remaining working cost (rwc). As to be shown, POOL schedules jobs using SRPT. It is useful to compute the remaining working cost (rwc) required to serve the remaining jobs on any machine i that runs SRPT at a speed $s \geq 1$. For any time t and any $q \geq 0$, let $n_{i,t}(q)$ be the number of active jobs in machine i with remaining work at least q. Then at any time t, the *rwc* of machine i equals $\int_{q=0}^{\infty} \sum_{k=1}^{n_{i,t}(q)} (\frac{k+\mu}{s})\,\mathrm{d}q$. Moreover, if a job j of size $p(j)$ arrives at time t and is dispatched to machine i immediately, the increase in *rwc* due to j equals $\int_{q=0}^{p(j)} (\frac{n_{i,t}(q)+1+\mu}{s})\,\mathrm{d}q$ (note that $n_{i,t}(q)$ refers to the number before j arrives). Obviously, for a sequence of jobs J, the sum of the increases in *rwc* at their release times equals the total working cost of serving J.

2.1 Algorithm POOL

The core idea is to maintain a small pool P of *dispatchable machines*. P contains one sleeping machine initially and is always non-empty. At any time, machines in P are either all asleep or all awake, and P is said to be asleep and awake, respectively. Machines not in P are all asleep and do not have active jobs. POOL would gradually include more machines into P as jobs arrive, and they are put into the same state as P. POOL exploits three simple concepts to manage P: (1) total flow for triggering all machines in P to wake up; (2) total working cost for determining when to include more machines into P; and (3) total idling energy for determining when to put an idle machine to sleep and remove it from P.

To this end, POOL maintains three (real-value) counters B, C and D to keep track of the accumulated flow (when P is asleep), increase in working cost and idling energy, respectively. Initially, all counters equal 0. C only increases when a job arrives. When P is asleep, B increases (continuously) at rate of the number of active jobs. When P is awake, D increases at rate of σ times the number of idle machine, but once D reaches ω, its value is capped there. Intuitively, when D reaches ω, we could remove one idle machine from P. But this turns out to

be too aggressive. Let P_{idle} be the set of idle machines in P. Indeed POOL never sleeps an idle machine if it is the only idle machine in P (i.e., $|P_{\text{idle}}| = 1$) but $|P| \geq 2$; it does so only if there are two or more idle machines or $|P| = 1$.

When a job j arrives, POOL first assumes that j is dispatched to a machine with no active jobs and calculates the increase in *rwc*. Denote this amount of increase as $null\_Inc\_rwc(j)$. If $null\_Inc\_rwc(j)$ can compensate the wake-up energy, POOL will include one machine into P and dispatch j to this machine (even if P already has an idle machine). Otherwise, POOL dispatches j to a machine i in P that minimizes the increase in *rwc*; below we denote machine i as $\ell(j, P)$ and the amount of increase in *rwc* as $min\_Inc\_rwc(j, P)$. Note that $min\_Inc\_rwc(j, P) \geq null\_Inc\_rwc(j)$, and the total *rwc* also increases by the same amount in both cases. C keeps track of the increase in *rwc*; whenever it reaches a multiple of ω, we include one more machine into P.

Job dispatching (& expand P): When a job j arrives,
If $((|P| < m)$ & $(C + null\_Inc\_rwc(j) \geq \omega))$,
add a machine to P; dispatch j to this machine; $C = C + null\_Inc\_rwc(j) - \omega$;
else dispatch j to $\ell(j, P)$; $C = C + min\_Inc\_rwc(j, P)$;
While $((C \geq \omega)$ & $(|P| < m))$ **do** { add a machine to P; $C = C - \omega$. }
If $C > \omega$ **then** $C = \omega$.
Wake up P: If (P is asleep) & $(B = \omega)$, wake up all P's machines; reset $B = 0$.
Sleep a machine (& shrink P): When $D = \omega$,
- if $|P_{\text{idle}}| \geq 2$, remove one idle machine from P and put it to sleep; reset $D = 0$;
- if $|P_{\text{idle}}| = |P| = 1$, put P to sleep; reset $C = D = 0$.

Job scheduling in each machine of P: When awake, use SRPT policy.

Intuitively, scheduling for a small job (say, $null\_Inc\_rwc(j) < \omega$) is not obvious. It is too small to justify waking up a machine, yet dispatching it to an awake machine may preempt other jobs there (due to SRPT policy) and suddenly cause a huge increase of *rwc*. Interestingly, POOL can maintain a useful property that it always has at least one awake machine with a small workload and dispatching j to it cannot increase the *rwc* too much (say, $\leq 2\omega$).

Property 1. Every time after POOL executes the job dispatching procedure, it maintains the invariant that if $|P| < m$, there exists a machine with $rwc < \omega$.

POOL never lets an awake machine idle if that machine has active jobs. Thus, a machine can accumulate flow only when it is working or sleeping. Consider a schedule of POOL. We divide POOL's total flow F into two parts: the *working* flow F_{W} and the *sleeping* flow F_{S}, which refer to the total flow incurred by the machines when they are working and sleeping, respectively. We also divide POOL's energy usage E into three parts: E_{I} is the idling energy (static energy usage during the idle state), E_{W} the working energy, and U the wake-up energy. Note that POOL's working cost $G_{\text{W}} = F_{\text{W}} + E_{\text{W}}$, and its total cost $G = F_{\text{W}} + F_{\text{S}} + E_{\text{W}} + E_{\text{I}} + U$.

The analysis of POOL centers on a rather complicated potential analysis of F_{W}, which gives Lemma 1(i) below. Note that we will bound F_{W} by OPT's total cost G^* together with the sleeping flow F_{S}, because the sleep management

of POOL sometimes delays jobs and increases their flow. Such excess in flow is related to F_S but not OPT. To upper bound the other components of G, we show Lemma 1(ii). Lemmas 1(i) and (ii) would imply that POOL is $O(1)$-competitive.

Lemma 1. (i) $F_W \leq (9 + \frac{10}{\epsilon})G^* + \frac{1}{\epsilon}F_S$; (ii) $G \leq 4F_W + 7G^*$.

Proof (Sketch of Lemma 1(ii)). We derive three properties of POOL: (a) $F_S \leq G^*$; (b) $U \leq G_W + G^*$; (c) $E_I \leq U + E_w$. As POOL uses $(1+\epsilon)$-speedup machines, E_W is less than OPT's working energy and thus G^*. Thus, Lemma 1(ii) follows.

It is useful to partition the timeline into intervals called P-intervals, each of which consists of a maximal asleep period of P, followed by a maximal awake period of P. For a P-interval I, we can show that the total cost incurred by OPT within I (denoted by $G^*(I)$) is at least ω. For (a), the sleeping flow accumulated by POOL in the asleep period of I is $\omega \leq G^*(I)$. Summing over all I's gives $F_S \leq G^*$. For (b), within I, except for the first machine added to P, a machine is added to P when the accumulated increase of *rwc* (i.e., counter C) reaches a multiple of ω. When I ends, this accumulated increase in *rwc* would be fully reflected in the working cost G^* incurred within I. Thus, within I, the total wake-up energy (including that for the first added machine) is at most the working cost plus ω (at most $G^*(I)$). Summing over all I's gives $U \leq G_W + G^*$. For (c), E_I can be incurred when $D < \omega$ and when $D = \omega$. The first type increases at the same rate as D and is at most U. The second type is incurred when $|P_{\text{idle}}| = 1$ but $|P| \geq 2$, i.e., a working machine exists in P, which is thus at most E_W. □

2.2 Potential Analysis of F_W

One might think that POOL is rather conservative in waking up machines and might sacrifice flow for energy. Indeed POOL can always catch up in time its number of machines and keep its flow under control. In particular, we can upper bound POOL's increase of flow even when it is using fewer machines than OPT. The analysis is complicated because POOL and OPT may use different subsets of (awake) machines. The rest of this section shows Lemma 1(i) using a potential function that allows different match-up between machines of POOL and OPT.

Restricting OPT. In analyzing F_W, we restrict OPT to be the optimal offline algorithm that always uses *SRPT* for job selection and has at most one procrastinating machine at any time. In Section 4, we show that such OPT incurs at most three times the total cost of an unrestricted one. Henceforth, we focus on the restricted OPT and show that POOL is $O(1)$-competitive against it.

Let $F_W(t)$ denote the working flow F_W incurred up to time t by POOL. Similarly, define $G^*(t)$ for OPT's total cost G^*. Assume that machines are labeled with integers from 1 to n. At any time, we match each machine in POOL with a certain machine in OPT. Below we denote $x(i)$ as the machine in OPT currently matched with machine i in POOL. This matching is only for the purpose of analysis and not known to the algorithms. Initially $x(i) = i$ for all i. To show Lemma 1(i), we define a potential function $\Phi(t)$ that reflects POOL's remaining

working cost discounted in view of OPT's workload in the corresponding machines. Technically, we want $\Phi(t)$ to satisfy the following conditions: **(i)** *Boundary condition:* $\Phi = 0$ before any job is released and after all jobs are completed. **(ii)** *Job completion and state transition condition:* Φ does not increase when a job is completed or a machine changes its state in POOL or OPT. **(iii)** *Job arrival condition:* After a job arrives and gets dispatched, we re-match the machines. Φ may increase, yet the total increase due to all job arrivals is upped bounded by $O(G^*)$ (precisely, $(8+\frac{9}{\epsilon})\cdot G^*$). **(iv)** *Running condition:* At any other time t, $\frac{\mathrm{d}F_{\mathrm{w}}(t)}{\mathrm{d}t} + \frac{\mathrm{d}\Phi(t)}{\mathrm{d}t} \le (1+\frac{1}{\epsilon})\cdot\frac{\mathrm{d}G^*(t)}{\mathrm{d}t} + \frac{1}{\epsilon}\cdot\frac{\mathrm{d}F_{\mathrm{s}}}{\mathrm{d}t}$. By integrating the above conditions over time, we have $F_{\mathrm{w}} \le (8+\frac{9}{\epsilon}+1+\frac{1}{\epsilon})G^* + \frac{1}{\epsilon}F_{\mathrm{s}}$. Then Lemma 1(i) follows.

Potential function Φ. For any machine i of POOL, for any $q \ge 0$, recall that $n_{i,t}(q)$ denotes the number of active jobs with remaining work at least q at time t. Define $n^*_{i,t}(q)$ similarly for OPT. We will drop the parameter t when t refers clearly to the current time. Let $(\cdot)_+ = \max(\cdot, 0)$. The potential function is

$$\Phi(t) = \textstyle\sum_{i=1}^{m} \Phi_i(t) \quad \text{where} \quad \Phi_i(t) = \frac{1}{\epsilon}\int_0^\infty \sum_{k=1}^{n_i(q)} (k - n^*_{x(i)}(q) + \mu)_+ \,\mathrm{d}q \ .$$

Machine re-matching. $x(1), \ldots, x(m)$ form a permutation of $1, 2, \cdots, m$. At any time once a new job has been dispatched to a machine by *ALG* and OPT, we keep swapping $x(i)$ and $x(j)$ as long as we find machines i and j satisfying:

- POOL has i not in P and OPT has $x(i)$ awake or procrastinating; and
- POOL has j in P and OPT has $x(j)$ sleeping.

Note that Φ_i, Φ_j and Φ may change after a swapping. Interestingly, we can verify that this change must be non-increasing.

Lemma 2. *After some $x(i)$ and $x(j)$ are swapped, Φ_i and Φ_j does not increase.*

We can easily show the boundary, job completion and state transition conditions. The running condition depends solely on the job scheduling policy (SRPT) and can be analyzed independently for each matched pair of machines using techniques for the single-machine analysis [5,11]; details will be given in full paper.

The core of the potential analysis is the arrival condition, which depends on both sleep management and job dispatching policies. Below we show that when a job arrives, after a special re-matching of machines, the increase of flow or technically Φ can be bounded in terms of some non-overlapping cost of OPT.

Lemma 3. *The sum over all jobs of the increase in Φ due to a job arrival (and machine re-matching) is at most $(8+\frac{9}{\epsilon})\cdot G^*$.*

At the point after a job j arrives and gets dispatched by POOL and OPT, we re-match their machines (i.e., compute a new matching function $x(i)$) before we re-calculate Φ. This re-matching process is for analysis sake and can make reference to any information in the POOL and OPT's schedule in the past or future. To formally define the inputs to the re-matching process, we need to first construct a list of "interesting" events ordered by time. There are 6 types of events: *JobArrive*(j)—a new job arrives and is dispatched by

POOL and OPT; POOL_In(j)—POOL adds a machine into the pool P due to job j; POOL_Out—removes a machine from P; OPT_Wake—OPT wakes up a machine; OPT_Sleep—OPT sleeps a machine; and *Rematch*—execute the re-matching procedure based on POOL and OPT's status as defined up to the previous event. When there are multiple events at the same time, they are arranged in the following order: POOL_Out, all OPT_Wake, *JobArrive*(j), *Rematch*, all POOL_In(j), *JobArrive*(j'), *Rematch*, all POOL_In(j'), $\cdots$, all OPT_Sleep. Let $A = (e_1, e_2, \ldots e_c)$ be the list of events. For each event e, we define h_e to be the number of machines in P immediately after e, and h^*_e the number of awake machines in OPT. We omit e when it is clear that we refer to the current event. Notice that before the first event of A, $h = 1$ and $h^* = 0$.

Lazy intervals. A *lazy interval* is a maximal sequence of events in A containing at least one *JobArrive*, in which all events e have $h_e \leq h^*_e$. By definition, a lazy interval must start with a POOL_Out or OPT_Wake event and end before a POOL_In or OPT_Sleep event. For any lazy interval ℓ, excluding the first event, let I_ℓ, O_ℓ, W^*_ℓ and S^*_ℓ be respectively the number of POOL_In, POOL_Out, OPT_Wake and OPT_Sleep events in ℓ. W.r.t. the first and the last event of a lazy interval, $h = h^*$. This implies the following useful property of a lazy interval.

Property 2. For a lazy interval ℓ, $I_\ell + S^*_\ell = W^*_\ell + O_\ell$.

Type-0, Type-1 and Type-2 jobs. Define Type-0 jobs to be jobs which POOL dispatches to a zero-*rwc* machine (i.e., no active jobs). For any other job, it is Type-1 if its *JobArrive* event e is in a lazy interval and $h_e < m$; otherwise, it is Type-2. Type-0 jobs are easy to analyze. Roughly speaking, Type-2 jobs arrive when POOL is using more machines than the OPT (or is using all m machines); after re-matching the machines, it is relatively easy to show that Φ has limited increase (see Lemma 4; proof will be given in the full paper). For Type-1 jobs, POOL might be using very few machines and POOL's increase in *rwc* can be way larger than OPT's. We analyze Type-1 jobs interval by interval (instead of job by job) and show that POOL's increase in *rwc* is bounded by the static energy and wakeup energy of OPT (see Lemma 5(b)). Then Lemma 3 follows.

Lemma 4. *The total increase in Φ due to Type-2 jobs is at most $\frac{1}{\epsilon} \cdot G^*$.*

For Type-0 and Type-1 jobs, it is relatively easy to see that the total increase in Φ is at most $(1 + \frac{1}{\epsilon})$ times the total increase in *rwc* of POOL. Thus, Lemma 5 implies that the increase in Φ due to Type-0 and Type-1 jobs is most $8(1+\frac{1}{\epsilon}) \cdot G^*$.

Lemma 5. (a) POOL*'s total increase in rwc due to Type-0 jobs is at most G^*.* (b) POOL*'s total increase in rwc due to Type-1 jobs is at most $7G^*$.*

Lemma 5(a) is obvious: when a Type-0 job arrives, POOL's increase in *rwc* cannot exceed that of OPT. To show Lemma 5(b), let $\Delta G'_\ell$ and $\Delta G'$ be the increase in *rwc* to POOL due to Type-1 jobs in a lazy interval ℓ and all Type-1 jobs, respectively. Let L be the set of all lazy intervals and let $|L|$ be the size of L. Define $I_L = \sum_{\ell \in L} I_\ell$ and similarly for O_L, W^*_L and S^*_L. It is useful to define

E^*_ℓ to be the total wake-up energy used by OPT during ℓ plus the static energy used by OPT during ℓ (precisely, during the time period enclosing all events in ℓ), and E^*_L to be the sum of E^*_ℓ over all $\ell \in L$. Obviously, $E^*_L \le G^*$. We can show Lemma 6 below; details will be given in the full paper. Roughly speaking, Lemma 6(i) and (ii) show respectively that POOL can wake up more machines in react to a large increase in working cost, and POOL does not put machines to sleep too frequently. Thus, POOL is not over-conservative when having fewer awake machines than OPT. Together with Property 2, Lemma 5(b) follows.

Lemma 6. *(i)* $\Delta G' < (I_L + 2|L|)\omega$*; (ii)* $(W^*_L + O_L)\omega \le E^*_L; |L|\omega \le 3E^*_L$.

3 Sleep Management and Speed Scaling

In this section, we consider the speed scaling model, where each processor can scale its speed s in $[0, \infty)$ and consumes energy at rate $P(s) = s^\alpha + \sigma$, where $\alpha > 1$. We adapt the algorithm POOL presented in Section 2 such that each awake processor in P scales its speed by AJC (active job count) [11], as follows. For machine i, define $n_{i,t}$ and $n_{i,t}(q)$ similarly as before.

> **Speed scaling in each machine of P:** When awake, if machine i has active jobs ($n_{i,t} > 0$), set its speed to $(n_{i,t} + \sigma)^{1/\alpha}$; else its speed is 0.

We can verify that the *rwc* of a machine i becomes $2\int_{q=0}^{\infty} \sum_{k=1}^{n_{i,t}(q)} (k+\sigma)^{1-1/\alpha}\, \mathrm{d}q$.

Theorem 2. *With speed scaling,* POOL *is* $O(\alpha)$*-competitive for flow plus energy.*

To prove Theorem 2, our main idea is similar to that in Section 2, as most properties of POOL do not depend on the power function and remains valid. In particular, we compare POOL with a restricted optimal algorithm OPT that keeps at most one machine procrastinating and follows SPRT and AJC. Such restriction allows us to calculate the *rwc* of OPT. In Section 4, we show that this only increases the competitive ratio by six times. We show Lemma 7 below which is analogous to Lemma 1 in Section 2. The sleeping flow F_S, idling energy E_I and wake-up energy U can be bounded using the same techniques, which gives Lemma 7 (ii). Yet in the speed scaling model, we can no longer bound E_W by E^*_W easily. Even though we restrict OPT to use AJC as the speed scaling policy, there is no simple relation between E_W and E^*_W. Thus, we will consider E_W and F_W together and analyze G_W, the working cost. Using a modified potential function, the potential analysis framework used in the fixed speed model is adaptable to the speed scaling model, allowing us to show Lemma 7(i).

Lemma 7. *In the speed scaling model,* (i) $G_\mathrm{W} \le 11\alpha \cdot G^* + (2\alpha - 2) \cdot F_\mathrm{S}$*;* (ii) $G \le 4G_\mathrm{W} + 4G^*$.

We now define the potential function Φ for proving Lemma 7(i). As shown in Section 2, we want Φ to capture POOL's remaining working cost in discounted in view of OPT's workload in the corresponding machines. We modify Φ in view of

the new *rwc* of POOL. At any time, let $x(i)$ to be the machine in OPT currently matched with machine i in POOL. Then we define

$$\Phi(t) = \sum_{i=1}^{m} \Phi_i(t) \quad \text{where} \quad \Phi_i(t) = 2\alpha \int_0^\infty \sum_{k=1}^{n_{i,t}(q)} (k - n^*_{x(i),t}(q) + \sigma)_+^{1-1/\alpha} \, \mathrm{d}q \ .$$

We will follow a similar framework in analyzing Φ. It is easy to show the boundary, job completion and state transition conditions. The arrival and running conditions can be proven using similar ideas but requires some modification mostly due to the new definition of *rwc*. We only state the arrival and running conditions and leave the detailed proofs in the full paper.

Lemma 8. (i) *The sum over all jobs of the increase in Φ due to a job arrival (and machine re-matching) is at most $9\alpha \cdot G^*$.* (ii) *Consider any time t without job arrival, completion, machine reordering and state transition in both* POOL *and* OPT. $\frac{\mathrm{d}G_{\mathrm{w}}}{\mathrm{d}t} + \frac{\mathrm{d}\Phi}{\mathrm{d}t} \le 2\alpha \cdot \frac{\mathrm{d}G^*}{\mathrm{d}t} + (2\alpha - 2)\frac{\mathrm{d}F_{\mathrm{s}}}{\mathrm{d}t}$.

4 Transformation of Offline Schedule

This section describes the transformation for getting the suitable offline schedule.

Theorem 3. *Given any schedule S that uses speed scaling, we can transform S into another schedule S' that also uses speed scaling, and* (i) *S' has at most one procrastinating machine at any time,* (ii) *it uses SRPT for job selection and AJC for speed scaling, and* (iii) *the total cost of S' is at most 6 times that of S.*

We first transform S into a schedule S_0 with the following invariants: (1) If a job j is assigned to some machine i, then j is completed before i goes to sleep for the first time after the release of j (this invariant is mainly for simplifying analysis); and (2) it has at most one procrastinating machine at any time. The total cost of S_0 is at most 3 times that of S. Then, we show how to transform S_0 to S' that uses SRPT and AJC, and this will further blow up the cost by a factor of at most 2. Observe that we have a similar transformation for the fixed speed model, which blows up the total cost by a factor of 3 instead of 6: we transform S_0 to S' that uses SRPT, which does not further increase the cost.

We now give the essential ideas for proving Theorem 3; details will be given in the full paper. The construction of S_0 from S starts by copying the wakeup and sleeping times of each machine from S into S_0 (i.e., each machine has the same sequence of awake periods in both schedules). Then, we schedule each job into S_0 in order of release times; a job may be assigned to a different machine and different time slots than it is in S. Suppose we are considering job j, whose "execution profile" in S is $\langle i, (t_1, \ell_1, s_1), (t_2, \ell_2, s_2) \ldots, (t_m, \ell_m, s_m)\rangle$, meaning that j is assigned to i, running at time t_1 for ℓ_1 time units at speed s_1, and so on. Then, we copy this execution profile to S_0. If the resulting schedule violates an invariant, we do the following before moving on to schedule the next job.

Suppose Invariant (1) is violated, i.e., $[t_1, t_m + \ell_m]$ covers several awake periods of i. Then we "leftpack" the schedule as follows: Let u be the time machine i goes to sleep for the first time after the release of j. Then $u < t_m + \ell_m$.

We extend the awake period that ends at u as much as possible by executing j in this period. If j still cannot be completed before the next sleep time, we repeat this process. For example, suppose that j is released at time 8 and its execution profile in S is $\langle i, (12, 1, 1), (17, 6, 3), (30, 1, 1)\rangle$, and i's awake periods after 8 is $[10, 13], [17, 23], [29, 32]$. After leftpacking, the schedule for j in S_0 becomes $\langle i, (12, 1, 1), (13, 4, 3), (17, 2, 3), (23, 1, 1)\rangle$, and the awake periods of i after 8 is $[10, 24]$ and $[29, 32]$. The extra cost for executing j during $[13, 17]$ equals the cost saved by not executing j during $[19, 23]$, and during this period, i becomes idle and costs static energy; but this will at most double the original energy cost.

Now we briefly describe how to maintain Invariant (2). Before scheduling j, Invariant (2) ensures that S_0 has at most one procrastinating machine p at the release time u of job j. If $p = i$, the new schedule still satisfies Invariant (2). Suppose $p \neq i$, and after leftpacking, there is a period $[u, v]$ during which both p and i are procrastinating. If j's length (i.e. total execution time) is larger than $|[u, v]|$, we can wake up p earlier at u so that it becomes awake during $[u, v]$. Note that the extra static energy cost during this idle period is no greater than that for processing j and thus we double the energy at most. If the length of j is smaller than $|[u, v]|$, we re-assign j to p and execute it during $[v - \ell, v]$ where ℓ is the length of j. Now machine i no longer procrastinates during $[u, v]$ and hence Invariant (2) is satisfied. But there is a subtle problem here: it is possible that j completes before v in S, so in the new schedule j has an increased flow time. In such case, we employ a different strategy which involves moving multiple jobs from i to p, and the analysis becomes rather complicated (see the full paper).

Finally, we describe how to transform S_0 to S', which uses SRPT and AJC. Consider any machine i. Whenever i is awake in S_0, i is also awake in S', processing jobs using SRPT and AJC (or idles if no job remains). If i goes to sleep in S_0 at some time t, i also goes to sleep in S at time t only if it has no active jobs; otherwise it stays awake and processes the jobs using SRPT and AJC until there are no more active jobs, say at time t'. Then it copies the status of i at time t' in S_0, i.e., it goes to sleep in S' if and only if i is asleep in S_0 at time t'.

References

1. Albers, S.: Energy-efficient algorithms. CACM 53(5), 86–96 (2010)
2. Albers, S., Fujiwara, H.: Energy-efficient algorithms for flow time minimization. ACM Transactions on Algorithms 3(4), 49 (2007)
3. Andrew, L., Wierman, A., Tang, A.: Optimal speed scaling under arbitrary power functions. ACM SIGMETRICS Performance Evaluation Review 37(2), 39–41 (2009)
4. Avrahami, N., Azar, Y.: Minimizing total flow time and total completion time with immediate dispatching. In: Proc. SPAA, pp. 11–18 (2003)
5. Bansal, N., Chan, H.L., Pruhs, K.: Speed scaling with an arbitrary power function. In: Proc. SODA, pp. 693–701 (2009)
6. Chekuri, C., Goel, A., Khanna, S., Kumar, A.: Multi-processor scheduling to minimize flow time with ϵ resource augmentation. In: Proc. STOC, pp. 363–372 (2004)

7. Gandhi, A., Gupta, V., Harchol-Balter, M., Kozuch, M.: Optimality analysis of energy-performance trade-off for server farm management. Performance Evaluation 67(11), 1155–1171 (2010)
8. Gupta, A., Krishnaswamy, R., Pruhs, K.: Scalably scheduling power-heterogeneous processors. In: Abramsky, S., Gavoille, C., Kirchner, C., Meyer auf der Heide, F., Spirakis, P.G. (eds.) ICALP 2010. LNCS, vol. 6198, pp. 312–323. Springer, Heidelberg (2010)
9. Greiner, G., Nonner, T., Souza, A.: The bell is ringing in speed-scaled multiprocessor scheduling. In: Proc. SPAA, pp. 11–18 (2009)
10. Khuller, S., Li, J., Saha, B.: Energy efficient scheduling via partial shutdown. In: Proc. SODA, pp. 1360–1372 (2010)
11. Lam, T.W., Lee, L.K., Ting, H.F., To, I., Wong, P.: Sleep with guilt and work faster to minimize flow plus energy. In: Albers, S., Marchetti-Spaccamela, A., Matias, Y., Nikoletseas, S., Thomas, W. (eds.) ICALP 2009. LNCS, vol. 5555, pp. 665–676. Springer, Heidelberg (2009)
12. Lam, T.W., Lee, L.K., To, I., Wong, P.: Competitive non-migratory scheduling for flow time and energy. In: Proc. SPAA, pp. 256–264 (2008)
13. Lam, T.W., Lee, L.K., To, I., Wong, P.: Speed scaling functions for flow time scheduling based on active job count. In: Halperin, D., Mehlhorn, K. (eds.) Esa 2008. LNCS, vol. 5193, pp. 647–659. Springer, Heidelberg (2008)
14. Pruhs, K., Sgall, J., Torng, E.: Online scheduling. In: Handbook of Scheduling: Algorithms, Models and Performance Analysis, pp. 15-1–15-41. CRC Press, Boca Raton (2004)
15. U.S. Environmental Protection Agency. EPA Report on server and data center energy efficiency (2007)
16. Yao, F., Demers, A., Shenker, S.: A scheduling model for reduced CPU energy. In: Proc. FOCS, pp. 374–382 (1995)

Meeting Deadlines: How Much Speed Suffices?*

S. Anand[1], Naveen Garg[1], and Nicole Megow[2]

[1] Indian Institute of Technology Delhi, India
anand.42@gmail.com, naveen@cse.iitd.ac.in
[2] Max-Planck-Institut für Informatik, Saarbrücken, Germany
nmegow@mpi-inf.mpg.de

Abstract. We consider the online problem of scheduling real-time jobs with hard deadlines on m parallel machines. Each job has a processing time and a deadline, and the objective is to schedule jobs so that they complete before their deadline. It is known that even when the instance is feasible it may not be possible to meet all deadlines when jobs arrive online over time. We therefore consider the setting when the algorithm has available machines with speed $s > 1$.

We present a new online algorithm that finds a feasible schedule on machines of speed $e/(e-1) \approx 1.58$ for any instance that is feasible on unit speed machines. This improves on the previously best known result which requires a speed of $2 - 2/(m+1)$. Our algorithm only uses the relative order of job deadlines and is oblivious of the actual deadline values. It was shown earlier that the minimum speed required for such algorithms is $e/(e-1)$, and thus, our analysis is tight. We also show that our new algorithm outperforms two other well-known algorithms by giving the first lower bounds on their minimum speed requirement.

1 Introduction

We consider the problem of scheduling real-time jobs with hard deadlines on multiple machines. In this problem, a set of jobs $J = \{1, \ldots, n\}$ must be scheduled on m identical parallel machines, each of which can process at most one job at the time. A job $j \in J$ arrives at its release date $r_j \in \mathbb{N}$, has processing time $p_j \in \mathbb{N}$, and deadline $d_j \in \mathbb{N}$. It can be processed on any of the m machines. It can also be preempted and restarted, either on the same machine or a different machine. An instance is called *feasible*, if there exists a schedule such that no job misses its deadline. An algorithm is *optimal* if it can schedule every feasible instance so that all deadlines are met. Given a feasible instance, a feasible schedule can be computed by solving a maximum flow problem [4].

In this paper, we consider the online model, in which an algorithm learns about a job j only at its release date r_j. Several algorithms are known to be optimal on a single machine [3]. But on multiple machines, the online problem is much more difficult than its offline counterpart. In fact, for $m \geq 2$, there does not exist any optimal online algorithm [3]. To overcome this hardness, Phillips, Stein, Torng, and Wein [8] proposed the use of *resource augmentation* [5]: Given an online algorithm A we determine the speed $s \geq 1$ such that A is optimal on m speed-s processors for any instance that is feasible for m processors of unit speed. We are interested in the smallest s for which

* Research supported by the Indo-German Max Planck Center for Computer Science (IMPECS).

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 232–243, 2011.

there is an optimal online algorithm. It is known, that any optimal online algorithm needs speed at least 6/5 [8].

An algorithm is *deadline ordered* if the schedule it yields depends only on the relative ordering of the deadlines of the jobs and not on the actual deadline values. A well-known example of a deadline ordered algorithm is *Earliest Deadline First* (EDF) which at any time schedules the m jobs with the earliest deadline. EDF is optimal on a single machine [2]. On m machines, speed $s = 2 - 1/m$ is necessary and sufficient to guarantee its optimality [8]. Since its introduction more than a decade ago, this upper bound on the speed requirement for online algorithms has been improved only marginally. Lam and To [6] proposed a more complex deadline ordered algorithm with a speed requirement of $2 - 2/(m + 1)$. They also showed that any deadline ordered online algorithm for m machines needs a speed of at least

$$\alpha_m := \frac{1}{1 - \left(1 - \frac{1}{m}\right)^m}.$$

For $m = 2$ this quantity equals 4/3, matching the currently best known upper bound [6], and for arbitrary m it is at most $e/(e - 1) \approx 1.58$.

Our main result (Section 3) is a new deadline ordered online algorithm which is optimal with speed α_m. Both, the algorithm and its analysis, build on a simple and elegant online estimate of an optimal schedule (Section 2), proposed in [6]. The matching lower bound in [6] proves that α_m is the exact speed requirement for our algorithm.

We also consider two well-known non-deadline ordered algorithms and provide lower bounds on the speed necessary for them to schedule a feasible instance. Let $p_j(t)$ denote the remaining processing time of job j at time $t \geq r_j$. The laxity of j at time t is defined as $\ell_j(t) = d_j - t - p_j(t)$. The algorithm *Least Laxity First* (LLF) schedules at any point in time m jobs with minimum laxity among the available jobs. LLF is also optimal on a single machine [3], and more generally, it is optimal on m machines when running at speed $2 - 1/m$ [8]. In this paper we provide a lower bound on the speed required (Section 4) by demonstrating a feasible instance for which LLF requires a speed

$$s \geq \frac{1 + \sqrt{1 + 4x^2}}{2x} \quad \text{with } x = \frac{m}{m - 1}.$$

This quantity is $(1 + \sqrt{17})/4 \approx 1.281$, for $m = 2$, and approaches the golden ratio $(1 + \sqrt{5})/2 \approx 1.618$, when m goes to infinity. To the best of our knowledge, this is the first lower bound (beyond the general one) on the speed necessary for LLF. It also shows that our new deadline ordered algorithm outperforms LLF: Indeed, for $m \geq 7$ the lower bound for LLF exceeds the upper bound on the speed required by our new algorithm.

An algorithm that tries to combine features of EDF and LLF is *Earliest Deadline until Zero Laxity* (EDZL) introduced in [1]. At any point in time, EDZL gives highest priority to jobs which cannot be delayed further, i.e, have zero laxity, and other jobs are scheduled in EDF order. This algorithm dominates EDF in the sense that any instance that is schedulable by EDF, is also schedulable by EDZL, whereas the opposite is not true [1,7]. However, it remained open if EDZL is optimal for speed less than $2-1/m$, the speed necessary for EDF. In Section 5, we answer this question negatively by providing a feasible instance on which EDZL fails for speed less than $2 - 1/m$.

2 The Yardstick Schedule

A key challenge in designing an online algorithm for the deadline scheduling problem is to obtain an estimate of an optimal schedule. Clearly, at any time, we can compute an optimal schedule for the currently known partial instance by solving a maximum flow problem [4]. However, these optimal schedules may differ fundamentally, and it is unclear how an online algorithm can use this information as it cannot change the decisions from the past. Intuitively, we need a less powerful algorithm that computes a simpler optimal solution under some relaxed assumptions. Lam and To [6] proposed a simple and elegant schedule called *yardstick*, which can be constructed online. It has the property that all jobs meet their deadlines but it may not be feasible as it processes a job sometimes simultaneously on multiple machines. The main idea of yardstick is to allow parallelization of a job only if it is *underworked*, i.e., the total amount of processing done on it is smaller than the time period since it was released. We will use yardstick as a reference in the design and analysis of our feasible online algorithm.

The yardstick schedule is constructed as follows. Whenever a new job is released, we consider all unfinished available jobs in increasing order of deadlines and schedule their remaining processing time on the m machines of unit speed. When scheduling job j, we consider the earliest time at which some machine is available and schedule the job on all available machines till it is not underworked any more. Once the total processing done on j equals the time it has been available, we schedule the job on the lowest numbered machine (assuming an arbitrary numbering on the machines) that is available.

Since the yardstick schedule may run a job simultaneously on multiple machines, it does not give a feasible schedule. However, it has the following crucial property.

Lemma 1 ([6]). *If a scheduling instance is feasible, the yardstick schedule completes all jobs before their deadlines.*

In general, it is not necessary to specify for each job the particular machine by which it is processed, but in our case it makes the analysis simpler. In particular, yardstick distributes processing volume in a *staircase profile*: in any time slot, machine $i' > i$ is used only if machine i is also occupied, and between any two release dates the number of machines used is not increasing over time; see Figure 1.

The online algorithm that we present in the following section will at any release time t make reference to the part of the yardstick schedule after time t. This part has a non-increasing staircase profile. Given some (release) time t, let all jobs that have been released by t be indexed in increasing order of deadlines. Recall that this is the order in which yardstick considers jobs. Let Y^j denote the yardstick schedule for jobs $1, 2, \ldots, j$ starting at time t. Let $y_1^j, y_2^j, \ldots, y_i^j, \ldots$ be the time points at which there is a *step* (the total work assigned to the m machines changes) in Y^j. In particular, let $y_1^j \geq t$ be the first point in time when not all machines are used to their full capacity. Further, let x_j denote the last point in time at which job j is running on multiple machines simultaneously in the yardstick schedule, and let f_j ($f_j > x_j$) denote the time that it finishes processing (see Figure 1). For the sake of readibility we omit in our notation the parameter t since it will always be clear from the context.

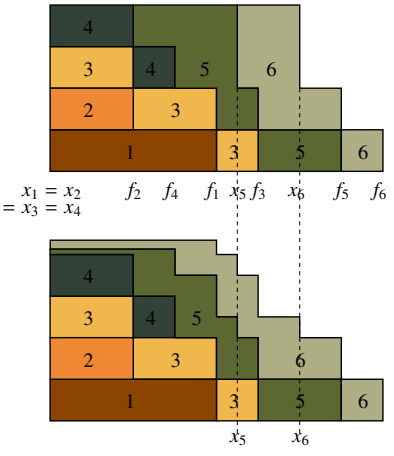


Fig. 1. The yardstick schedule (top) and our schedule (bottom)

The yardstick schedule for jobs released by time t has the following properties.

(i) If $j < k$, then $x_j < x_k$ although f_j may be greater than f_k.
(ii) Since yardstick runs a job j on multiple machines simultaneously only if it is underworked, we have $f_j \geq r_j + p_j$.
(iii) Since all machines are occupied for all times before x_j, we have $y_1^j \geq x_j$.
(iv) In going from Y^{j-1} to Y^j, all steps before x_j disappear and new steps get created at x_j and f_j (if they do not exist).

3 A Best Possible Deadline Ordered Online Algorithm

We propose a new deadline ordered online algorithm and determine the speed that is sufficient to guarantee that it is optimal.

3.1 Description of the Algorithm

Our algorithm mimics the yardstick schedule to a large extent. In particular, it will finish every job by the same time as in the yardstick schedule, which is by Lemma 1 before its deadline. Moreover, our algorithm will have processed at any time at least as much work of any job as the yardstick schedule. Our algorithm will keep up with the yardstick schedule by using faster machines instead of parallelizing jobs. The key idea is that in time periods, in which yardstick schedules a job for some positive amount, our algorithm does not process more than that. In particular when yardstick processes a job which is not underworked and hence uses only one machine, our schedule too processes only one unit of the job in one time unit even though the additional speed would allow for more.

The algorithm works as follows: At any time t when a job is released, we recompute the yardstick schedule. (This is done completely independent of our current online schedule.) Then we consider the set of available unfinished jobs, J_t, in our schedule. We consider jobs in J_t in increasing order of deadlines and assign for each job $j \in J_t$ its remaining processing time $p_j(t)$ at time t in our schedule as follows:

- We schedule one unit of work in each unit-length time slot between x_j and f_j.
- We assign the remaining processing time, $p_j(t) - (f_j - x_j)$, to the slots before x_j with α units assigned to each slot between $s_j := x_j - (p_j(t) - f_j + x_j)/\alpha$ and x_j, where $\alpha \in \mathbb{R},\ 1 \le \alpha < 2$, is a the full speed of the machines.

In this manner we determine for each time slot (after t, as we prove below) the set of jobs to be processed and the extent to which they have to be processed. Notice that we do not allocate particular machines to the jobs. However, the algorithm never assigns more work of an individual job to a time slot than can be processed sequentially. Thus, when assuming integral job parameters and allowing preemption at any time, a round robin like processing yields a schedule with each job using at most one machine at the time and no two jobs using the same machine simultaneously.

Consider the workload profile computed by our algorithm. For the analysis we desire that it has a staircase profile. However the procedure described so far may not satisfy this. In the following we show how to correct this.

At any (release) time t, let the jobs in J_t be indexed in increasing order of deadlines. Note that for both online schedules, yardstick and our schedule, we consider only the part of the schedules after time t, and that both schedules are built by considering jobs (not necessarily the same ones) in the order of increasing deadlines. Similarly as for yardstick we define A^j to be our schedule for jobs $1, 2, \ldots, j \in J_t$ after time t. Let $a_1^j, a_2^j, \ldots, a_i^j, \ldots$ be the time points at which there is a step in A^j. Let $a_1^j \ge t$ be the first point in time when not all machines are used to their full capacity after time t.

Our schedule may not have a staircase profile after the reassignment at some time t, because s_j for some job j may lie between a_i^{j-1} and a_{i+1}^{j-1}. In that case we distribute the part of j scheduled in the interval $[s_j, a_{i+1}^{j-1}]$ uniformly over the interval $[a_i^{j-1}, a_{i+1}^{j-1}]$. This however, may not suffice since the height of the profile in the interval $[a_{i+1}^{j-1}, a_{i+2}^{j-1}]$ might exceed the height of the profile in some of the preceding intervals. If this is the case, we move a suitable amount of job j from $[a_{i+1}^{j-1}, a_{i+2}^{j-1}]$ to the interval preceding it so that these two intervals have the same height in A^j. This process is repeated until we get a staircase profile for A^j. Note that as a consequence of this operation we will be scheduling at most α units of job j in each time slot preceding a_{i+2}^{j-1}.

3.2 Analysis of the Algorithm

In this section we show the following main result.

Theorem 1. *The speed $\alpha_m = (1 - (1 - 1/m)^m)^{-1}$ is necessary and sufficient for our algorithm to schedule each job feasibly before its deadline.*

We first show the correctness of our algorithm with respect to time feasibility.

Lemma 2. *If an instance is feasible, then for any job j our algorithm assigns p_j units to feasible time slots between r_j and d_j. In particular, at any time $t \geq r_j$ it (re-)assigns the remaining processing requirement $p_j(t)$ to time slots not earlier than t.*

Proof. First, consider our algorithm without the correction step achieving a staircase profile. At time r_j our algorithm assigns job j to time slots between s_j and f_j. By Lemma 1, $f_j \leq d_j$. The starting time $s_j = x_j - (p_j - f_j + x_j)/\alpha$ is at least $f_j - p_j \geq r_j$ for $\alpha \geq 1$.

Consider some release time $t > r_j$ and the yardstick schedule with the last moments of parallelization x_j (resp. x'_j) and the finishing times f_j (resp. f'_j) of j before (resp. after) rescheduling. Observe that $x'_j \geq x_j$ and $f'_j \geq f_j$. The reason is that yardstick has to schedule more jobs than before and maintains the same scheduling order depending only on deadlines. Our algorithm reassigns the remaining work $p_j(t)$ of j to $[s'_j, f'_j]$. Since this amount has been scheduled in $[t, f_j]$ before, with at most α units per time slot in $[t, f_j]$ and one unit per time slot in $[x_j, f_j]$, our algorithm reassigns it now to later time slots if $f'_j > f_j$ and to a larger amount per time slot if $x'_j > x_j$. Thus, the start time does not decrease, i.e., $s'_j \geq t$. Finally by Lemma 1, we have again $f'_j \leq d_j$ since the instance is feasible.

We have shown that $s_j \geq t$ before the correction step. Now suppose that we must start a job earlier than s_j to obtain a staircase profile. Since $a_i^{j-1} \geq t$, the proof still holds. □

To prove that our algorithm never assigns more work to a time slot than it can process on m fast machines, we proceed as follows. First, we show that at any time the remaining processing time of any job in our schedule is not more than this quantity in the yardstick schedule. Then, we show that at any point in time, our algorithm can distribute the remaining processing time that any job has *in the yardstick schedule* without exceeding the processing capacity under speed α_m.

Lemma 3. *A^j is identical to Y^j for all $t' \geq x_j$.*

Proof. We prove this by induction on j. Suppose the statement is true for all $j \leq k$. Then Y^k and A^k are identical for all t' after x_k and, since $x_{k+1} \geq x_k$, for all $t' \geq x_{k+1}$. The job $k+1$ is scheduled for one unit in each time slot between x_{k+1} and f_{k+1} in both A^{k+1} and Y^{k+1}. This implies both schedules are identical after x_{k+1}. □

Lemma 4. *For any job j and time $t' \leq f_j$, the amount of processing of j remaining at time t' in schedule A^j is less than that remaining at time t' in schedule Y^j.*

Proof. Since schedules A^j and Y^j are identical after x_j (Lemma 3), the remaining processing time of j at any time $t' \geq x_j$ is the same in both schedules.

Since in Y^j, job j is scheduled only after time x_{j-1}, for all times $t' \leq x_{j-1}$ the remaining processing time of j in Y^j is $p_j(t)$ and this is, trivially, at least as large as the remaining processing time of j at any time t' in A^j. Hence we only need to prove the statement for t' in the interval (x_{j-1}, x_j).

In Y^j one or more units of j are scheduled in each time slot between x_{j-1} and x_j. We now consider two cases.

1. At least 2 units of j are scheduled in each slot between x_{j-1} and x_j in Y^j. However, in A^j at most $\alpha < 2$ units of j are scheduled in each slot between x_{j-1} and x_j. Hence for any time t' in the interval (x_{j-1}, x_j), the remaining processing time of j in Y^j exceeds that of j in A^j.
2. Only one unit of j is scheduled in Y^j for some slots in (x_{j-1}, x_j). Since Y^{j-1} has a staircase profile, for all slots between $x_{j-1} = y_1^{j-1}$ and y_2^{j-1}, j must be scheduled for only one unit, while for slots between y_2^{j-1} and x_j, j must be scheduled for at least 2 units.

   By the argument in previous case, it follows that for all time $t' \geq y_2^{j-1}$, the remaining processing time of j in Y^j exceeds that of j in A^j. This implies that before time y_2^{j-1}, job j is processed to a larger extent in A^j than in Y^j. From our procedure for scheduling job j it follows that we schedule j for β units in each time slot in the interval (y_1^{j-1}, y_2^{j-1}), where $1 \leq \beta \leq \alpha$. This in turn implies that the amount of j processed before t', $t' \in (y_1^{j-1}, y_2^{j-1})$, is larger in A^j than in Y^j which proves the lemma. □

We discuss some more properties of our schedule with respect to the yardstick schedule. We first argue that in going from A^k to A^{k+1} no new steps are created before x_{k+1}.

Lemma 5. *For every i where $a_i^{k+1} < x_{k+1}$ there is a p such that $a_i^{k+1} = a_p^k$.*

Proof. The proof follows from the way we schedule job $k+1$. Before time x_{k+1} we schedule job $k+1$ to an extent of α units in a time slot. Besides, when we redistribute the job over consecutive steps, then no new step is created. □

We use the above lemma for proving the following lemma which will be crucial for our analysis.

Lemma 6. *Consider a job j. In any schedule A^k, $k \geq j$, and for any i such that $a_i^k \leq x_j$, either job j begins at or after a_i^k or it is the case that in all slots between a_i^k and x_j, α units of j is scheduled.*

Proof. Suppose in the schedule A^j, job j begins at time a_p^j. By our method of scheduling j it follows that in all slots between a_{p+1}^j and x_j, α units of j is scheduled. Thus for all $i \leq p$ it is the case that job j begins at or after a_i^j while for $i > p$, all slots between a_i^j and x_j have α units of j.

When we go from schedule A^j to A^{j+1} then, by Lemma 5, we do not create any new steps before x_{j+1}. Since $x_j \leq x_{j+1}$ no new steps are created before x_j either. Further, we do not modify the schedule of job j and so the lemma continues to hold for the schedule A^{j+1} and in a similar manner for all subsequent schedules. □

Now, we are ready to show the correctness of our algorithm with respect to the processing capacity when given speed $\alpha = \alpha_m$. To do so, consider any release time t and the set of available unfinished jobs and their remaining processing times in *the yardstick schedule*. We show that our algorithm applied to these jobs assigns always at most αm units to time slots after t. By Lemma 4 this is only more than our algorithm actually has to schedule.

For the sake of contradiction, assume that this is not the case and let k be the first job for which we fail. This implies that the height of the first step in A^k exceeds αm. In this proof we will assume that $t = 0$. It is straightforward to extend the argument for arbitrary t. Let $p_j(t) = p_j$ be the remaining processing time of j in the yardstick schedule.

Let $a_1^k = z$. Consider the set of jobs scheduled in A^k. We will partition this set into four disjoint subsets. Define B to be the set of jobs, j, for which $f_j < z$ and C as the set of jobs, j, for which $x_j \le z < f_j$. The remaining jobs are the ones for which $z \le x_j$ and these we partition into two sets; D is the set of those jobs which begin at or after z in our schedule while E is the set of jobs which begin before z. Counting the total processing time of all jobs in two different ways we get

$$\sum_{j\in B\cup C\cup D\cup E} p_j > m\alpha z + \sum_{j\in C}(f_j - z) + \sum_{j\in D} p_j + \sum_{j\in E}((f_j - x_j) + \alpha(x_j - z)).$$

Note that since a job $j \in E$ begins before $z = a_1^k$ in the schedule A^k, by virtue of Lemma 6, α units of j would have been scheduled in each time slot between z and x_j. Rearranging terms now yields

$$\sum_{j\in B} p_j + \sum_{j\in C}(p_j - f_j + z) + \sum_{j\in E}(p_j - f_j + x_j) > \alpha(mz + \sum_{j\in E}(x_j - z))$$

If α was chosen such that the above inequality is not satisfied, then this would imply that our algorithm never uses more machine capacity than is available. Thus, chosing an α with

$$\alpha > \frac{\sum_{j\in B} p_j + \sum_{j\in C}(p_j - f_j + z) + \sum_{j\in E}(p_j - f_j + x_j)}{mz + \sum_{j\in E}(x_j - z)}$$

guarantees that our algorithm finds a feasible schedule for all jobs when given m machines of speed α. In the following we determine the smallest α that satisfies this condition.

Observe that $\sum_{j\in B} p_j + \sum_{j\in C}(p_j - f_j + z) < mz$.

For $j \in E$ define $b_j = (p_j - f_j + x_j)/z$ and $a_j = x_j/z$. Then

$$0 \le b_j \tag{1}$$

$$b_j \le a_j \tag{2}$$

$$a_j \ge 1 \tag{3}$$

Let us number the jobs in E from 1 to $k = |E| \le m$ in the order of their deadlines. Then for $1 \le j \le k$,

$$a_j \ge a_{j-1} + b_j/m \tag{4}$$

where $a_0 = 1$.

Hence it suffices to choose α as the optimal value of the optimization problem

$$\max_{a_i, b_i, 1\le i\le k} \left\{ \frac{m + \sum_{i=1}^k b_i}{m - k + \sum_{i=1}^k a_i} \,\middle|\, (1) - (4) \right\}. \tag{P}$$

Consider an optimal assignment of b_i, a_i, $1 \le i \le k$ for (P). It has the following property.

Lemma 7. *For every* $1 \le i \le k$, $a_i = a_{i-1} + b_i/m$ *and either* $b_i = a_i$ *or* $b_i = 0$.

Proof. Let p be the largest index for which the statement of the lemma is not true. For every $i \ge p$ we will determine an ϵ_i, δ_i so that the solution remains feasible when for all $i \ge p$ we set

$$a_i \leftarrow a_i + \epsilon_i$$
$$b_i \leftarrow b_i + \delta_i$$

and also when for all $i \ge p$ we set

$$a_i \leftarrow a_i - \epsilon_i$$
$$b_i \leftarrow b_i - \delta_i$$

If the original solution had value X/Y, then the first solution has value $(X+\epsilon)/(Y+\delta)$, where $\epsilon = \sum_{i=p}^{k} \epsilon_i$ and $\delta = \sum_{i=p}^{k} \delta_i$, while the second solution has value $(X-\epsilon)/(Y-\delta)$. If one of these solutions has value greater than that of the original solution then we would have obtained a contradiction.

Otherwise, both solutions have the same value. Our choice of ϵ_i, δ_i will be such that the condition of the lemma remains true for all $i > p$ in both solutions built. Further, in one of the two solutions we will satisfy both conditions for index p (if one was satisfied to begin with) or satisfy one of the conditions for index p (if none were satisfied to begin with). Thus by picking one of these two solutions, which, has the same value as our original solution we get closer to proving the lemma for all indices.

To determine ϵ_i, δ_i for $i = p$ we consider three cases.

$a_p > a_{p-1} + b_p/m, b_p < a_p$: Then $\delta_p = 0$ and $\epsilon_p = \min(a_p - b_p, a_p - (a_{p-1} - b_p/m))$.
$a_p > a_{p-1} + b_p/m, b_p = a_p$: Then $a_p > ma_{p-1}/(m-1)$ and hence $\delta_p = \epsilon_p = a_p - ma_{p-1}/(m-1)$.
$a_p = a_{p-1} + b_p/m, 0 < b_p < a_p$: Then $\delta_p = \min(b_p, a_p - b_p)$ and $\epsilon_p = \delta_p/m$.

The values for $i \ge p+1$ are determined by considering the following cases

$b_i = a_i$: Then $a_i = a_{i-1} + b_i/m = a_{i-1} + a_i/m$ which implies $a_i = ma_{i-1}/(m-1)$. Hence, $\delta_i = \epsilon_i = m\epsilon_{i-1}/(m-1)$.
$b_i = 0$: Then $\delta_i = 0$ and $\epsilon_i = \epsilon_{i-1}$. □

Let $i_1 < i_2 < \cdots < i_r$ be the indices, i for which $b_i = a_i$. It can be easily shown by induction that $a_0 = a_i = 1, i < i_1$, and $a_{i_1} = m/(m-1) = a_{i_2-1}, a_{i_2} = (m/(m-1))^2 = a_{i_3-1}$, and $a_{i_r} = (m/(m-1))^r$. Thus,

$$\sum_{i=1}^{k} b_i = \sum_{i=1}^{r} \left(\frac{m}{m-1}\right)^i.$$

In an optimal solution to (P), the sum $\sum_{i=1}^{k} a_i$ is minimized. This is the case, when the indices for which $b_i = 0$ are the lowest ones, i.e. $b_i = 0$, $1 \le k - r$. Then $a_0 = a_1 = \cdots = a_{k-r} = 1$, and thus,

$$\sum_{i=1}^{k} a_i = (k-r) + \sum_{i=1}^{r} \left(\frac{m}{m-1}\right)^i.$$

Hence the value of this solution is

$$\frac{m + \sum_{i=1}^{r} \left(\frac{m}{m-1}\right)^i}{m - k + k - r + \sum_{i=1}^{r} \left(\frac{m}{m-1}\right)^i} = \frac{m\left(\frac{m}{m-1}\right)^r}{m\left(\frac{m}{m-1}\right)^r - r} = \frac{m}{m - r\left(\frac{m-1}{m}\right)^r}.$$

Since $r \leq k \leq m$ and $r(1-1/m)^r < m(1-1/m)^m$, the above ratio is at most

$$\alpha_m = \frac{1}{1-(1-1/m)^m},$$

and this is our choice of α. For $m = 2$ this quantity equals $4/3$ and for large m, α is at most $e/(e-1)$, since $(1-1/m)^m < e^{-1}$. This upper bound combined with the lower bound on the speed requirement of any deadline ordered online algorithm [6] concludes the proof of our main result Theorem 1.

4 A Lower Bound for LLF

We give a lower bound on the speed that is necessary for LLF to schedule feasible instances. We first give a necessary condition.

Recall that $\ell_j(t)$ is the laxity of job j at time t, and $p_j(t)$ denotes the remaining processing time of j at this time.

Lemma 8. *Let $1 \leq s \leq 2 - 1/m$ be the speed required for LLF to be optimal. Consider an instance that is feasible for m unit speed machines and a time t by which all jobs released before t could have completed in a feasible schedule. If job j has not completed by time t in LLF on m speed-s machines, then*

$$\ell_j(t) \geq \frac{p_j(t)}{s(s-1)}, \tag{5}$$

Proof. Suppose that LLF does not satisfy condition (5) for some job j at time t. We show how to augment the current set of jobs with *blocking jobs* such that LLF will miss the deadline d_j or cannot schedule the blocking jobs feasibly. A set of *blocking jobs* consists of m jobs each having the same size and release time and 0 laxity.

Define the *relative laxity* $\ell_j^r(t)$ of a job as the ratio $\ell_j(t)/p_j(t)$. We first show that if condition (5) is not satisfied, we can decrease the relative laxity arbitrarily. When the relative laxity of the job is sufficiently small, we release m blocking jobs so that no matter how we schedule the jobs, we cannot finish all the jobs by their deadlines.

The procedure for decreasing the relative laxity of some job j is as follows. It is no loss of generality to assume that $p_j(t) = 1$. Since condition (5) is not satisfied, $\ell_j^r(t) = \ell_j(t) = \frac{k}{s(s-1)}$ with $k < 1$. Now we release m blocking jobs each of size q. These jobs will take up q/s time on each machine in the speed-s LLF schedule leaving $q - q/s$ time to process job j. Thus, at time $t' = t + q$, we will have $\ell_j(t') = \ell_j(t) - q/s$ and $p_j(t') = p_j(t) - s(q - q/s)$ whereas an optimal algorithm could have finished all the blocking jobs. We choose q such that the new relative laxity $\ell_j(t')/p_j(t')$ is half the original relative laxity $\ell_j(t)/1$, that is, $q = \frac{2-k}{k(s-1)}$.

Note that we do not release big blocking jobs of size q at one time. We release many small blocking jobs, m at a time, so that each job has laxity less than $\ell_j(t')$. The total size of these small jobs scheduled on a machine is q. This way we can ensure that j is never scheduled in parallel with the blocking jobs. Once these m jobs are finished in the optimal schedule, we release another batch of m blocking jobs.

With the above procedure, we can halve the relative laxity of job j. Repeating this process, we can reduce the relative laxity arbitrarily. Once the relative laxity $\ell_j^r(t'')$ is less than $p_j(t'')/2m$, we release m blocking jobs of size $p_j(t'')$. It is easy to see that these jobs cannot be scheduled within their deadlines. □

Theorem 2. *Let $x = \frac{m}{m-1}$. LLF is not optimal for speed less than*

$$\frac{1 + \sqrt{1 + 4x^2}}{2x},$$

which is $(1 + \sqrt{17})/4 \approx 1.281$ for $m = 2$, and tends to $(1 + \sqrt{5})/2 \approx 1.618$ for $m \to \infty$.

Proof. Let $1 \le s < 2 - 1/m$ be the speed of the machines available to LLF. We construct a worst case instance consisting of *main* and *blocking* jobs. We have $m + 1$ main jobs with $r_j = 0, p_j = 1, d_j = \frac{m}{m-1}$, for $j = 1, \ldots, m$, and $r_{m+1} = 0, p_{m+1} = \frac{m}{m-1}, d_{m+1} = \left(1 + \frac{1}{s}\right)\left(\frac{m}{m-1}\right)$.

There is a schedule S on m speed-1 machines in which the main jobs are completed by time $t = \frac{m}{m-1}$. Indeed, start the long job k at time 0 and schedule all other jobs in a round robin fashion on the remaining $m - 1$ machines.

In contrast, LLF with speed s schedules the small jobs in a round robin fashion on all m machines and completes them by time $t' = 1/s$. Only then the long job begins processing. To see that, observe that the laxity of the long job $\ell_{m+1}(t'') \ge \ell_1(t'') = \frac{m}{m-1} - \frac{1}{s}$ for any $t'' \le t'$. The remaining processing time for job $m+1$ at time $t = m/(m-1)$ is $p_{m+1}(t) = \frac{m}{m-1} - \left(\frac{m}{m-1} - 1/s\right)s = 1 - \left(\frac{m}{m-1}\right)(s-1)$. The lower bound on the speed requirement for LLF now follows directly from Lemma 8. □

5 A Lower Bound for EDZL

We show that the speed requirement of EDZL to guarantee an optimal schedule for a feasible instance is no less than the one for EDF.

Theorem 3. *EDZL is not optimal for speed $s = 2 - \frac{1}{m} - \epsilon$ for any $\epsilon > 0$.*

Proof. We first give the proof for $m = 2$ and show later how this can be generalized to arbitrary $m > 2$. At time $t = 0$, three jobs are released: two of them have size L and deadline $2L$. The remaining job (call this job j) has size $2L$ and deadline $3L/s$. At $t = 2L$, two jobs of size 1 and deadline $2L + 1$ are released. Assume that $L \gg 1$ so that $3L/s > xL + 1$.

We first note that there is a feasible schedule for this instance. Schedule job j on one machine and the other jobs in a round robin fashion on the other machine. All the jobs are finished by time $2L$. Then schedule the newly arrived jobs on separate machines.

EDZL will schedule the size L jobs on 2 machines till time L/s. At $t = L/s$, the size L jobs will be finished and job j has zero laxity. Since we assume that $L \gg 1$, one machine must be assigned to this job for the remainder of the schedule. Clearly the newly released jobs at time $t = 2L$ cannot both be scheduled on the other machine. Thus EDZL is not optimal for $s = 3/2 - \epsilon$ for $m = 2$.

For general m, let $x = \frac{m}{m-1}$. We release $m + 1$ jobs at time 0 with one job of size xL_1 and deadline $(x + 1)L_1/s$ and the remaining m jobs of size L_1 and deadline xL_1. Similar to the above analysis, at time $t = xL_1$, all jobs will be finished by an optimal offline schedule while there is one tight job in EDZL. We choose L_1 large enough so that one machine is tied up to this job for the rest of the schedule. We repeat the construction with $m-1$ machines with some large enough L_2 and continue until we reach the case $m = 2$ in which a job will fail its deadline. □

6 Concluding Remarks

As our main result we have introduced a new online algorithm which is optimal for speed $\alpha_m \leq e/(e-1) \approx 1.58$. This is the first significant improvement since the seminal results in [8]. Our algorithm is best possible in the class of deadline ordered algorithms with respect to speed resource augmentation. Nevertheless, this does not generally rule out online algorithms that are optimal for less speed. However, we showed that our algorithm outperforms all algorithms, deadline ordered and non-deadline ordered, for which provable upper bounds are known in the literature.

References

1. Cho, S., Lee, S.K., Ahn, S., Lin, K.-J.: Efficient real-time scheduling algorithms for multiprocessor systems. IEICE Transactions on Communications E85-B(12), 2859–2867 (2002)
2. Dertouzos, M.L.: Control robotics: The procedural control of physical processes. In: IFIP Congress, pp. 807–813 (1974)
3. Dertouzos, M.L., Mok, A.K.: Multiprocessor on-line scheduling of hard-real-time tasks. IEEE Transactions on Software Engineering 15(12), 1497–1506 (1989)
4. Horn, W.A.: Some simple scheduling algorithms. Naval Research Logistics Quarterly 21(1), 177–185 (1974)
5. Kalyanasundaram, B., Pruhs, K.: Speed is as powerful as clairvoyance. Journal of the ACM 47(4), 617–643 (2000)
6. Lam, T.W., To, K.-K.: Trade-offs between speed and processor in hard-deadline scheduling. In: Proc. of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 623–632 (1999)
7. Park, M., Han, S., Kim, H., Cho, S., Cho, Y.: Comparison of deadline-based scheduling algorithms for periodic real-time tasks on multiprocessor. IEICE Transactions 88-D(3), 658–661 (2005)
8. Phillips, C.A., Stein, C., Torng, E., Wein, J.: Optimal time-critical scheduling via resource augmentation. Algorithmica 32(2), 163–200 (2002)

Range Majority in Constant Time and Linear Space*

Stephane Durocher[1], Meng He[2], J. Ian Munro[2], Patrick K. Nicholson[2], and Matthew Skala[1]

[1] Department of Computer Science, University of Manitoba, Canada
[2] Cheriton School of Computer Science, University of Waterloo, Canada

Abstract. Given an array A of size n, we consider the problem of answering range majority queries: given a query range $[i..j]$ where $1 \leq i \leq j \leq n$, return the majority element of the subarray $A[i..j]$ if it exists. We describe a linear space data structure that answers range majority queries in constant time. We further generalize this problem by defining range α-majority queries: given a query range $[i..j]$, return all the elements in the subarray $A[i..j]$ with frequency greater than $\alpha(j-i+1)$. We prove an upper bound on the number of α-majorities that can exist in a subarray, assuming that query ranges are restricted to be larger than a given threshold. Using this upper bound, we generalize our range majority data structure to answer range α-majority queries in $O(\frac{1}{\alpha})$ time using $O(n \lg(\frac{1}{\alpha}+1))$ space, for any fixed $\alpha \in (0,1)$. This result is interesting since other similar range query problems based on frequency have nearly logarithmic lower bounds on query time when restricted to linear space.

1 Introduction

The *majority element*, or *majority*, of an array $A[1..n]$ is the element, if any, that occurs more than $\frac{n}{2}$ times in A. The *majority element problem* is to determine whether a given array has a majority element, and if so, to report that element. This problem is fundamental to data analysis and has been well studied. Linear time deterministic and randomized algorithms for this problem, such as the Boyer-Moore voting algorithm [4], are well known, and they are sometimes included in the curriculum of introductory courses on algorithms.

In this paper, we consider the data structure counterpart to this problem. We are interested in designing a data structure that represents an array $A[1..n]$ to answer *range majority queries*: given a query range $[i..j]$ where $1 \leq i \leq j \leq n$, return the majority element of the subarray $A[i..j]$ if it exists, and ∞ otherwise. Here we define the majority of a subarray $A[i..j]$ as the element whose *frequency* in $A[i..j]$, i.e., the number of occurrences of the element in $A[i..j]$, is more than half of the size of the interval $[i..j]$.

We further generalize this problem by defining the *α-majorities* of a subarray $A[i..j]$ to be the elements whose frequencies are more than $\alpha(j-i+1)$, i.e., α

* This work was supported by NSERC and the Canada Research Chairs program.

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 244–255, 2011.

times the size of the range $[i..j]$, for $0 < \alpha < 1$. Thus an *α-majority query* on array $A[1..n]$ can be defined as: given a query range $[i..j]$ where $1 \leq i \leq j \leq n$, return the α-majorities of the subarray $A[i..j]$ if they exist, and ∞ otherwise. A range α-majority query becomes a range majority query when $\alpha = \frac{1}{2}$.

For the case of range majority, we describe a linear space data structure that answers queries in constant time. We generalize this data structure to the case of range α-majority, yielding an $O(n \lg(\frac{1}{\alpha} + 1))$ space[1] data structure that answers queries in $O(\frac{1}{\alpha})$ time, for any fixed $\alpha \in (0, 1)$. Similar range query problems based on frequency are the range mode and k-frequency problems [8]. A *range mode query* for range $[i..j]$ returns an element in $A[i..j]$ that occurs at least as frequently as any other element. A *k-frequency query* for range $[i..j]$ determines whether any element in $A[i..j]$ occurs with frequency exactly k. Both of these problems have a lower bound that requires $\Omega(\frac{\lg n}{\lg \lg n})$ query time for any linear space data structure [8]. In light of this lower bound, it is interesting that a linear space data structure can answer range α-majority queries in constant time for fixed constant values of α.

1.1 Related Work

Computing the Mode, Majority, and Plurality of a Multiset. The mode of a multiset S of n items can be found in $O(n \lg n)$ time by sorting S and counting the frequency of each element. The decision problem of determining whether the frequency m of the mode exceeds one reduces to the element uniqueness problem, resulting in a lower bound of $\Omega(n \lg n)$ time [16]. Better bounds are obtained by parameterizing in terms of m: Munro and Spira [13] and Dobkin and Munro [6] describe $O(n \lg(\frac{n}{m}))$ time algorithms and corresponding lower bounds of $\Omega(n \lg(\frac{n}{m}))$ time. Misra and Gries [12] give $O(n)$ and $O(n \lg(\frac{1}{\alpha}))$ time algorithms for computing an α-majority when $\alpha \geq \frac{1}{2}$ and $\alpha < \frac{1}{2}$, respectively. The problem of computing α-majorities has also recently been studied in the approximate setting, using the term *heavy hitters* instead of α-majorities [5].

The *plurality* of a multiset S is a unique mode of S. That is, every multiset has a mode, but it might not have a plurality. The mode algorithms mentioned above can verify the uniqueness of the mode without any asymptotic increase in time. Numerous results establish bounds on the number of comparisons required for computing a majority, α-majority, mode, or plurality (e.g., [1,2,6,13]).

Range Mode, Frequency, and Majority Queries. Krizanc et al. [11] describe data structures that provide constant time range mode queries using $O(\frac{n^2 \lg \lg n}{\lg n})$ space and $O(n^\epsilon \lg n)$ time queries using $O(n^{2-2\epsilon})$ space, for any fixed $\epsilon \in (0, \frac{1}{2}]$. Petersen and Grabowski [15] improve the first bound to constant time and $O(\frac{n^2 \lg \lg n}{\lg^2 n})$ space. Petersen [14] and Durocher and Morrison [7] improve the second bound to $O(n^\epsilon)$ time and $O(n^{2-2\epsilon})$ space, for any fixed $\epsilon \in [0, \frac{1}{2})$. Durocher and Morrison [7] describe four $O(n)$ space data structures that return the mode

[1] In this paper $\lg n$ denotes $\log_2 n$.

of a query range $[i..j]$ in $O(\sqrt{n})$, $O(k)$, $O(m)$, and $O(|j-i|)$ time, respectively, where k denotes the number of distinct elements. Greve et al. [8] prove a lower bound of $\Omega(\frac{\lg n}{\lg(\frac{sw}{n})})$ query time for any range mode query data structure that uses s memory cells of w bits. Finally, various data structures support approximate range mode queries, in which the objective is to return an element whose frequency is at least ε times the frequency of the mode, for a fixed $\varepsilon \in (0,1)$ (e.g., [3,8]).

Greve et al. [8] examine the range k-frequency problem, in which the objective is to determine whether any element in the query range has frequency exactly k, where k is either fixed or given at query time. They note that when k is fixed a straightforward linear space data structure exists for determining whether any element has frequency at least k in constant time; determining whether any element has frequency *exactly* k requires a different approach. For any fixed $k > 1$, they describe how to support range k-frequency queries in $O(\frac{\lg n}{\lg \lg n})$ optimal time. When k is given at query time, Greve et al. show their lower bound of $\Omega(\frac{\lg n}{\lg \lg n})$ time applies to either query: exactly k or at least k.

The best result applicable to the range α-majority problem is that of Karpinski and Nekrich [10]. They study the problem in a geometric setting, in which points on the real line are assigned colors, and the goal is to find τ-*dominating colors*: given a range Q, return all the colors that are assigned to at least a τ fraction of the points in Q. If we treat each entry of an array $A[1..n]$ as a point in a bounded universe $[1,n]$, their data structure can be used to represent A in $O(\frac{n}{\alpha})$ space to support range α-majority queries in $O(\frac{(\lg \lg n)^2}{\alpha})$ time.

1.2 Our Results

Our results can be summarized as follows:

- In Section 2 we present a data structure for answering range majority queries in the word-RAM model with word size $\Omega(\lg n)$. It uses $O(n)$ words and answers range majority queries in constant time. The data structure is conceptually simple and based on the idea that, for query ranges above a certain size threshold, only a small set of *candidate* elements need be considered in order to determine the majority. In order to verify the frequency of these elements efficiently we present a novel decomposition technique that uses wavelet trees [9].
- In Section 3 we generalize our data structure to answer range α-majority queries, for any fixed $\alpha \in (0,1)$. Note that although α is fixed, it is not necessarily a constant. For example, setting $\alpha = \frac{1}{\lg n}$ is permitted. Our structure uses $O(n \lg(\frac{1}{\alpha}+1))$ words and answers range α-majority queries in $O(\frac{1}{\alpha})$ time. The first part of the section proves an upper bound on the number of potential range α-majority values that need be stored by our data structure. These bounds are of independent interest, and are tight for the case of $\alpha = \frac{1}{2}$. In order to generalize our data structure when $\frac{1}{\alpha}$ is large, i.e., when $\frac{1}{\alpha} = \omega(1)$, we make use of *batched* queries over wavelet trees.

2 Range Majority Data Structure

In this section we describe a linear space data structure that supports range majority queries in constant time. To provide some intuition, suppose we partition the input array $A[1..n]$ into four contiguous equally sized blocks. If we are given a query range that contains one of these four blocks, then it is clear that a majority element for this query must have frequency greater than $\frac{n}{8}$ times in A. Thus, at most seven elements need be considered when computing the majority for queries that contain an entire block.

Of course, not all queries contain one of these four blocks. Therefore, we decompose the array into multiple levels in order to support arbitrary queries (Sections 2.1 and 2.2). Using this decomposition in conjunction with succinct data structures, we design a linear space data structure that answers range majority queries in constant time (Section 2.3). The data structure works by counting the frequency of a constant number of *candidate* elements in order to determine the majority element for a given query. While a loose bound on the number of candidates that need be considered suffices to show that our data structures occupy linear space, it is more challenging to prove a tighter bound, such as that of Section 3.

2.1 Quadruple Decomposition

The first stage of our decomposition is to construct a notional complete binary tree T over the range $[1..n]$, in which each node represents a subrange of $[1..n]$. Let the root of T represent the entire range $[1..n]$. For a node corresponding to range $[a..b]$, its left child represent the left half of its range, i.e., the range $[a..\lfloor\frac{(a+b)}{2}\rfloor]$, and its right child represents the right half, i.e., the range $[\lfloor\frac{(a+b)}{2}\rfloor + 1..b]$. For simplicity, we assume that n is a power of 2. Each leaf of the tree represents a range of size 1, which corresponds to a single index of the array A. We refer to ranges represented by the nodes of T as *blocks*. Note that the tree T is for illustrative purposes only, so we need not store it explicitly.

The tree T has $\lg n + 1$ levels, which are numbered 0 through $\lg n$ from top to bottom. For each level ℓ, T partitions A into 2^ℓ blocks of size $\frac{n}{2^\ell}$. Let $T(\ell)$ denote the set of blocks at level ℓ in T.

The second stage of our decomposition consists of arranging adjacent blocks within each level $T(\ell)$, $2 \leq \ell \leq \lg n$, into groups. Each group consists of four blocks and is called a *quadruple*. Formally, we define a quadruple U_q to be a range $[a..b]$ at level $\ell \geq 2$ of size $\frac{4n}{2^\ell}$, where $a = \frac{2(q-1)n}{2^\ell} + 1$ and $b = \frac{2(q-1)n}{2^\ell} + \frac{4n}{2^\ell}$, for $1 \leq q \leq 2^{\ell-1} - 1$. In other words, each quadruple at level ℓ contains exactly 4 consecutive blocks, and its starting position is separated from the starting position of the previous quadruple by 2 blocks. To handle border cases, we also define an extra quadruple $U_{2^{\ell-1}}$ which contains both the first two and last two blocks in $T(\ell)$. Thus, at level ℓ there are $2^{\ell-1}$ quadruples, and each block in $T(\ell)$ is contained in two quadruples. These definitions are summarized in Figure 1.

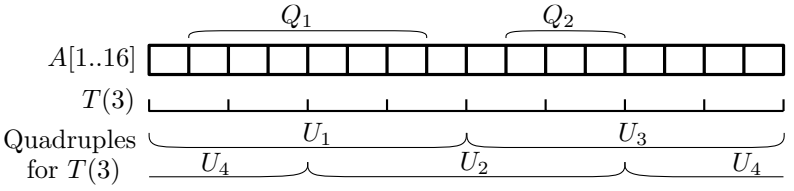


Fig. 1. An example where $n = 16$. Blocks in $T(3)$ have size 2, and each of the 4 quadruples contain 4 blocks. Query ranges Q_1 and Q_2 are associated with quadruples U_1 and U_3 respectively.

2.2 Candidates

Based on the decomposition from the previous section, we observe the following:

Observation 1. *For every query range Q there exists a unique level ℓ such that Q contains at least one and at most two consecutive blocks in $T(\ell)$, and, if Q contains two blocks, then the nodes representing these blocks are not siblings in the tree T.*

Let U be a quadruple consisting of four consecutive blocks, B_1 through B_4 from $T(\ell)$, where ℓ is the level referred to in the previous observation. We *associate* Q with U if Q contains B_2 or B_3; for convenience we also say that Q is associated with level ℓ. Note that Q may contain both B_2 and B_3; see Q_1 in Figure 1. The following lemma can be proved by an argument analogous to that described at the beginning of Section 2:

Lemma 1. *There exists a set C of at most 7 elements such that, for any query range Q associated with quadruple U, the majority element for Q is in C.*

For a quadruple U, we define the set of *candidates* for U to be the elements in C. In Section 3.2 we improve the upper bound on $|C|$ from 7 to 5, which, as illustrated by the following example, is tight.

Example 1. Let U be a quadruple containing 4 blocks, each of size 32, and $(e)^y$ denote a sequence of y occurrences of the element e. In ascending order of starting position, the first block begins with an arbitrary element and is followed by $(e_1)^{28}$ and $(e_2)^3$. The second block contains $(e_2)^{15}$, and $(e_3)^{17}$. The third block contains $(e_1)^8$, $(e_4)^{17}$, and $(e_5)^7$. The final block contains $(e_5)^{19}$, followed by any arbitrary sequence of elements. Assume the range contained by the quadruple is [1..128]. The queries [2..72], [30..64], [33..64], [65..96] and [65..115] are all associated with U, and have e_1 through e_5 as majority elements respectively.

The elements in C can be found in $O(|U|)$ time; complete details will appear in a later version of this paper. This implies that the sets of candidates for all the quadruples in all of the $\lg n + 1$ levels can be found in $O(n \lg n)$ time.

2.3 Data Structures for Counting

We now describe the data structures for each level ℓ of the tree T, for $2 \leq \ell \leq \lg n$. Given a quadruple U_q in level ℓ, for $1 \leq q \leq 2^{\ell-1}$ we store the set of candidates for U_q in an array F_q. Let Y_q be a string of length $|U_q|$, where the i-th symbol in Y_q is f if the i-th symbol in U_q is $F_q[f]$, and a unique symbol otherwise. Let Y be the concatenation of the strings Y_1 through $Y_{2^{\ell-1}}$. We use the *wavelet tree* data structure [9] to represent Y, which has alphabet size $\sigma' = |F_q| + 1 \leq 6$. This representation uses $O(n)$ bits to provide constant time support for the operation $\texttt{rank}_c(Y, i)$, which returns the number of occurrences of the character c in $Y[1..i]$.

Theorem 1. *Given an array $A[1..n]$, there exists an $O(n)$ word data structure that supports range majority queries on A in $O(1)$ time, and can be constructed in $O(n \lg n)$ time.*

Proof. Given a query $Q = [a..b]$, we first want to find the level ℓ and the index q of the quadruple U_q with which Q is associated. This can be reduced to finding the length of the longest common prefix of the $(\lg n)$-bit binary representations of a and b, which can be done in constant time using a lookup table of $o(n)$ bits. We only show how to answer queries associated with a quadruple at levels ℓ, for $2 \leq \ell \leq \lg n$; the case in which $0 \leq \ell \leq 1$ can be handled similarly.

The representation of quadruple U_q in Y begins at $s = \frac{4(q-1)n}{2^\ell} + 1$. Let $t = \frac{2(q-1)n}{2^\ell} + 1$. For each f in $[1..|F_q|]$, we count the frequency of $F_q[f]$ in $[a..b]$ using $\texttt{rank}_f(Y, s + b - t) - \texttt{rank}_f(Y, s + a - 1 - t)$, and report $F_q[f]$ if it is a majority. Since Y has a constant sized alphabet, this process takes $O(1)$ time.

In addition to the input array, we must store the arrays F_q for each of the $O(n)$ quadruples, and each array requires a constant number of words. For each of the $\lg n + 1$ levels in T we store a wavelet tree on an alphabet of size $\sigma' \leq 6$, requiring $O(n \lg n)$ bits. To answer queries in constant time, we require $o(n)$ bits of additional space for a lookup table to determine ℓ and q. Thus, the additional space requirements beyond the input array are $O(n)$ words. □

3 Generalization to Range α-Majority Queries

In this section we provide an upper bound on the number of candidates we need from each quadruple to support α-majority queries (Section 3.2). Using this upper bound, we are able to generalize Theorem 1 to the case of α-majority queries (Section 3.3).

3.1 Definitions

We refer to the range $[a..b']$, where $a \leq b' \leq b$, as a *prefix* of the range $[a..b]$. Similarly, the range $[a'..b]$, where $a \leq a' \leq b$, is a *suffix* of $[a..b]$. For a block $L \in T(\ell)$, we refer to the *successor* of L, which is the block $L_s \in T(\ell)$ such that the range represented by L_s immediately follows the range represented by L. The *predecessor* is defined analogously.

Consider a query $[a..b']$ that contains block $L = [a..b] \in T(\ell)$, $b \leq b' < b + |L|$. Thus, $[a..b']$ contains L and a prefix of the successor of L. We refer to a query of this form as a *prefix query*. We refer to the symmetric case, where a query $[a'..b]$ contains L and $a - |L| < a' \leq a$ as a *suffix query*. Finally, let $|A[i..j]|_t$ denote the frequency of an element t in $A[i...j]$.

3.2 Relaxed Triples

Suppose we are given a block L, where L_p and L_s are the predecessor and successor of L respectively; we call $L_p \cup L \cup L_s$ a *triple*. We relax the restriction that blocks in the triple have equal size, and only require that $|L_p| + |L_s| \leq 2|L|$. Furthermore, we also relax the restriction that blocks and occurrences of elements are of integer size; i.e., the ranges described in this section may start and end at arbitrary real numbers. Although the ranges are real-valued, we still refer to "occurrences" of elements. Thus, in the continuous setting described in this section, an occurrence of an element may contain an arbitrary fraction of a block; for example, inside a block there may be a contiguous range of occurrences of element e that has length 5.22. We refer to these generalized triples as *relaxed triples*.

Let $e_1, ..., e_m$ denote the m distinct α-majorities that exist for a query Q where $L \subseteq Q \subset (L_p \cup L \cup L_s)$; i.e. Q is a query contained in the relaxed triple and Q contains L. For brevity, whenever we refer to a query in the context of a relaxed triple, it is assumed to have this form. Let $\mathcal{Q} = \{Q_1, ..., Q_m\}$ be a set of queries within a relaxed triple such that Q_i is the smallest query for which e_i is an α-majority, breaking ties by taking the query with smallest starting position. We refer to $\mathcal{Q}$ as the *canonical query set* for the relaxed triple. If query Q_i is a prefix query or a suffix query we refer to it as *one-sided*. If Q_i is not one-sided, then it is *two-sided*. Note that the query $Q_i = L$ is one-sided, since it is both a suffix *and* a prefix query.

For two-sided canonical queries $Q_i \in \mathcal{Q}$, the element at both the starting position and ending position of Q_i must be e_i; otherwise we could reduce the size of Q_i. Thus, for all two-sided canonical queries $Q_i \in \mathcal{Q}$, no $Q_j \in \mathcal{Q}$ $(j \neq i)$ exists having the same starting or ending position as Q_i. However, there may be several occurrences of the query L in $\mathcal{Q}$, since many elements can share that particular range as a canonical query. From this point on we only consider relaxed triples where element e_i occurs only within the range Q_i for $1 \leq i \leq m$. Since the goal of this section is to find an upper bound on m, occurrences of e_i outside range Q_i can be removed without decreasing m.

Lemma 2. *Given a relaxed triple and its canonical query set $\mathcal{Q} = \{Q_1, ..., Q_m\}$, the elements $\{e_1, ..., e_m\}$ associated with $\mathcal{Q}$ can be rearranged such that they each appear in at most two contiguous ranges in the relaxed triple. This reordering induces a new canonical query set $\mathcal{Q}' = \{Q'_1, ..., Q'_m\}$, such that $|Q'_j| \leq |Q_j|$ for all $1 \leq j \leq m$.*

Proof. First, we describe a procedure for reordering the elements in L_p. Let $L'_p = L_p$, $\mathcal{Q}' = \mathcal{Q}$, and $Q_b \in \mathcal{Q}'$ be the query with the smallest starting position

in L'_p. Then Q_b contains a non-empty suffix of L'_p; if no such query exists, then L'_p is empty and we are done. Let e_b be the element associated with Q_b. We swap the positions of all the occurrences of e_b in L'_p such that they occupy a prefix P of Q_b. All elements that were in P are shifted toward L. Thus, it may be possible to reduce the size of a query $Q_i \in \mathcal{Q}'$ that originally had a starting position in P, and we recompute $\mathcal{Q}'$. Let L'_p be the largest suffix of L_p that does not contain any occurrences of e_b. At this point we recurse and compute the next Q_b.

After we have finished moving e_b, at no point later in the procedure will an occurrence of e_b in L_p be touched. At the end of the procedure each element in L_p that is associated with a canonical query will occupy a contiguous block. Furthermore, $|\mathcal{Q}'| = |\mathcal{Q}|$, since moving elements in P closer to the ending position of L_p will not decrease the ratio of their frequency to canonical query size. The procedure for reordering L_s is identical, though we process the elements in decreasing order by ending position.

After executing the procedure on L_p and L_s, consider an element e_i associated with Q_i. We can delete all k occurrences of e_i in L and insert k copies of e_i immediately before the first occurrence of e_i in L_s. This does not change the relative order of any other elements in the relaxed triple, and shifts all other elements in L_s in positions before the new first occurrence of e_i closer to L. Thus, each element appears in at most two contiguous ranges. □

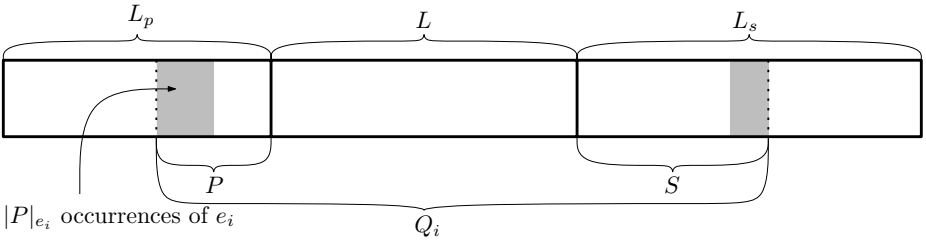


Fig. 2. Illustration of the relaxed triple using notation from Step 1 in Lemma 3

Lemma 3. *Given a relaxed triple and its canonical query set* $\mathcal{Q} = \{Q_1, ..., Q_m\}$, *we can rearrange its elements, creating a new relaxed triple that has a canonical query set* $\mathcal{Q}' = \{Q'_1, ..., Q'_m\}$ *such that* Q'_i *is one-sided for* $1 \leq i \leq m$.

Proof. We describe a procedure for rearranging the elements in the relaxed triple.

Step 1: Choose an arbitrary two-sided query, $Q_i \in \mathcal{Q}$. We apply Lemma 2 to the triple, such that all occurrences of e_i appear in the prefix and suffix of Q_i. Let P represent the prefix of Q_i contained in L_p and S the suffix of Q_i contained in L_s. P is contained in $c \geq 0$ queries in $\mathcal{Q}$, distinct from Q_i, and S is contained in $d \geq 0$ queries in $\mathcal{Q}$, distinct from Q_i. Without loss of generality, assume $c \geq d$.

Note that $|L|_{e_i} = \alpha|L| - \Delta_L$ for $\Delta_L \geq 0$; otherwise L would be the canonical query for e_i. Since $|Q_i|_{e_i} > \alpha(|L| + |P| + |S|)$, we have $|P|_{e_i} = \alpha|P| + \Delta_P$, and $|S|_{e_i} = \alpha|S| + \Delta_S$, where $\Delta_P + \Delta_S > \Delta_L$. Note that $\Delta_P > 0$ and $\Delta_S > 0$; if $\Delta_P \leq 0$, then $S \cup L$ would be the canonical query for e_i, and the same argument applies to Δ_S. This implies that $|P| \geq \frac{\Delta_P}{1-\alpha}$ and $|S| \geq \frac{\Delta_S}{1-\alpha}$. See Figure 2.

Step 2: We remove all $|P|_{e_i} = \alpha|P| + \Delta_P \geq \frac{\Delta_P}{1-\alpha}$ occurrences of e_i from L_p. This shifts the starting position of c queries in $\mathcal{Q}$ closer to L. Let Q_j be the innermost of the c queries, i.e., Q_j has the largest starting position of the c queries. Since there were no occurrences of e_j in the removed block, in order for e_j to be an α-majority for Q_j, there must have been at least f occurrences of e_j to pay for the removed block, where $f = \alpha(|P|_{e_i} + f)$. This implies $f = |P|_{e_i} \frac{\alpha}{1-\alpha}$. Generalizing this formula to consider the number of occurrences of the c different elements required to pay for the removed block, as well as the payments made by the innermost queries, we get a recurrence relation. Let f_i be the savings of the i-th innermost of the c queries. It follows that $f_i = \frac{\alpha}{1-\alpha}(\delta_p + \sum_{j=1}^{i-1} f_j)$, for $1 \leq i \leq c$. Thus, we have reduced the size of L_p by the total sum $\delta_p + \sum_{i=1}^{c} f_c$.

Step 3: We insert $\frac{\Delta_P}{1-\alpha} \leq |P|_{e_i}$ elements immediately after the last occurrence of e_i in S. After this, there exists a prefix query on the relaxed triple which returns e_i as a majority. This insertion causes the ending positions of d queries in $\mathcal{Q}$ to be shifted farther from L. By the same argument as before, we must insert at most $\sum_{i=1}^{d} f_d$ elements in order to correct for this shift. Since $c \geq d$, our new arrangement satisfies the constraint $|L_p| + |L_s| \leq 2|L|$, and is therefore a relaxed triple.

Step 4: We reorder the elements according to Lemma 2 and recompute the canonical query set. The procedure described in the proof of Lemma 2 does not increase the number of two-sided queries. If any two-sided queries remain, then go to step 1.

After rearranging element e_i, Q_i will remain one-sided in any future iteration of the procedure; no occurrence of e_i will subsequently be moved back to L_p. Each iteration guarantees that e_i will be an α-majority for a one-sided query, and that the size of the canonical set remains unchanged. □

Remark 1. We note that Lemma 3 only holds in the continuous setting where we can manipulate fractional parts of elements. For an example where Lemma 3 does not hold in the discrete setting, consider the case where $|L| = |L_p| = |L_s| = 3$, and $L_p = \{e_5, e_5, e_4\}$, $L = \{e_1, e_2, e_3\}$, $L_s = \{e_4, e_6, e_6\}$, and $\frac{2}{7} < \alpha < \frac{1}{3}$. In this example, we cannot rearrange the triple such that Q_4 is one-sided, without decreasing the size of the canonical query set.

We have shown that to give an upper bound on the number of candidates in a relaxed triple, it suffices to examine the worst case restricted to prefix and suffix queries in the successor and predecessor of L, respectively. Without loss of generality, we consider the successor, then prove an upper bound on the size of the canonical query set in a relaxed triple. First, we require the following recurrence relation; the proof will appear in a later version of this paper:

Lemma 4. *If* $d_j = \frac{\alpha}{1-\alpha}(1 + \sum_{i=1}^{j-1}(1 + d_i))$ *for* $j \geq 1$, *then* $d_j = \frac{1}{(1-\alpha)^j} - 1$.

Next, we bound the number of candidates for prefix queries over a relaxed triple.

Lemma 5. *Let* L *be a block and* L_s *its successor in a relaxed triple. There exists a set of elements* C, *of size less than*

$$\left\lceil \frac{1}{\alpha} \right\rceil + \frac{\lg(1 + \frac{|L_s|}{|L|})}{\lg \frac{1}{1-\alpha}} \ ,$$

such that for all prefix queries Q *containing* L, *all* α*-majorities for* Q *are contained in* C.

Proof. We keep the set $F = \{f_1, ..., f_h\}$ of the $h = \lceil \frac{1}{\alpha} \rceil$ most frequently occurring elements from the block $L = [a..b]$. Let prefix query $Q_1 = [a..b_1]$, where $b_1 = b$ initially, and increase b_1 until a new element $e_1 \notin F$ becomes an α-majority for Q_1. We continue this process k times, where k is a value determined later: for $1 \leq i \leq k$, define $Q_i = [a..b_i]$, where $b_i = b_{i-1}$ initially, and b_i is increased until a new element $e_i \notin F \cup \{e_1, ..., e_{i-1}\}$ becomes an α-majority. Let R_i be the largest prefix of L_s contained in Q_i, and $d_i = |R_i| = \frac{b_i - b}{|L|}$ for $1 \leq i \leq k$. In order for Q_i to be a prefix query, $0 < d_i < \frac{|L_s|}{|L|}$ must hold for each $1 \leq i \leq k$. We want to determine the maximum value of k for which $d_k < \frac{|L_s|}{|L|}$ for the specific value of α. The value $h + k$ provides an upper bound on the number of elements we need examine to determine the α-majorities for any prefix query.

To maximize k, assume that all elements in F are α-majorities for the query $Q' = L$. Applying Lemma 2, each element f_i appears in a contiguous block within L. Note that any extra occurrences of f_i can be removed without decreasing k.

With the exception of at most one element e_{k+1}, we can assume $L \cup L_s$ only contains elements e' for which there exists some prefix query that returns e' as an α-majority; otherwise, we could replace all occurrences of these elements with e_{k+1}. We have filled L entirely with elements in F, and each element e_i can only occur in a single contiguous block in L_s, for $1 \leq i \leq k$, by Lemmas 2 and 3. Thus, each e_i appears in a contiguous block immediately following e_{i-1}.

Now we have an upper bound, $|R_j|_{e_j} \leq d_j|L| - \sum_{i=1}^{j-1} |R_i|_{e_i}$, and a lower bound, $|R_j|_{e_j} > \alpha(1 + d_j)|L| - |L|_{e_j}$, for $1 \leq j \leq k$. By our construction, $|L|_{e_j} = 0$ for all $1 \leq j \leq k$. Rearranging the upper and lower bounds, we get that $d_k > \frac{\alpha}{1-\alpha} + \sum_{i=1}^{k-1} \frac{|R_i|_{e_i}}{|L|(1-\alpha)}$, which implies that $d_k > \frac{\alpha}{1-\alpha} + \sum_{i=1}^{k-1} \frac{\alpha(1+d_i)}{(1-\alpha)}$. By Lemma 4, $d_k > \frac{1}{(1-\alpha)^k} - 1$. Since $\frac{|L_s|}{|L|} > d_k$, this is equivalent to the statement $1 + \frac{|L_s|}{|L|} > \frac{1}{(1-\alpha)^k}$. After isolating k we get that $k < \left(\lg\left(1 + \frac{|L_s|}{|L|}\right)\right) / \left(\lg \frac{1}{1-\alpha}\right)$. □

Extending the above lemma to arbitrary queries on relaxed triples yields the following lemma:

Lemma 6. *The canonical query set $\mathcal{Q}$ of any relaxed triple has size less than*

$$\left\lceil \frac{1}{\alpha} \right\rceil + \frac{2}{\lg \frac{1}{1-\alpha}} \ .$$

Proof. We consider the worst case in both predecessor and successor of L, noting that the contents of L are shared. We apply Lemmas 3 and 5 to L_p and L_s. Recall the constraint $|L_p| + |L_s| \leq 2|L|$, and note that the expression $\lceil \frac{1}{\alpha} \rceil + \left(\lg \left(1 + \frac{|L_s|}{|L|} \right) + \lg \left(1 + \frac{|L_p|}{|L|} \right) \right) / \left(\lg \frac{1}{1-\alpha} \right)$ is maximized when $|L_s| = |L_p| = |L|$. □

We extend Lemma 6 to the case of quadruples. The complete details of the proof will appear in a later version of this paper.

Theorem 2. *For any quadruple U there exists a set C such that*

$$|C| < 2 \left\lceil \frac{1}{\alpha} \right\rceil + \frac{2}{\lg \frac{1}{1-\alpha}} \ ,$$

and for any Q associated with U, all α-majorities for Q are in C.

3.3 Handling Large Alphabets

Now that we have an upper bound on the number of candidates required to answer α-majority queries, we can generalize Theorem 1. For a given α, if the number of candidates, $|C|$, required by Theorem 2 is $\omega(1)$, then we use the following observation about executing batched rank queries on a wavelet tree.

Observation 2. *A string $S[1..n]$ over alphabet $[\sigma]$, where $\sigma \leq n$, can be represented using a wavelet tree such that given an index i, the results of $\texttt{rank}_f(S, i)$ for all $f = 1, 2, \cdots, \sigma$ can be computed in $O(\sigma)$ time.*

With the above observation we present the following theorem:

Theorem 3. *Given an array $A[1..n]$ and any fixed $\alpha \in (0, 1)$, there exists an $O(n \lg(\frac{1}{\alpha} + 1))$ word data structure that supports range α-majority queries on A in $O(\frac{1}{\alpha})$ time, and can be constructed in $O(\frac{n \lg n}{\alpha})$ time.*

Proof. Based on Theorems 1, 2 and Observation 2 the query time follows, so we focus on analyzing the space. We observe that if $\alpha < \frac{1}{4}$, then we need not keep data structures at level $\lg n$ in T, since every distinct element contained in a query range, Q, associated with this level is a $(\frac{1}{4} - \varepsilon)$-majority for Q, for $0 < \varepsilon < \frac{1}{4}$. Instead, we perform a linear scan of the query range in $O(\frac{1}{\alpha})$ time, returning all the distinct elements. Continuing this argument, we observe that we only require the array F_q, for quadruple q, if q is in the top $O(\lg n - \lg \frac{1}{\alpha})$ levels in T. Since there are $O(n\alpha)$ quadruples in these levels, the arrays require $O(n\alpha \times \frac{1}{\alpha} \lg n) = O(n \lg n)$ bits in total. The overall space required for the wavelet tree data structures is $O(n \lg(\frac{1}{\alpha} + 1) \times \lg n)$ bits, and this term dominates the overall space requirements. We defer the details of the construction time to a later version of this paper. □

4 Concluding Remarks

We have presented an $O(n)$ word data structure that answers range majority queries in constant time, and an $O(n \lg(\frac{1}{\alpha}+1))$ word data structure that answers range α-majority queries in $O(\frac{1}{\alpha})$ time, for any fixed $\alpha \in (0,1)$. It would be interesting to determine if the space bound of $O(n \lg(\frac{1}{\alpha}+1))$ words can be improved, while maintaining the $O(\frac{1}{\alpha})$ query time.

References

1. Aigner, M.: Variants of the majority problem. Discrete Applied Mathematics 137, 3–25 (2004)
2. Alonso, L., Reingold, E.M.: Determining plurality. ACM Transactions on Algorithms 4(3), 26:1–26:19 (2008)
3. Bose, P., Kranakis, E., Morin, P., Tang, Y.: Approximate range mode and range median queries. In: Diekert, V., Durand, B. (eds.) STACS 2005. LNCS, vol. 3404, pp. 377–388. Springer, Heidelberg (2005)
4. Boyer, R.S., Moore, J.S.: MJRTY - A fast majority vote algorithm. In: Boyer, R.S. (ed.) Automated Reasoning: Essays in Honor of Woody Bledsoe. Automated Reasoning Series, pp. 105–117. Kluwer, Dordrecht (1991)
5. Cormode, G., Muthukrishnan, S.: An improved data stream summary: the count-min sketch and its applications. Journal of Algorithms 55(1), 58–75 (2005)
6. Dobkin, D., Munro, J.I.: Determining the mode. Theoretical Computer Science 12(3), 255–263 (1980)
7. Durocher, S., Morrison, J.: Linear-space data structures for range mode query in arrays (2011), arXiv:1101.4068v1
8. Greve, M., Jørgensen, A.G., Larsen, K.D., Truelsen, J.: Cell probe lower bounds and approximations for range mode. In: Abramsky, S., Gavoille, C., Kirchner, C., Meyer auf der Heide, F., Spirakis, P.G. (eds.) ICALP 2010. LNCS, vol. 6198, pp. 605–616. Springer, Heidelberg (2010)
9. Grossi, R., Gupta, A., Vitter, J.S.: High-order entropy-compressed text indexes. In: Proc. of the 14th Symposium on Discrete Algorithms, pp. 841–850 (2003)
10. Karpinski, M., Nekrich, Y.: Searching for frequent colors in rectangles. In: Proc. of the 20th Canadian Conference on Computational Geometry, pp. 11–14 (2008)
11. Krizanc, D., Morin, P., Smid, M.: Range mode and range median queries on lists and trees. Nordic Journal of Computing 12, 1–17 (2005)
12. Misra, J., Gries, D.: Finding repeated elements. Science of Computer Programming 2(2), 143–152 (1982)
13. Munro, J.I., Spira, M.: Sorting and searching in multisets. SIAM Journal on Computing 5(1), 1–8 (1976)
14. Petersen, H.: Improved bounds for range mode and range median queries. In: Geffert, V., Karhumäki, J., Bertoni, A., Preneel, B., Návrat, P., Bieliková, M. (eds.) SOFSEM 2008. LNCS, vol. 4910, pp. 418–423. Springer, Heidelberg (2008)
15. Petersen, H., Grabowski, S.: Range mode and range median queries in constant time and sub-quadratic space. Information Processing Letters 109, 225–228 (2009)
16. Skiena, S.: The Algorithm Design Manual, 2nd edn. Springer, Heidelberg (2008)

Dynamic Planar Range Maxima Queries

Gerth Stølting Brodal and Konstantinos Tsakalidis

MADALGO*,
Department of Computer Science,
Aarhus University, Denmark
{gerth,tsakalid}@madalgo.au.dk

Abstract. We consider the dynamic two-dimensional maxima query problem. Let P be a set of n points in the plane. A point is *maximal* if it is not dominated by any other point in P. We describe two data structures that support the reporting of the t maximal points that dominate a given query point, and allow for insertions and deletions of points in P. In the pointer machine model we present a linear space data structure with $O(\log n + t)$ worst case query time and $O(\log n)$ worst case update time. This is the first dynamic data structure for the planar maxima dominance query problem that achieves these bounds in the worst case. The data structure also supports the more general query of reporting the maximal points among the points that lie in a given 3-sided orthogonal range unbounded from above in the same complexity. We can support 4-sided queries in $O(\log^2 n + t)$ worst case time, and $O(\log^2 n)$ worst case update time, using $O(n \log n)$ space, where t is the size of the output. This improves the worst case deletion time of the dynamic rectangular visibility query problem from $O(\log^3 n)$ to $O(\log^2 n)$. We adapt the data structure to the RAM model with word size w, where the coordinates of the points are integers in the range $U = \{0, \ldots, 2^w - 1\}$. We present a linear space data structure that supports 3-sided range maxima queries in $O(\frac{\log n}{\log \log n} + t)$ worst case time and updates in $O(\frac{\log n}{\log \log n})$ worst case time. These are the first sublogarithmic worst case bounds for all operations in the RAM model.

1 Introduction

Given a set P of n points in the plane, a point $p = (p_x, p_y)$ *dominates* another point $q = (q_x, q_y)$ if both $p_x \geq q_x$ and $p_y \geq q_y$ hold. The m points that are not dominated by any other point in the set are called *maximal*. The maximal points are also called the *staircase* of the set, since when they are sorted by increasing x-coordinate they are also sorted by decreasing y-coordinate. We consider the problem of designing a data structure that supports insertions and deletions of points in the set, and allows reporting the maximal points that dominate a given query point q (*maxima dominance* query). Actually we consider the more

* Center for Massive Data Algorithmics - a Center of the Danish National Research Foundation.

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 256–267, 2011.

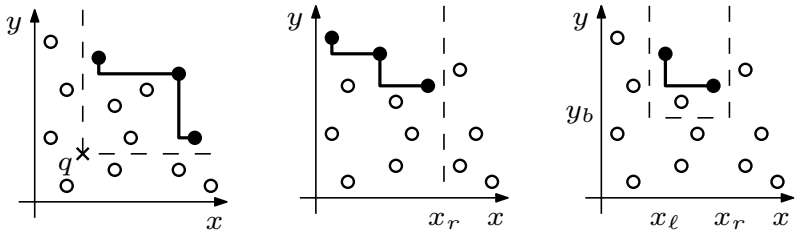


Fig. 1. Different range maxima queries. Black points are reported.

general *3-sided range maxima* queries. That is, given parameters x_ℓ, x_r and y_b, to report the maximal points of the points in the set $P \cap ([x_\ell, x_r] \times [y_b, +\infty[$. The special case where the query range is $]-\infty, x_r] \times]-\infty, +\infty[$ is also known as *maxima contour* queries.

Previous Results. The maximal points of a static set of two-dimensional points can be computed in optimal $O(n \log n)$ time [2].[1] In the pointer machine linear space dynamic data structures have been presented that support reporting all maximal points. They achieve $O(m)$ worst case [8,9,3,13] or amortized [15] query time. They support deciding whether a given query point lies above or below the staircase (*maxima emptiness* query) in $O(\log n)$ [8,9,15] or $O(\log m)$ [3,13] worst case time. Maxima contour queries are supported by the structures of [3] and [9] in $O(\log n + t)$ worst case time, where t is size of the output.

Regarding updates, the structure of Overmars and van Leeuwen [3] supports the insertion and deletion of an arbitrary point in $O(\log^2 n)$ worst case time. Frederickson and Rodger [8], and independently Janardan [9], improve only the worst case insertion time to $O(\log n)$, preserving the other complexities. D'Amore et al. [13] improve both the insertion and deletion time to $O(\log n)$ worst case, under the assumption that the updated point has the maximum or minimum x-coordinate among the points in the set (boundary updates). Kapoor [15] shows how to support both the insertion and deletion of an arbitrary point in $O(\log n)$ worst case time. The deletion time is improved at the expense of worst case query time $O(m+chng \cdot \log n)$, where *chng* is the total number of changes that the update operations have caused to the staircase reported by the latest query operation. Kapoor modifies the structure such that a sequence of queries, and n insertions and d deletions of points requires $O(n \log n + d \log n + r)$ worst case time, where r is the total number of reported maximal points. The worst case update time remains $O(\log n)$, however the query needs $O(r)$ amortized time.

An application of the maxima contour query is the rectangular visibility query problem. A point $p \in P$ is *rectangularly visible* from a point q if the orthogonal rectangle with p and q as diagonally opposite corners contains no other point in P. The structure of Overmars and Wood [5] supports reporting the t points that are rectangularly visible from a given query point in $O(\log^2 n + t)$ worst case

[1] $\log n = \log_2 n$

Table 1. Running times for updates and the following query operations. Parameter t is the number of reported points. "All": Report all maximal points. "Emptiness": Is the query point above or below the staircase? "Dominance": Report the maximal points that dominate the query point. "Contour": Report the maximal points among the points that lie to the left of a given vertical query line. All structures occupy linear space. (†only boundary updates, ‡amortized bounds)

	Model	All	Emptiness	Dominance	Contour	Insertion	Deletion
[3]	PM	$O(m)$	$O(\log m)$	$O(\log m + t)$	$O(\log n + t)$	$O(\log^2 n)$	$O(\log^2 n)$
[8]	PM	$O(m)$	$O(\log n)$	$O(\log n + t)$	$O(\min\{\log^2 n + t, (t+1)\log n\})$	$O(\log n)$	$O(\log^2 n)$
[9]	PM	$O(m)$	$O(\log n)$	$O(\log n + t)$	$O(\log n + t)$	$O(\log n)$	$O(\log^2 n)$
[13]	PM	$O(m)$	$O(\log m)$	$O(\log m + t)$	-	$O(\log n)$	$O(\log n)^\dagger$
[15]	PM‡	$O(m)$	$O(\log n)$	$O(\log n + t)$	-	$O(\log n)$	$O(\log n)$
New	PM	$O(m)$	$O(\log n)$	$O(\log n + t)$	$O(\log n + t)$	$O(\log n)$	$O(\log n)$
New	RAM	$O(m)$	$O(\frac{\log n}{\log\log n})$	$O(\frac{\log n}{\log\log n} + t)$	$O(\frac{\log n}{\log\log n} + t)$	$O(\frac{\log n}{\log\log n})$	$O(\frac{\log n}{\log\log n})$

time, insertions and deletions of points in $O(\log^3 n)$ worst case time and uses $O(n \log n)$ space, when the structure of [3] is applied. Only the insertion time is improved to $O(\log^2 n)$ by applying the structure of [9]. Both the insertion and deletion time can be improved to $O(\log^2 n)$ worst case, using $O(n \log n)$ space, at the expense of $O(t \log n)$ worst case query time [5, Theorem 3.5].

Our Results. In the pointer machine model we present a linear space data structure that supports maxima dominance queries, and more general 3-sided range maxima queries in $O(\log n+t)$ worst case time. An arbitrary point can be inserted and deleted in $O(\log n)$ worst case time. This is the first dynamic data structure that achieves the update times of [15], and supports more general range maxima queries with worst case complexities. Our structure can be generalized to solve *4-sided range maxima* queries, namely to report the maximal points of the point set $P \cap ([x_\ell, x_r] \times [y_b, y_t])$, in $O(\log^2 n+t)$ worst case time. Updates take $O(\log^2 n)$ worst case time and the space usage is $O(n \log n)$. Using our 4-sided range maxima structure we can solve the dynamic rectangular visibility query problem with the same bounds. In the word-RAM model we present a linear space data structure that supports 3-sided range maxima queries in $O(\frac{\log n}{\log\log n} + t)$ worst case time, and updates in $O(\frac{\log n}{\log\log n})$ worst case time. This is the first dynamic data structure in the RAM model that supports these operations in sublogarithmic worst case time. Both the pointer machine and the RAM data structures support reporting all maximal points in $O(m)$ worst case time.

Outline of Solution. We follow the basic approach of previous structures. We store the points sorted by x-coordinate at the leaves of a tree, and maintain a tournament in the internal nodes of the tree with respect to their y-coordinates. An internal node u is assigned the point with maximum y-coordinate in its subtree. We observe that the nodes in the subtree of u that contain this point form a path. We also observe that the maximal points among the points assigned to the nodes hanging to the right of this path are also maximal among the points

in the subtree of u. We store these points in node u in order to allow the query algorithm to recursively access only points to be reported.

The implementation of our structures is based on the fact that we can obtain the set of points we store in a node from the set stored in its child node that belongs to the same path. In particular if that is the left child, we need to delete the points that are dominated by the point assigned to the right child and insert this point into the set. Sundar [7] shows how to implement a priority queue that supports exactly this operation (*attrition*) in $O(1)$ worst case time. This allows us to complete the insertion and deletion of a point to the construction in $O(\log n)$ worst case time. To achieve linear space we implement every path that contains the same point as a *partially persistent* priority queue with attrition. The priority queue with attrition abides by the assumptions for the technique of Brodal [12] that makes a pointer-based data structure partially persistent, since it can be implemented as a doubly linked list pointed by a constant number of additional pointers. We adapt the above construction to the word-RAM by increasing the degree of the tree to $\Theta(\log^{\varepsilon} n)$ for some $0<\varepsilon<1$. To search and update a node in $O(1)$ time, we make use of precomputed lookup-tables and the *Q-heap* of Fredman and Willard [10].

2 Preliminaries

Partial Persistence. Driscoll et al. [6] show a general technique to make pointer based data structures *partially persistent*. Suppose that an update operation on a dynamic data structure creates a new version of the data structure. A dynamic data structure is called partially persistent when only the latest version can be updated, and any previous version can be queried given a pointer to it. We obtain a list of versions called the *version list*, by ordering the versions such that an update to version i creates version $i+1$. A *rollback* discards the latest version by reversing the update that created it, and sets its preceding version in the version list as the new updatable version.

Let D be a dynamic data structure that supports queries in worst case time q and updates in worst case time u. Under the assumption that D can be modeled by a graph where the in- and out-degree of each node is bounded by a constant, Brodal [12] presented a method to make D partially persistent, such that a query to a particular version can be supported in $O(q)$ worst case time, and an update to the latest version in $O(u)$ worst case time. This improves the original partial persistence technique of Driscoll et al. [6], that only achieves amortized $O(u)$ update time, and enables the rollback operation in $O(u)$ worst case time. Moreover after performing a sequence of s atomic update operations, the partially persistent data structure occupies $O(s)$ space.

Priority Queue with Attrition. Sundar [7] introduces the *priority queue with attrition (PQA)* that supports the following operations in $O(1)$ worst case time on a set of elements drawn from a total order: DeleteMax deletes and returns the maximum element from the PQA, and InsertAndAttrite(x) inserts element x into the PQA and removes all elements smaller than x from the PQA.

The PQA uses space linear to the number of inserted elements, and is implemented as a doubly linked list with a constant number of additional pointers. Given a value x, the elements in the PQA that are larger than x can be reported in sorted order in $O(t+1)$ time, where t is the size of the output.

Red-black Tree. A *red-black tree* [1] is a balanced binary search tree that maintains a set of n elements drawn from a total order. It supports the following operations in $O(\log n)$ worst case time, using $O(n)$ space: INSERT(x) inserts element x into the tree, DELETE(x) deletes element x from the tree, and PREDECESSOR(x) returns the largest element in the tree that is smaller or equal to element x.

Q-Heap. A *Q-heap* [10] stores a subset of at most $\log^{1/4} n$ integers from the set $U=\{0,\ldots,2^w-1\}$. It operates in the RAM with word size $w \geq \log n$, and supports the operations INSERT(x), DELETE(x) and PREDECESSOR(x) in $O(1)$ worst case time. It uses $O(\log^{1/4} n)$ space, and utilizes an $O(n)$ space global precomputed lookup-table that needs $O(n)$ preprocessing time.

3 Pointer-Based Data Structure

In this section we present our pointer based data structure that supports 3-sided range maxima queries in $O(\log n + t)$ worst case time, where t is the number of reported points. The insertion and deletion of a point are supported in $O(\log n)$ worst case time. The structure occupies $O(n)$ space. Comparisons are the only allowed computation on the coordinates of the points.

3.1 Data Structure

We store the points sorted by increasing x-coordinate at the leaves of a red-black tree T. We maintain a *tournament* on the internal nodes of T with respect to the y-coordinates of the points. In particular, every internal node u contains the points $p_{x-\max}(u)$ and $p_{y-\max}(u)$ that are respectively the points with maximum x- and y-coordinate among the points in the subtree of u. The points $p_{x-\max}(u)$ allow us to search in T with respect to the x-coordinate. The points $p_{y-\max}(u)$ define n disjoint *winning paths* in T whose nodes contain the same point.

Let u be an internal node. Let u belong to a winning path π. We denote by π_u the suffix of π that starts at node u and ends at the leaf that stores $p_{y-\max}(u)$. We denote by R_u the set of right children of the nodes in π_u that do not belong to π_u themselves. We denote by $\mathrm{MAX}(u)$ the points that are maximal among $\{p_{y-\max}(v) \mid v \in R_u\}$. We can obtain $\mathrm{MAX}(u)$ from the set of points $\mathrm{MAX}(c)$, where c is the child of u that belongs to π_u. In particular, if that is the right child u_R then $R_u = R_{u_R}$ and thus $\mathrm{MAX}(u) = \mathrm{MAX}(u_R)$. Otherwise if that is the left child u_L, then $R_u = R_{u_L} \cup \{u_R\}$. Point $p_{y-\max}(u_R)$ belongs to $\mathrm{MAX}(u)$ since it has the largest x-coordinate among all points in R_u. Moreover, all the points in $\mathrm{MAX}(u_L)$ with y-coordinate at most $p_{y-\max}(u_R)_y$ should be excluded from $\mathrm{MAX}(u)$ since they are dominated by $p_{y-\max}(u_R)$. See Figure 2 for an illustration of R_u and $\mathrm{MAX}(u)$.

We implement the sets MAX(u) of the nodes u along a winning path π as the versions of a *partially persistent priority queue with attrition* (PPPQA). That is, every internal node u stores the y-coordinates of the points in MAX(u) in a priority queue with attrition (PQA). If u is the i-th node of π, then it contains the i-th version of the PPPQA. The leaf of π contains version 0. The child of u that belongs to π contains the $i-1$-th version. If that is u_R, the i-th version and the $i-1$-th versions have the same content. Else if that is u_L, the i-th version is obtained by executing InsertAndAttrite($p_{y-\max}(u_R)_y$) to the $i-1$-th version. Moreover, for every point in MAX(u) we store a pointer to the node of R_u that stores the point. This node is the highest node of its winning path. Note that R_u is not explicitly maintained anywhere in the data structure.

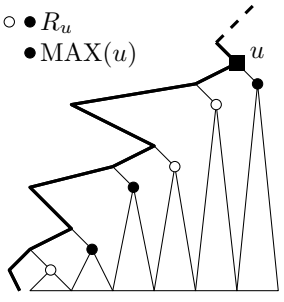


Fig. 2. The winning path π is bold. The circular nodes are in R_u. The black circular nodes are in MAX(u).

Let p be the point with maximum y-coordinate among the points $p_{y-\max}(u_L)$ and $p_{y-\max}(u_R)$. To *extend* the winning path that contains p from the child node to node u, we assign to u the points $p_{x-\max}(u_R)$ and p, and compute MAX(u).

Lemma 1. *Extending a winning path from an internal node to its parent node needs $O(1)$ time and $O(1)$ extra space.*

Proof. Points $p_{x-\max}(u)$ and $p_{y-\max}(u)$ can be computed from the children of u in $O(1)$ time. To extend the winning path that contains $p_{y-\max}(u_R)$, we create the reference to the new version of the PPPQA for the winning path in $O(1)$ time. To extend the winning path that contains $p_{y-\max}(u_L)$, we record persistently the operation InsertAndAttrite($p_{y-\max}(u_R)_y$) and create a reference to the new version in $O(1)$ worst case time using $O(1)$ extra space [12,7]. □

Lemma 1 implies that T can be constructed bottom-up in $O(n)$ worst case time, assuming that the n points are given sorted by increasing x-coordinate. The total space usage is $O(n)$.

3.2 Query

In the following we describe how to answer 3-sided range maxima queries. This immediately also gives us maxima dominance and maxima contour queries, since these are special cases where the query ranges are $[q_x, +\infty[\times[q_y, +\infty[$ and $]-\infty, x_r]\times]-\infty, +\infty[$ respectively.

Reporting the maximal points that lie above q_y. We first show how to report all maximal points with y-coordinate larger than a given value q_y among the points that belong to a subtree T_u of T rooted at an internal node u. If $p_{y-\max}(u)_y \leq q_y$ we terminate since no maximal point of T_u has to be reported. Otherwise we

report $p_{y-\max}(u)$ and compute the prefix $p[1],\ldots,p[k]$ of $\mathrm{MAX}(u)$ of points with y-coordinate larger than q_y. To do so we report the elements of the PQA of u that are larger than q_y. Let u_i be the node of R_u such that $p_{y-\max}(u_i) = p[i]$. We report recursively the maximal points in T_{u_i} with y-coordinate larger than $p[i+1]_y$ for $i \in \{1,\ldots,k-1\}$, and larger than q_y for $i = k$. The algorithm reports the maximal points in T_u in decreasing y-coordinate and terminates when the maximal point with the smallest y-coordinate larger than q_y is reported.

For the correctness of the above observe that the point $p_{y-\max}(u)$ is the leftmost maximal point among the points in T_u. The x-coordinates of the rest of the maximal points in T_u are larger than $p_{y-\max}(u)_x$. The subtrees rooted at nodes of R_u divide the x-range $]p_{y-\max}(u)_x, +\infty[$ into disjoint x-ranges. The point p' with maximum y-coordinate of each x-range is stored at a node u' of R_u. The points in $\mathrm{MAX}(u)$ are maximal among the points in T_u. Let $p[i+1]$ be the leftmost maximal point in T_u to the right of p'. If p' does not belong to $\mathrm{MAX}(u)$ then none of the points in $T_{u'}$ are maximal, since $p'_y \leq p[i+1]_y$ and $p[i+1]_x$ is larger than the x-coordinate of all points in $T_{u'}$. Otherwise p' belongs to $\mathrm{MAX}(u)$ and more precisely $p' = p[i]$. Point $p[i]$ is the leftmost maximal point among the points in $T_{u'}$. The maximal points among the points in T_u with x-coordinate at least $p[i]_x$ and y-coordinate larger than $p[i+1]_y$ belong to $T_{u'}$. In particular, they are the maximal points among the points in $T_{u'}$ with y-coordinate larger than $p[i+1]_y$.

Lemma 2. *Reporting the maximal points with y-coordinate larger than q_y among the points of a subtree of T takes $O(t+1)$ worst case time, where t is the number of reported points.*

Proof. If $p_{y-\max}(u)$ is not reported we spend in total $O(1)$ time to assess that no point will be reported. Otherwise $p_{y-\max}(u)$ is reported. We need $O(k+1)$ time to compute the k points in $\mathrm{MAX}(u)$ with y-coordinate larger than q_y. If $k = 0$ we charge the $O(1)$ time we spent in node u to the reporting of $p_{y-\max}(u)$. Otherwise we charge the time to compute $p[i]$ and to access node u_i to the reporting of $p[i]$. In total every reported point is charged $O(1)$ time. □

In order to report all maximal points of P we apply the above algorithm to the root of T with $q_y{=}-\infty$. The total time is $O(m)$.

3-sided Range Maxima Queries. We show how to report all maximal points of the point set $P \cap ([x_\ell, x_r] \times [y_b, +\infty[)$. We search for the leaves ℓ and r of T that contain the points with the largest x-coordinate smaller than x_r and smallest x-coordinate larger than x_ℓ, respectively. Let π_ℓ and π_r be the root-to-leaf paths to ℓ and r, respectively. Let R denote the set of right children of nodes of $\pi_\ell \setminus \pi_r$ that do not belong to π_ℓ themselves, and the set of left children of nodes of $\pi_r \setminus \pi_\ell$ that do not belong to π_r themselves. The subtrees rooted at the nodes u of R divide the x-range $]x_\ell, x_r[$ into disjoint x-ranges. We compute the maximal points $p[1],\ldots,p[k]$ among the points $p_{y-\max}(u)$ in R with y-coordinate larger than y_b using [2]. Let u_i be the node of R such that $p_{y-\max}(u_i) = p[i]$. We report recursively the maximal points in T_{u_i} with y-coordinate larger than $p[i+1]_y$ for $i \in \{1,\ldots,k-1\}$, and larger than y_b for $i = k$.

Lemma 3. *Reporting the maximal points for the 3-sided range maxima query takes $O(\log n + t)$ worst case time, where t is the number of reported points*

Proof. There are $O(\log n)$ nodes u in R. The points $p_{y-\max}(u)$ are accessed in decreasing x-coordinate. Therefore we can compute the maximal points $p[i], i \in \{1, \ldots, k\}$ in $O(\log n)$ time [2]. Let $p[i] = p_{y-\max}(u_i)$ for a particular node u_i of R, and let there be t_i maximal points to be reported in the subtree T_{u_i}. Since $p[i]$ will be reported we get that $t_i \geq 1$. By Lemma 2 we need $O(t_i)$ time to report the t_i points. Therefore the total worst case time to report the $t = \sum_{i=1}^{k} t_i$ maximal points is $O(\log n + t)$. □

In order to answer whether a given query point $q = (q_x, q_y)$ lies above or below the staircase (maxima emptiness query) in $O(\log n)$ time we terminate a 3-sided range maxima query for the range $[q_x, +\infty[\times[q_y, +\infty[$ as soon as the first maximal point is reported. If so, then q lies below the staircase. Else if no point is reported then q lies above the staircase.

3.3 Update

To insert (resp. delete) a point $p = (p_x, p_y)$ in the structure, we search for the leaf ℓ of T that contains the point with the largest x-coordinate smaller than p_x (resp. contains p). We traverse the nodes of the parent(ℓ)-to-root search path π top-down. For each node u of π we discard the points $p_{x-\max}(u)$ and $p_{y-\max}(u)$, and rollback the PQA of u in order to discard MAX(u). We insert a new leaf ℓ' for p immediately to the right of the leaf ℓ (resp. delete ℓ) and we rebalance T. Before performing a rotation, we discard the information stored in the nodes that participate in it. Finally we recompute the information in each node u missing $p_{x-\max}(u)$, $p_{y-\max}(u)$, and MAX(u) bottom-up following π. For every node u we extend the winning path of its child u' such that $p_{y-\max}(u) = p_{y-\max}(u')$ as in Section 3.1. Correctness follows from the fact that every node that participates in a rotation either belongs to π or is adjacent to a node of π. The winning path that ends at the rotated node will be considered when we extend the winning paths bottom-up.

Lemma 4. *Inserting or deleting a point from T takes $O(\log n)$ worst case time.*

Proof. The height of T is $O(\log n)$. Rebalancing a red-black takes $O(\log n)$ time. We need $O(1)$ time to discard the information in every node of π. By Lemma 1 we need $O(1)$ time to extend the winning path at every recomputed node. Thus we need in total $O(\log n)$ time to update the internal nodes of T. □

4 4-Sided Range Maxima Queries and Rectangular Visibility

In the following we describe how to support 4-sided range maxima queries in $O(\log^2 n + t)$ worst case time and updates in $O(\log^2 n)$ worst case time, using $O(n \log n)$ space, by borrowing ideas from [5].

For the rectangular visibility query problem, where we want to report the points that are rectangularly visible from a given query point $q = (q_x, q_y)$, we note that it suffices to report the maximal points p that lie to the lower left quadrant defined by point q (namely the points where $p_x \leq q_x$ and $p_y \leq q_y$ holds). The rectangularly visible points of the three other quadrants can be found in a symmetric way. This corresponds exactly to a 4-sided range maxima query for the range $]-\infty, q_x] \times]-\infty, q_y]$. The rectangular visibility query problem can be solved in the same bounds, using four 4-sided range maxima structures.

To support 4-sided range maxima queries, as in [5], we store all points sorted by increasing y-coordinate at the leaves of a weight-balanced B-tree S [16]. In every internal node u of S we associate a secondary structure that can answer 3-sided range maxima queries on the points that belong to the subtree S_u of S.

To perform a 4-sided range maxima query we first search for the leaves of S that contain the point with the smallest y-coordinate larger than y_b and the point with largest y-coordinate smaller than y_t, respectively. Let π_b and π_t be the root-to-leaf search paths respectively. Let L denote the set of nodes of S that are left children of the nodes in $\pi_t \setminus \pi_b$ and do not belong to π_t themselves, and the set of nodes of S that are right children of the nodes in $\pi_b \setminus \pi_t$ and do not belong to π_b themselves. The subtrees rooted at the nodes of L divide the y-range $]y_b, y_t[$ into $O(\log n)$ disjoint y-ranges. We consider the nodes $u_1, \ldots, u_k$ of L in decreasing y-coordinate. In the secondary structure of u_1 we perform a 3-sided range maxima query with the range $]x_\ell, x_r] \times]-\infty, +\infty[$. In the secondary structure of every other node u_i of L we perform a 3-sided range maxima query with the range $]b_{i-1}, x_r] \times]-\infty, +\infty[$, where b_{i-1} is the x-coordinate of the last point reported from the secondary structures of $u_1, \ldots, u_{i-1}$, i.e. the rightmost point that lies to the left of x_r and lies in y-range spanned by the subtrees $S_{u_1}, \ldots, S_{u_{i-1}}$. See Figure 3 for the decomposition of a 4-sided query.

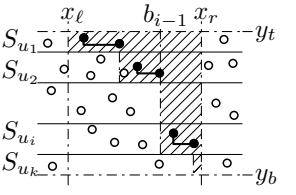


Fig. 3. 4-sided range maxima query

To insert (resp. delete) a point in S, we first search S and create a new leaf (resp. delete the leaf) for the point. Then we insert (resp. delete) the point to the $O(\log n)$ secondary structures that lie on the search path. Finally we rebalance the weight-balanced B-tree S and reconstruct the secondary structures at rebalanced nodes.

Theorem 1. *Given a set of n points in the plane, the 4-sided range maxima query problem can be solved using $O(n \log n)$ space, $O(\log^2 n + t)$ worst case query time, and $O(\log^2 n)$ worst case update time, where t is the output size.*

Proof. There are $k = O(\log n)$ nodes in L where we execute a 3-sided range maxima query. By Lemma 3 each such query takes $O(\log n + t_i)$ worst case time, where t_i is the number of reported points. In total the worst case time to report the $t = \sum_{i=1}^{k} t_i$ points is $O(\log^2 n + t)$. Each point p occurs once in every secondary structure of a node of S that is an ancestor of the leaf that contains p. There are $O(\log n)$ such ancestor nodes, thus the total space is $O(n \log n)$. By

Lemma 4 we need $O(\log n)$ time to insert and delete a point p from each secondary structure at an ancestor node. The worst case time to insert and delete p to the secondary structures that contain it is $O(\log^2 n)$. When the incremental rebalancing algorithm of [16] is applied, then for each insertion and deletion to S, $O(1)$ updates are applied to the secondary structures at each of $O(\log n)$ levels of S. By Lemma 4 each such update needs $O(\log n)$ worst case time, thus the rebalancing costs $O(\log^2 n)$ worst case time. □

5 3-sided Range Maxima in the RAM Model

In this section we consider the RAM model of computation, where the coordinates are represented by a word of $w \geq \log n$ bits and belong to the range of integers $U = \{0, 1, \ldots, 2^w - 1\}$. We present a data structure that supports 3-sided range maxima queries in $O(\frac{\log n}{\log \log n} + t)$ worst case time, where t is the number of reported points. The insertion and deletion of a point are supported in $O(\frac{\log n}{\log \log n})$ worst case time. The occupied space is linear to the number of stored points.

Structure. We store the points of P sorted by increasing x-coordinate at the leaves of an (a, b)-tree T, such that $a = \frac{1}{2}\log^{\frac{1}{4}} n$ and $b = \log^{\frac{1}{4}} n$. The height of T is $O(\frac{\log n}{\log \log n})$. Every node u is assigned the points $p_{y-\max}(u)$ and $p_{x-\max}(u)$. Define $C_y(u) = \{p_{y-\max}(u_i)_y \mid u_i \text{ is a child of } u\}$ and similarly let $C_x(u)$ be the x-coordinates of the points with maximum x-coordinate assigned to the children of u. In every internal node u we store $C_x(u)$ and $C_y(u)$ in two Q-heaps $X(u)$ and $Y(u)$, respectively. We store two global lookup-tables for the Q-heaps. We store a global look-up table S that supports 3-sided range maxima queries for a set of at most $\log^{\frac{1}{4}} n$ points in rank-space. Every node u stores a reference to the entry of S that corresponds to the permutation in $C_y(u)$. We adopt the definitions for the winning paths and MAX(u) from Section 3.1, with the difference that now R_u denotes the children of nodes v of π_u that contain the second maximal point in $c_y(v)$. As in Section 3.1, we implement the sets MAX(u) of the nodes u along a winning path as the versions of a PPPQA. For every point p in MAX(u) we store a pointer to the node v of π_u whose child is assigned p. The tree and the look-up tables use $O(n)$ space.

Query. To report the maximal points in a subtree T_u with y-coordinate larger than q_y, we first compute the prefix $p[1], \ldots, p[k]$ of the points in MAX(u) with y-coordinate larger than q_y, as in Section 3.1. For each computed point $p[i]$ we visit the node v of π_u whose child is assigned $p[i]$. We use the $Y(v)$ and the reference to S to compute the maximal points $p'[j], j \in \{1, \ldots, k'\}$ among the points in $C_y(v)$ that lie in the range $]p[i]_x, +\infty[\times]p[i+1]_y, +\infty[$. Let v_j be the child of v such that $p'[j] = p_{y-\max}(v_j)$. We recursively report the maximal points in the subtree T_{v_j} with y-coordinate larger than $p'[j+1]_y$ for $j \in \{1, \ldots, k'-1\}$, and larger than $p[i+1]_y$ for $j = k'$. If $i = k$ and $j = k'$ we recursively report the points in T_{v_j} with y-coordinate larger than q_y. Correctness derives from the fact that $p[i]_x < p'[1]_x < \cdots < p'[k']_x < p[i+1]_x$ and $p[i]_y > p'[1]_y > \cdots > p'[k']_y > p[i+1]_y$

hold, since $p[i]$ and $p'[1]$ are respectively the first and the second maximal points among the points in $C_y(v)$. The worst case query time is $O(t)$.

To answer a 3-sided range maxima query, we first use the Q-heaps for the x-coordinates in order identify the nodes on the paths π_ℓ and π_r to the leaves that contain the points with smallest x-coordinate larger than x_ℓ and with largest x-coordinate smaller than x_ℓ, respectively. This takes $O(\frac{\log n}{\log\log n})$ worst case time. For each node on π_ℓ and π_r we identify the interval of children that are contained in the x-range of the query. For each interval we identify the child c with maximum $p_{y-\max}(c)$ using S in $O(1)$ time. These elements define the set R from which we compute the maximal points using [2] in $O(\frac{\log n}{\log\log n})$ worst case time. We apply the above modified algorithm to this set, ignoring the maximal points $p'[j]$ outside the x-range of the query. The worst case time for 3-sided range maxima query is $O(\frac{\log n}{\log\log n} + t)$.

Update. To insert and delete a point from T we proceed as in Section 3.1. To extend a winning path to a node v we first find the child u of v with the maximum $p_{y-\max}(u)_y$ using $Y(v)$, i.e. the winning path of u will be extended to v. Then we set $p_{x-\max}(v) = p_{x-\max}(u_k)$ where u_k is the rightmost child of v, we set $p_{y-\max}(v) = p_{y-\max}(u)$, insert $p_{x-\max}(u)_x$ and $p_{y-\max}(u)_y$ to $X(v)$ and $Y(v)$ respectively, we recompute the reference to the table S using $Y(v)$, and we recompute $\mathrm{MAX}(v)$ from $\mathrm{MAX}(u)$ as in Section 3.1. In order to discard the information from a node u we remove $p_{x-\max}(u)$ and $p_{y-\max}(u)$ from node u and from the Q-heaps $X(v)$ and $Y(v)$ respectively, and rollback $\mathrm{MAX}(u)$. These operations take $O(1)$ worst case time. Rebalancing involves splitting a node and merging adjacent sibling nodes. To facilitate these operation in $O(1)$ worst case time we execute them incrementally in advance, by representing a node as a pair of nodes. Therefore we consider each Q-heap as two separate parts Q_ℓ and Q_r, and maintain the size of Q_ℓ to be exactly a. In particular, whenever we insert an element to Q_ℓ, we remove the rightmost element from Q_ℓ and insert it to Q_r. Whenever we remove an element to Q_ℓ, we remove the leftmost element from Q_r and insert it to Q_ℓ. In case Q_r is empty we remove the rightmost or leftmost element from the immediately left or right sibling node respectively and insert it to Q_ℓ. Otherwise if both sibling nodes have a elements in their Q-heaps, we merge them. The total worst case time for an update is $O(\frac{\log n}{\log\log n})$ since we spend $O(1)$ time per node.

Theorem 2. *Given a set of n planar points with integer coordinates in the range $U = \{0, 1, \ldots, 2^w - 1\}$, the 3-sided range maxima query problem can be solved in the RAM model with word size w using $O(n)$ space, $O(\frac{\log n}{\log\log n} + t)$ worst case query time, and $O(\frac{\log n}{\log\log n})$ worst case update time, where t is the output size.*

6 Conclusion

In the comparison model, it can be shown that $O(\log n)$ update time is optimal for the attained query time, by reduction from the Insert/Delete/FindMax problem [11]. In the cell probe model, it can be shown that $O(\frac{\log n}{\log\log n})$ time

for the maxima emptiness query of the RAM structure is optimal for the attained update time, by equivalence to the dynamic planar dominance emptiness problem [14]. Reporting maximal or rectangularly visible points of dimension larger than two, admits logarithmic factors on the output-sensitive part of the query complexity [5,15]. Improving them even for the static case remains an open problem.

References

1. Bayer, R.: Symmetric Binary *B*-Trees: Data Structure and Maintenance Algorithms. Acta Inf. 1, 290–306 (1972)
2. Kung, H.T., Luccio, F., Preparata, F.P.: On Finding the Maxima of a Set of Vectors. J. ACM 22(4), 469–476 (1975)
3. Overmars, M.H., van Leeuwen, J.: Maintenance of Configurations in the Plane. J. Comput. Syst. Sci. 23(2), 166–204 (1981)
4. Huddleston, S., Mehlhorn, K.: A New Data Structure for Representing Sorted Lists. Acta Inf. 17, 157–184 (1982)
5. Overmars, M.H., Wood, D.: On Rectangular Visibility. J. Alg. 9(3), 372–390 (1988)
6. Driscoll, J.R., Sarnak, N., Sleator, D.D., Tarjan, R.E.: Making Data Structures Persistent. J. Comput. Syst. Sci. 38(1), 86–124 (1989)
7. Sundar, R.: Worst-Case Data Structures for the Priority Queue with Attrition. Inf. Process. Lett. 31(2), 69–75 (1989)
8. Frederickson, G.N., Rodger, S.H.: A New Approach to the Dynamic Maintenance of Maximal Points in a Plane. Discrete & Comp. Geom. 5, 365–374 (1990)
9. Janardan, R.: On the Dynamic Maintenance of Maximal Points in the Plane. Inf. Process. Lett. 40(2), 59–64 (1991)
10. Fredman, M.L., Willard, D.E.: Trans-Dichotomous Algorithms for Minimum Spanning Trees and Shortest Paths. J. Comput. Syst. Sci. 48(3), 533–551 (1994)
11. Brodal, G.S., Chaudhuri, S., Radhakrishnan, J.: The Randomized Complexity of Maintaining the Minimum. Nord. J. Comput. 3(4), 337–351 (1996)
12. Brodal, G.S.: Partially Persistent Data Structures of Bounded Degree with Constant Update Time. Nord. J. Comput. 3(3), 238–255 (1996)
13. d'Amore, F., Franciosa, P.G., Giaccio, R., Talamo, M.: Maintaining Maxima under Boundary Updates. In: Bongiovanni, G., Bovet, D.P., Di Battista, G. (eds.) CIAC 1997. LNCS, vol. 1203, pp. 100–109. Springer, Heidelberg (1997)
14. Alstrup, S., Husfeldt, T., Rauhe, T.: Marked Ancestor Problems. In: Proc. 39th Foundations of Computer Science, pp. 534–544. IEEE Press, Los Alamitos (1998)
15. Kapoor, S.: Dynamic Maintenance of Maxima of 2-d Point Sets. SIAM J. Comput. 29(6), 1858–1877 (2000)
16. Arge, L., Vitter, J.S.: Optimal External Memory Interval Management. SIAM J. Comput. 32(6), 1488–1508 (2003)

Compact Navigation and Distance Oracles for Graphs with Small Treewidth

Arash Farzan[1] and Shahin Kamali[2]

[1] Max-Planck-Institut für Informatik, Saarbrücken, Germany
[2] Cheriton School of Computer Science, University of Waterloo, Ontario, Canada

Abstract. Given an unlabeled, unweighted, and undirected graph with n vertices and small (but not necessarily constant) treewidth k, we consider the problem of preprocessing the graph to build space-efficient encodings (oracles) to perform various queries efficiently. We assume the word RAM model where the size of a word is $\Omega(\log n)$ bits.

The first oracle, we present, is the navigation oracle which facilitates primitive navigation operations of adjacency, neighborhood, and degree queries. By way of an enumerate argument, which is of independent interest, we show the space requirement of the oracle is optimal to within lower order terms for all treewidths. The oracle supports the mentioned queries all in constant worst-case time. The second oracle, we present, is an exact distance oracle which facilitates distance queries between any pair of vertices (i.e., an all-pair shortest-path oracle). The space requirement of the oracle is also optimal to within lower order terms. Moreover, the distance queries perform in $O(k^2 \log^3 k)$ time. Particularly, for the class of graphs of our interest, graphs of bounded treewidth (where k is constant), the distances are reported in constant worst-case time.

1 Introduction

Graphs are arguably one of the most prolific structures to model relationships among entities. With the ever-growing size of objects to model, the corresponding graphs increase in size. As a result compact representation of graphs has always been of interest. In this paper, we consider the problem of representing graphs compactly while allowing efficient access and utilization of the graph by showing fast support of navigation and distance queries.

Random graphs are highly incompressible [2]. Fortunately, graphs that arise in practice are not random and turn out to have some combinatorial structural property. Therefore, researchers have considered graphs with various combinatorial structures for the purpose of space-efficient representation (see [9] for a review of the exiting results). In this paper, we are interested in compact representation of graphs with a small treewidth (to be defined in Section 2). Graphs of bounded treewidth are of interest since many NP-hard problems on general graphs are solvable in polynomial time on these graphs. In addition, graphs with small treewidth occur in many more real-world applications [5,6]. We assume the standard word RAM model where a word is at least $\lg n$ bits wide and n is the number of vertices (lg denotes $\log_2$).

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 268–280, 2011.

1.1 Contribution

In the first part of the paper (Section 4), we describe a data structure that encodes a given undirected and unlabeled graph with n vertices and treewidth at most k in $k(n + o(n) - k/2) + O(n)$ bits and supports degree, adjacency, and neighborhood queries in constant time. Degree query is to report the degree of a vertex. Adjacency query is given two vertices u, w to determine if edge (u, v) exists. Neighborhood query is to report all neighbors of a given vertex in constant time per neighbor. These three queries constitute the set of primitive navigational queries often required in a graph [2,10,3].

In [12] an implicit representation of graphs with treewidth k in $n(\lg n + O(k \lg \lg(n/k)))$ bits is presented, which is not compact for all values of k. Although it is not explicitly mentioned, the succinct representation of separable graphs given in [3] yields an optimal navigation oracle for graphs of bounded treewidth for any treewidth $k = O(1)$. This is since graphs with a constant treewidth are also separable (a graph is separable if it and its subgraphs can be partitioned into two approximately equally sized parts by removing a relatively small number of vertices [2]). The storage requirement of the oracle for separable graphs is optimal to within lower order terms, and previously mentioned navigation queries perform in constant time. In this paper, we extend the result to graphs with treewidth k, where $k = \Omega(1)$.

Moreover, we show that the storage requirement of the oracle is optimal for all values of k by proving that $k(n - o(n) - k/2) + \delta n$ bits are required to encode graphs of treewidth k and n vertices (δ is a positive constant). Our proof is a counting argument which is of independent interest as to best of our knowledge, there existed no such enumerative result for graphs with a given treewidth (though a lower bound of $kn - o(kn)$ is known [11] for graphs with pagenumber k, which are a larger family of graphs which include graphs with treewidth smaller than k [8]). The desired oracle of this paper adopts the encoding of [3] for values $k = O(1)$ and the encoding outlined in this paper for $k = \Omega(1)$. Since the storage requirement of the oracle matches the entropy bound for constant values of k and our lower bound for non-constant values of k, both the space of the oracle and our lower bound are tight.

In the second part of the paper, we give distance oracles for undirected, unlabeled, and unweighted graphs with n vertices and treewidth k that for all values of k requires the entropy bound number of bits to within lower order terms. These are *exact* oracles that report the distance of two given vertices precisely. The distance queries perform in $O(k^2 \lg^3 k)$ in which k is the graph treewidth. We emphasize that for graphs of bounded treewidth where k is constant (the family of graphs of our interest), the queries are supported in constant time. Exact distance oracles for unweighted undirected graph require $\Omega(n^2)$ bits and there exists an oracle with about $0.79n^2$ bits [16]. Hence, we show that for graphs with a bounded treewidth, these results can be significantly improved as there is a linear size exact distance oracle (the number of edges can be $\theta(kn)$).

The time to construct the oracles depends on the time to compute the treewidth of the given graph and compute the tree decomposition correspondingly.

Determining the treewidth of a graph is NP-hard [5]. Fortunately however, for graphs with constant treewidth, the treewidth and the corresponding tree decomposition can be determined in linear time [6]. Moreover, for graphs with treewidth $k = \omega(1)$, there exists a polynomial time algorithm that approximates the treewidth within $O(\log k)$ factor and generates the corresponding tree decomposition [5]. All other aspects of navigation oracles can be constructed in $O(kn)$ time where k is the determined treewidth and n is the number of vertices. For the distance oracle, we pre-compute distances between every pairs of vertices at the initial stage, and this can be accomplished in $o(kn^2)$ time [7].

2 Tree Decompositions and Variations

We use the notion of tree decompositions of graphs to design the oracles:

Definition 1 ([5]). *A tree decomposition of a graph $G = (V, E)$ of width k is a pair $(\{X_i \| i \in I\}, T)$ where $\{X_i \| i \in I\}$ is a family of subsets of V (bags), and T is a rooted tree whose nodes are the subsets X_i such that*

- $\bigcup_{i \in I} X_i = V$ *and* $\max_{i \in I} |X_i| = k+1$.
- *for all edges $(v, w) \in E$, there exists an $i \in I$ with $v \in X_i$ and $w \in X_i$.*
- *for all $i, j, k \in I$: if X_j is on the path from X_i to X_k in T, then $X_i \cap X_k \subseteq X_j$.*

We say a vertex $v \in V$ is introduced *in node $X_i (i \in I)$ of the tree, if v is in the bag X_i ($v \in X_i$) but not in that of the parent of X_i. All vertices at the bag of the root node are introduced by definition.*

For each of the oracles, we use a specially adapted version of tree decomposition. For the navigation oracle, we use a *standard* tree decomposition, in which each bag contains exactly $k + 1$ vertices, and two neighboring nodes share exactly k vertex, i.e., each node introduces one vertex. It is known that a tree decomposition can be changed to form a standard tree decomposition in linear time [4]. To design distance oracles, we use the *height-restricted* tree decomposition T, whose height is logarithmic in the number of vertices, i.e., $height(T) = O(\log n)$. A tree decomposition can be transformed into a height-restricted tree decomposition by the following lemma:

Lemma 1. *Given a tree decomposition with treewidth k for a graph with n vertices, one can obtain, in linear time, a height-restricted tree decomposition with n nodes and width at most $3k + 2$ (proof in the long version of the paper).*

3 Lower Bound

To best of our knowledge, there exists no enumerative result for the number of unlabeled graphs with a given treewidth k. We prove one in this section. We use the concept of *asymmetric trees* (also known as *identity* trees), which are trees in which the only automorphism is the identity, i.e., each vertex can be uniquely

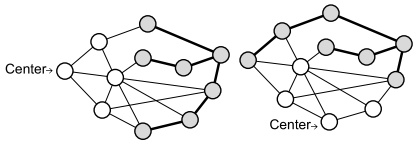


Fig. 1. Two different understanding of tree vertices in an asymmetric 4-graph. Dark vertices are tree vertices.

distinguished from others. Harary et al, showed the total number of asymmetric trees on n nodes is $u(n) \sim cn^{-5/2}\mu^{-n}$ in which c and μ are positive constants roughly equal to 0.299 and 0.397, respectively [14]. We use this result to count *asymmetric k-graphs* as a family of graphs with treewidth k.

Definition 2. *An asymmetric k-graph on n vertices is a graph which has an asymmetric tree of size $n-k$ as an induced subgraph. The involved vertices in this subgraph are called* tree vertices*. Among the other k vertices, there is one vertex, called* center*, which is connected to all other $k-1$ non-tree vertices and is not connected to any of the $n-k$ tree vertices (Figure 1).*

Lemma 2. *Any asymmetric k-graph has treewidth at most k.*

Proof. Consider a tree decomposition of the asymmetric tree, which has at most two vertices in each bag. Copy all other vertices except the center to all bags. Now each bag contains $2 + (k-1)$ vertices. Create a new bag of size k involving all non-tree vertices (including center) and attach it to an arbitrary position in the tree decomposition. The result is a legitimate tree decomposition of width k (at most $k+1$ vertices in each bag).

Therefore, to get a lower bound on the number of graphs with of treewidth k, we just count asymmetric k-graphs.

Theorem 1. *The number of asymmetric k-graphs with n vertices is at least $x_n \sim c2^{(k+\delta)n-(k^2+(3+2\delta)k-2)/2} \times (n-k)^{-5/2}/(n(k-1)!)$ where c and δ are constants roughly equal to 0.299 and 0.332 (proof in the long version of the paper).*

Since we match the bound with an encoding, the exponent is tight within lower order terms.

Corollary 1. *At least $k(n - o(n) - k/2) + \delta n$ bits are required to represent a graph of treewidth k with n vertices, where δ is a constant roughly equal to* 0.332*.*

4 Navigation Oracles

In this section, we provide a compact representation of graphs of treewidth k which supports adjacency, neighborhood, and degree queries in constant time in the $\lg n$-bit word RAM model. The representation requires $k(n + o(n) - k/2) +$

$O(n)$, which is tight given corollary 1. First we mention some existing results for auxiliary data structures we use in our representation.

Succinct rank/select structures: For a binary sequence S, we define $access_S(i)$ as the content of the i'th index of S, $rank_S(i,c)$ as the number of occurrences of c before index i, and $select_S(i,c)$ as the index of the i'th occurrence of c in S ($c \in \{0,1\}$). There are data structures which represent a binary sequence of length n using $n+o(n)$ bits which supports *access*, *rank*, *select* in constant time. Moreover, for sequences with m ones ($m \ll n$), the space can be reduced to $\lg \binom{n}{m} + O(n \lg\lg n / \lg n)$ to support queries in constant time [17].

Balanced Parenthesis and Multiple Parenthesis: A balanced parenthesis sequence of size $2n$, which is equivalent to an ordered tree of size n, can be represented in $2n+o(n)$ bits, with support of $access(v)$, $rank(v,'(')$, $select(v,'(')$, $findmatch(v)$, and $child(i,v)$ in constant time [15]; *access*, *rank*, and *select* are defined as before, $findmatch(v)$ finds the position of parenthesis matching the parenthesis at position v, and $child(v,q)$ finds the position of the q'th child of node v. A multiple parenthesis sequence, is an extension of balanced parenthesis to k types of parenthesis, where an open parenthesis of *type* i, denoted by $(_i$, can be matched by a closed parenthesis of the same type, denoted by $)_i$.

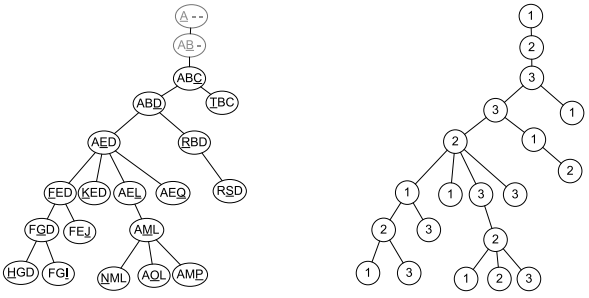
Lemma 3. *[1] A multiple parenthesis sequence with $2n$ parentheses of k types, in which the parentheses of any given type are balanced, can be represented using $(2+\epsilon)n \lg k + o(n \lg k)$ bits to support m_access, m_rank, m_select, m_findmatch and m_enclose in $O(1)$ time; all operations are defined as before, m_enclose(v,i) gives the position of the tightest open parenthesis of type i which encloses v.*

Compact Tables: Given a binary matrix M of size $k \times n$, we are interested in a compact representation of M which supports the following queries: $access(i,j)$ which gives the content of $M[i,j]$, $r\_successor(i,j)$ which gives the entry one in row i that comes after index j, and $c\_successor(i,j)$ which is defined identically on columns. For our purpose, we need to represent matrices in which $k \leq n$, and the first k columns form a triangular submatrix.

Lemma 4. *A $k \times n$ matrix, in which the first k columns form a triangular submatrix ($k \leq n$) can be presented using $kn - k^2/2 + o(kn)$ bits to support access and successor queries in constant time (proof in the long version of the paper).*

4.1 Representing the Tree Decomposition

Assume for a given graph $G = (V, E)$, a tree decomposition τ of width k is given in standard form. We assign *types* to all vertices in a top-down manner: for the vertices in the root, fix an arbitrary ordering $1 \,..\, k+1$ and give a vertex type i iff it has index i in this ordering. For a vertex v introduced in a bag X, define $type(x) = j$, where j is the type of the vertex in the parent of X which has been replaced by v. Note that the only information associated with each bag is the type of the vertex it introduces. So we can present the tree decomposition τ as an ordered tree with a single label, not larger than $k+1$, on each bag. We assume



$(_1\,(_2\,(_3\,(_3\,(_2\,(_1\,(_2\,(_1\,)_1\,(_3\,)_3\,)_2\,(_3\,)_3\,)_1\,(_1\,)_1\,(_3\,(_2\,(_1\,)_1\,(_2\,)_2\,(_3\,)_3\,)_2\,)_3\,(_3\,)_3\,)_2\,(_1\,(_2\,)_2\,)_1\,(_1\,)_1\,)_3\,)_2\,)_1$

Fig. 2. A standard tree decomposition, an ordered labeled tree, and a multiple parenthesis are all equivalent

the root introduces $k+1$ vertices of different types, and to make representation easier separate them in a path of bags, with labels 1 to $k+1$ (See Figure 2).

To represent τ efficiently, we use multiple parenthesis structure of Lemma 3. Assume a preorder traversal of τ; we open a parenthesis of type i ($i \leq k$) whenever we enter a bag with label i, and close it when we leave the bag. The result would be a balanced sequence of $2n$ parenthesis of $k+1$ types, and using Lemma 3, this can be presented using $(2+\epsilon)n \lg k + o(n \lg k)$ bits. We call this sequence the *MP sequence* and represent every vertex in the graph by the index of its opening parenthesis in this sequence.

To represent a graph of treewidth k, beside the tree decomposition, we need to store which edges are indeed present in each bag. In fact, the tree decomposition represents a (full) k-tree, which is the graph with maximal edges to respect the tree decomposition. In a graph of treewidth k when a new vertex is introduced in a bag X, it can be connected to any subset of k vertices present in X. These vertices all have distinct types as they all appear in bag X. For each vertex v, let l_v be a bitmap of size $k+1$, such that $l_v(j)$ denotes if there is an edge between v and u_j, where u_j is the unique vertex of type j present in bag X. Observe that u_j is introduced in the closest ancestor of v which has type j. Let all l_vs form the columns of a table M, referred as 'big table', where the vertices are arranged in preorder. So M is a matrix of size $(k+1) \times n$ and $M[j, v] = l_v(j)$ (see Figure 3). Since two vertices of the same type cannot be connected, for any vertex v we have $M[type(v), v] = 0$ (in the figure these entries are distinct by '*'). Also the first k columns of M are associated with the vertices introduced in the root and form a triangular submatrix. We apply Lemma 4 to store M in $kn - k^2/2 + o(kn)$ bits to support *access* and *successor* queries in constant time.

Assume we are given a vertex v (its index in the MP sequence), and we need to access its column in the big table, i.e., the index of v in the preorder walk. We use a *map structure* as follows: create a binary sequence S of size $2n$ with one at position i if the i'th element is an open parenthesis (of any type)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	*	1	1	0	1	*	0	*	1	1	1	0	0	*	0	1	1	*	1	*
2	-	*	0	1	*	0	*	1	0	1	0	1	*	0	*	1	1	0	*	1
3	-	-	*	*	1	0	1	1	*	*	1	*	0	1	1	*	*	1	1	1

Fig. 3. A big matrix associated with tree decomposition of Figure 2

and zero otherwise. We store this sequence using $2n + o(n)$ bits to support *rank* and *select* in constant time. Now $rank_S(v, 1)$ gives the index of v in the preorder walk, and $select_S(i, 1)$ retrieves the i'th vertex in the preorder walk. This enables us to interchangeably represent a vertex by its position in the preorder (the big matrix index) or its position in the MP sequence. Moreover, assume we are given a range R in the MP sequence which may start or end with a close parenthesis, and we need the preorder range which include the involved open parentheses. If both endpoints of R are open parentheses, we simply map them into preorder indices as discussed. If R starts (ends) with a closed parenthesis, we need to find the next (previous) open parenthesis of any type. We can use S to find the index of next (previous) open parenthesis as $select_S(rank_S(i, 1) \pm 1)$.

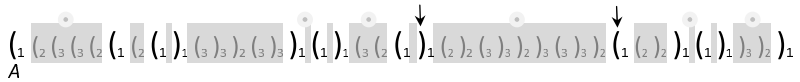
The MP sequence and the big table are sufficient for representing a graph of treewidth k. The other data structures used in the rest of this section are indices to support queries in constant time. Due to lack of space, we describe support for neighborhood queries here and support for degree and adjacency queries are presented in the long version of the paper.

4.2 Neighbor Report

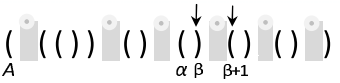
We are given a vertex v, and asked to report its neighbors in constant time per neighbor. We say a vertex u is a *potential neighbors* of v if there is a bag that contains both vertices. The column representing v in the big table distinct the actual neighbors of v among those potential neighbors which precede v in preorder walk. To report these neighbors, we successively apply *c_successor* on the big table M to visit all ones in the column of v. Assume we have $M[j, v] = 1$, then we report the parenthesis of type j which encloses v using $m\_enclose(v, j)$. Note that these operations take constant time per neighbor.

Next, we show how report neighbors which come after v in the preorder walk. Using the MP sequence, we can find the potential neighbors of v: we scan sequence from the position after v and report every vertex (open parenthesis) until we observe the first open parenthesis of the same type as v. Let w be such parenthesis, we jump to the matching parenthesis of w using $m\_findmatch_{MP}(w)$ in constant time, and continue this until we see the close parenthesis matching v. Therefore, in the tree decomposition, we skip the subtrees in which v has been overwritten by w.

The potential neighbors of each vertex form *segments* of consequent vertices in the preorder walk. A segment of type i is a range of elements in the MP sequence bordered by two parenthesis of type i. The bordering parenthesis can be open or closed, and their segment can be empty if they are adjacent in the MP sequence



(a) The MP sequence in which the segments of type 1 are highlighted. The circles show those associated to vertex A



(b) The contracted parenthesis of type 1 (C_1)

Fig. 4. The ignore sequence for vertex A (Ig_A) is 100100. To find the 4th segment associated to A in the MP sequence, we find the 3rd child of A in C_i (the starred vertex in (b)), its matching parenthesis and the one after (arrowed ones) are the boundaries in the contracted sequence, which can be mapped to the MP sequence.

(Figure 4(a)). Note that any segment is associated to exactly one vertex, which is a vertex of the same type which encloses it. To report actual neighbors in a given segment associated to vertex v, we successively apply $r\_successor_M$ on the row of the same type of v in the big table to find all ones in the range of the segment. So, we can report the neighbors inside a segment in constant time per neighbor. We also need to address how to select the appropriate segments. We say the segment is *good* if it includes at least one actual neighbor of v, and it is *bad* otherwise. Note that there may be a non-constant number of bad segments associated to a vertex, and we cannot probe all of them. For each vertex v, we define a bitmap Ig_v where $Ig_v(i)$ determines whether the ith segment associated to v is good ('1') or bad ('0'). We store an *ignore sequence* IG as follows: read vertices in preorder, for each vertex v write down a '2' followed by the sequence Ig_v. The result would be a sequence of size $3n-k$ on alphabet $\{0, 1, 2\}$, which can be stored using $O(n)$ bits to support *select* in constant time [13]. To see why the size of IG is $3n-k-1$, note that there $2n_i - 1$ segments of type i where n_i is the number of vertices with type i, so there are totally $2n-(k+1)$ segments. Since each segment is associated to exactly one vertex, the size of IG is $2n-(k+1)+n$. Note that Ig_v is the subsequent between $select_{IG}(i, 2)+1$ and $select_{IG}(i+1, 2)$, in which i is the index of v in preorder walk.

Using the ignore sequence, we can distinguish the index of good segments among all segments associated to a vertex v. Next, we need to locate these segments in the MP Sequence and use the map structure to locate the range of the segment in the big table. For each type i, we store a *contracted parenthesis* of type i, denoted by C_i, as a copy of the MP sequence in which all parenthesis except those of type i are deleted. The result would be a balanced parenthesis sequence, equivalently an ordered tree, for each type. The total size of these trees is equal to n and we need $2n + o(n)$ bits to represent them. Assume we need to locate the t'th segment of vertex v in the MP sequence, and let i be the type of v. If $t=1$, the desired segment starts with the parenthesis representing v and ends with the next parenthesis of the same type, which can be found in constant time.

If $t > 1$, we locate the segment in the contracted parenthesis sequence and then map it into the MP sequence. First we locate v in the contracted parenthesis, using $v_c = select_{C_i}(x, '(')$ where x is the rank of v among vertices of the same type, i.e., $x = rank_{MP}(v, '('_i)$. Observe the t'th segment of v starts after the close parenthesis matching the open parenthesis representing t'th child of v in the contracted parenthesis (Figure 4(b)). So we apply $\alpha = child_{C_i}(v_c, t)$ and $\beta = findmatch_{C_i}(\alpha)$ to find $\beta, \beta+1$ as the two neighboring parenthesis of type i which bound segment t in the contracted parenthesis. Using *rank* and *select*, respectively on C_i and MP we can locate these parenthesis in the MP sequence.

To summarize, to report neighbors of vertex v which succeed v in the preorder walk, we use the ignore sequence to find the indices of good segments among all segments associated to v. We use contracted parenthesis to find the actual position of the good segments in the MP sequence, and use map structure to find the range of the segments in the big table. Using *r_successor* operation in the big table we can report neighbors in constant time per neighbor.

The additional space used for supporting neighbor report are ignore sequence and contracted parenthesis, which are both stored in $O(n)$ bits. The index used for degree request needs $n \lg k + o(nk)$ bits, and there is no additional index for adjacency queries (details are presented in the long version of the paper). Together with the main structures (the MP sequence and the big table), the size of the oracle would be $k(n + o(n) - k/2) + O(n)$.

Theorem 2. *Given a graph of size n and treewidth k, an oracle is constructed to answer degree, adjacency, and neighborhood queries in constant time. The storage requirement of the oracle is optimal to within lower order terms.*

5 Distance Oracles

To give a distance oracle, we obtain the height-restricted tree decomposition T of the input graph G of treewidth k using Lemma 1. Let k' denote the maximum number of vertices in each node of T. Since treewidth of T is at most $3k + 2$, we have $k' \leq 3k + 3$. We define the weight of a node as the number of vertices introduced in that node. Correspondingly, we define the weight of a subtree as the sum of the weights of nodes in the subtree. There are two recursive decompositions of G into smaller subgraphs using its height-restricted tree decomposition T using the following lemma (proof in the long version of the paper):

Lemma 5. *For any parameter $1 \leq L \leq n$, a tree with n nodes such that the weight of nodes is at most k' can be decomposed into $\Theta(n/L)$ subtrees of weight at most $2L + k'$ which are pairwise disjoint form their roots. Furthermore, aside from edges stemming from the component root nodes, there is at most one edge leaving a node of a component to its child in another component.*

In the first phase, tree T is decomposed into smaller subtrees $T_1, T_2, \ldots$ using Lemma 5 with value $L = k' \lg^3(n)$ (the first phase is skipped if $L \geq n$). Let V_i be the set of graph vertices that occur in a node in subtree T_i. We define G_i as the subgraph of G induced on V_i.

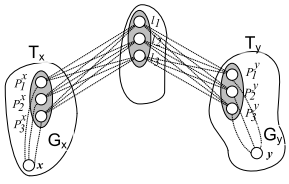


Fig. 5. Distance oracle: computing the distance between x and y

Lemma 5 guarantees that there are at most two nodes of each subtree T_i that are connected via a tree edge to other subtrees; we refer to these tree nodes as *portal* nodes. These nodes collectively contain $2k'$ graph vertices. We refer to these vertices as the *portal* vertices of G_i, and denote this set of vertices by P_i.

To reduce the distance oracle to within G_i's, we explicitly store the distance from each portal vertices to all vertices in an ancestor node of the corresponding portal nodes. Namely, for each vertex $v \in P_i$ in a portal tree node t and vertex u in a node an ancestor of t, we explicitly store the distance between v and u. Since the height of the tree is $O(\log n)$, there are $O(k' \log n)$ such vertices as u. The storage requirement of this list in number of bits is $O\left(\frac{n}{k' \lg^3 n} k' (k' \log(n)) \log(n)\right) = o(kn)$.

We also take the projection of tree T on portal nodes by adding an edge between two portal nodes if and only if the path in T between them does not contain another portal node. The projected tree is a tree on $O\left(n/(k' \log^3(n))\right)$ nodes. We preprocess and store the tree (in $O\left(n/(k' \log^3(n))\right)$ bits) to be able to answer lowest common ancestor queries in constant time [15].

If we can internally in any G_i determine the distance between any two vertices $s, t \in G_i$, then using the explicitly stored distances for portal vertices, we can determine the distances globally between any two vertices in G. Given two vertices x, y, we determine the subgraphs G_x, G_y they belong in. We use a *rank/select* structure to accomplish this task in constant time and $o(n)$ storage. We compute the distances from x to the portal vertices $p_1^x, \ldots, p_{2k'}^x$ in G_x and analogously the distances from y to the portal vertices $p_1^y, \ldots, p_{2k'}^y$ of G_y (see Figure 5). Let T_x and T_y be the subtrees corresponding to G_x, G_y. We determine in constant time the lowest common ancestor L of the roots of T_x and T_y. Portal vertices have their distances to vertices introduced in their ancestors explicitly stored. Therefore, $p_1^x, \ldots, p_{2k'}^x$ and also $p_1^y, \ldots, p_{2k'}^y$ have their distances to vertices $l_1, \ldots, l_{k'}$ in the bag of node L stored. Without loss of generality, we assume the harder case where roots of T_x, T_y are not an ancestor of each other, the details of the other case is deferred to the full version of this paper.

We repeat the previous step for each G_i by applying Lemma 1 to obtain a height-restricted tree decomposition and using Lemma 5 with value $L = k' \lg^2(k')(\lg \lg(n))^3$ to obtain smaller subgraphs G_i'. Additionally, we store the distance between each second-level portal vertex and all first-level portal vertices contained in the same subgraph G_i. This structure allows us to reduce the problem to within second-level subgraphs G_i', without paying a factor of k' for the

query time. The space requirement of this structure can be analyzed similarly to $o(kn)$. We repeat the step for a final time using value $L = k' \lg^2(k')(\lg\lg\lg(n))^3$ to obtain tiny subgraphs G_i''. Hence, the problem reduces to computing the distances of a vertex to third-level portal vertices confined to an individual third-level graph G_i''. As G_i'''s are subgraphs of the original graph, their treewidth is at most k. The corresponding tree decomposition for these graphs can be obtained trivially by projecting from the tree decomposition of the original graph. Hence, treewidths of G_i'''s are k and not k' any further.

We distinguish two cases according to the value of k. For smaller values of k where $\lg k \leq (\lg\lg\lg n)^3$, the size of third-level subgraphs G_i'' is very small, therefore, we use a look-up table to catalog all graphs with p vertices and treewidth $k-1$ such that $p < \lg(n)/(2k)$. We exhaustively list answers to all distance queries together with each graph. The representation of Section 4 bounds the number of such graphs and consequently the size of the table is $o(n)$. A third-level graph $G_i'' = (V_i'', E_i'')$ is represented by an index to within the look-up table and therefore, the space requirement of each G_i'' matches the entropy of graphs with $|V_i''|$ vertices and treewidth G_i'' (we note that every subgraph of a graph with treewidth k has treewidth at most k). Since $\sum_i |V_i''| = n + o(n)$, the distance oracle requires space which matches the entropy of graphs with treewidth k to within lower order terms. Distances in G_i'' are read in constant time from the table and there is an additive overhead of $O(k^2)$ for each level of recursion. Thus, the total distance query time is $O\left(k^2\right)$. Therefore, the distance query performs in constant time when k is constant.

For larger values of k, where $\lg k > (\lg\lg\lg n)^3$, we simply store third-level graphs $G_i'' = (V_i'', E_i'')$ using the navigation oracle representation of Section 4 to store each G_i'' in $k(|V_i''| + o(|V_i''|) - k/2) + O(|V_i''|)$ bits. Since $\sum_i |V_i''| = n + o(n)$, the total storage requirement for distance oracle in this case is $k(n + o(n) - k/2) + O(n)$. In order to determine the distance of a vertex in G_i'' to the third-level portals of G_i'', we simply perform a breadth first search (BFS). The time of performing a BFS is $O\left(k^2 \lg^2(k)(\lg\lg\lg(n))^3\right)$ which in this case is $O\left(k^2 \log^3(k)\right)$. This dominates the overhead of $O\left(k^2\right)$ from recursion, and hence distance queries perform in $O\left(k^2 \log^3(k)\right)$ time.

Theorem 3. *Given an unlabeled, undirected, and unweighted graph with n vertices and of treewidth k, an exact distance oracle is constructed to answer distance queries in time $O\left(k^2 \log^3 k\right)$. The storage requirement of the oracle is optimal to within lower order terms.*

6 Conclusion

We considered the problem of preprocessing a graph small treewidth to construct space-efficient oracles that answers a variety of queries efficiently. We gave a navigation oracle that answers navigation queries of adjacency, neighborhood, and degree queries in constant time. We also proposed a distance query which reports the distances of any pair of vertices in $O\left(k^2 \log^3 k\right)$ where k is the (determined)

treewidth. By way of an enumerative result, we showed the space requirements of the oracles are optimal to within lower order terms.

Acknowledgement

We are thankful to Magnus Wahlstrom for helpful discussions.

References

1. Barbay, J., Aleardi, L.C., He, M., Ian Munro, J.: Succinct representation of labeled graphs. In: Tokuyama, T. (ed.) ISAAC 2007. LNCS, vol. 4835, pp. 316–328. Springer, Heidelberg (2007)
2. Blandford, D.K., Blelloch, G.E., Kash, I.A.: Compact representations of separable graphs. In: Proceedings of 14th ACM-SIAM Symposium on Discrete Algorithms, SODA 2003, pp. 679–688 (2003)
3. Blelloch, G.E., Farzan, A.: Succinct representations of separable graphs. In: Amir, A., Parida, L. (eds.) CPM 2010. LNCS, vol. 6129, pp. 138–150. Springer, Heidelberg (2010)
4. Bodlaender, H.L.: NC-algorithms for graphs with small treewidth. In: van Leeuwen, J. (ed.) WG 1988. LNCS, vol. 344, pp. 1–10. Springer, Heidelberg (1989)
5. Bodlaender, H.L.: A tourist guide through treewidth. Acta Cybernetica 11, 1–23 (1993)
6. Bodlaender, H.L.: Treewidth: Characterizations, applications, and computations. In: Fomin, F.V. (ed.) WG 2006. LNCS, vol. 4271, pp. 1–14. Springer, Heidelberg (2006)
7. Chan, T.M.: All-pairs shortest paths for unweighted undirected graphs in o(mn) time. In: Proc. 17th ACM-SIAM Symposium on Discrete Algorithm, SODA 2006, pp. 514–523 (2006)
8. Dujmovic, V., Wood, D.R.: Graph treewidth and geometric thickness parameters. Discrete Comput. Geom. 37, 641–670 (2007)
9. Farzan, A.: Succinct Representation of Trees and Graphs. PhD thesis, School of Computer Science, University of Waterloo (2009)
10. Farzan, A., Ian Munro, J.: Succinct representations of arbitrary graphs. In: Halperin, D., Mehlhorn, K. (eds.) Esa 2008. LNCS, vol. 5193, pp. 393–404. Springer, Heidelberg (2008)
11. Gavoille, C., Hanusse, N.: On Compact Encoding of Pagenumber k Graphs. Discrete Mathematics & Theoretical Computer Science 10(3), 23–34 (2008)
12. Gavoille, C., Labourel, A.: Shorter implicit representation for planar graphs and bounded treewidth graphs. In: Arge, L., Hoffmann, M., Welzl, E. (eds.) ESA 2007. LNCS, vol. 4698, pp. 582–593. Springer, Heidelberg (2007)
13. Grossi, R., Gupta, A., Vitter, J.S.: High-order entropy-compressed text indexes. In: Proc. 14th ACM-SIAM Symposium on Discrete Algorithms, SODA 2003, pp. 841–850 (2003)
14. Harary, F., Robinson, R.W., Schwenk, A.J.: Twenty-step algorithm for determining the asymptotic number of trees of various species: Corrigenda. Journal of the Australian Mathematical Society 41(A), 325 (1986)

15. He, M., Ian Munro, J., Srinivasa Rao, S.: Succinct Ordinal Trees Based on Tree Covering. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 509–520. Springer, Heidelberg (2007)
16. Nitto, I., Venturini, R.: On compact representations of all-pairs-shortest-path-distance matrices. In: Ferragina, P., Landau, G.M. (eds.) CPM 2008. LNCS, vol. 5029, pp. 166–177. Springer, Heidelberg (2008)
17. Raman, R., Raman, V., Satti, S.R.: Succinct indexable dictionaries with applications to encoding k-ary trees, prefix sums and multisets. ACM Trans. Algorithms 3(4), 43 (2007)

Player-Centric Byzantine Agreement

Martin Hirt[1] and Vassilis Zikas[2,⋆]

[1] Department of Computer Science, ETH Zurich
hirt@inf.ethz.ch
[2] University of Maryland, USA
vzikas@cs.umd.edu

Abstract. Most of the existing feasibility results on Byzantine Agreement (BA) are of an *all-or-nothing* fashion: in Broadcast they address the question whether or not there exists a protocol which allows *any* player to broadcast his input. Similarly, in Consensus the question is whether or not consensus can be reached which respects pre-agreement on the inputs of *all* correct players. In this work, we introduce the natural notion of *player-centric BA* which is a class of BA primitives, denoted as PCBA $= \{\text{PCBA}(\mathcal{C})\}_{\mathcal{C}\subseteq\mathcal{P}}$, parametrized by subsets $\mathcal{C}$ of the player set. For each primitive PCBA$(\mathcal{C}) \in$ PCBA the validity is defined on the input(s) of the players in $\mathcal{C}$. Broadcast (with sender p) and Consensus are special (extreme) cases of PCBA primitives for $\mathcal{C} = \{p\}$ and $\mathcal{C} = \mathcal{P}$, respectively.

We study feasibility of PCBA in the presence of a general (aka non-threshold) mixed (active/passive) adversary, and give a complete characterization for perfect, statistical, and computational security. Our results expose an asymmetry of Broadcast which has, so far, been neglected in the literature: there exist non-trivial adversaries which can be tolerated for Broadcast with sender some $p_i \in \mathcal{P}$ but *not* for *some other* $p_j \in \mathcal{P}$ being the sender. Finally, we extend the definition of PCBA by adding fail corruption to the adversary's capabilities, and give exact feasibility bounds for computationally secure PCBA$(\mathcal{P})$ (aka Consensus) in this setting. This answers an open problem from ASIACRYPT 2008 concerning feasibility of computationally secure multi-party computation in this model.

1 Introduction

Byzantine agreement (BA) is one of the most studied problem in the areas of distributed protocols and multi-party computation. The problem was introduced by Lamport, Shostak, and Pease [26], where first solutions were also suggested. The high-level goal is to have n players agree on an output-value, where some (dishonest) players might try to prevent the others from reaching agreement. The potential dishonesty of players is modeled by considering a central adversary who corrupts players. The three most typical corruption types are active corruption (the adversary takes full control over the player), passive corruption (the adversary sees the player's internal state), and fail corruption (the adversary can make the player crash at some point during the protocol).

BA comes in two flavors, namely *Consensus* and *Broadcast*. In Consensus, every player has an input and it is required that they all agree on an output-value y (consistency), where if all correct players have the same input x then the output is $y = x$

⋆ Work done while the author was at ETH Zurich.

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 281–292, 2011.

(validity). In Broadcast, only one player, called the *sender*, has input, and the requirements are that all players should agree on an output-value y (consistency), such that if the sender correctly follows the protocol then y equals his input (validity)[1].

A protocol is said to *perfectly $\mathcal{A}$-securely realize* Consensus or Broadcast, if it achieves the above properties with probability 1, in the presence of a computationally unbounded adversary $\mathcal{A}$. If a protocol satisfies the above properties, except with negligible probability, in the presence of a computationally bounded (resp. unbounded) adversary, then we say this protocol *computationally* (resp. *statistically*) *$\mathcal{A}$-securely realizes* the corresponding primitive.

Known results. The first results on BA considered a threshold adversary who actively corrupts up to t players. In particular, in [27,26] it was shown that when no setup is assumed, Consensus and Broadcast are possible if and only if less than a third of the players are malicious (i.e., $t < n/3$). This model has been extensively studied [10,29,12,8,9,2,17] and protocols with optimal resiliency and complexity (communication and computation) polynomial in the number of players were suggested. Later solutions [11,4,28,6] considered a setting where a setup allowing digital signatures is available, and showed that Broadcast tolerating an arbitrary number of cheaters ($t < n$) is possible, whereas Consensus is possible if and only if $t < n/2$; for both primitives corresponding protocols with optimal resiliency and complexity polynomial in the number of players were suggested[2]. Lamport and Fischer [25] considered an adversary who can fail corrupt up to t players, and showed that any $n - 1$ players being fail corrupted can be tolerated for Broadcast. The above results were unified in [18] where it was shown that if at most t_a players are actively corrupted and, simultaneously, at most t_f are fail corrupted, and no setup is assumed, then $3t_a + t_f < n$ is a tight bound on feasibility of BA. In [23], it was observed that when a setup is assumed then the existing protocols for Broadcast and Consensus do not work for an adversary who can actively and, simultaneously, passively corrupt players. The reason is that in such a model the signatures of passively corrupted players are not reliable, as the adversary knows the signing keys and can trivially fake them. In [21] it is shown that given a Public-Key Infrastructure (PKI), an adversary who can actively corrupt up to t_a players and passively corrupt up to t_p players can be tolerated for Consensus if and only if $2t_a + \min\{t_a, t_p\} < n$.

Our Contributions. We put forward a player-centric approach to BA by introducing the class $\text{PCBA} = \{\text{PCBA}(\mathcal{C})\}_{\mathcal{C}\subseteq\mathcal{P}}$ parametrized by subsets $\mathcal{C}$ of the player set $\mathcal{P}$. Each primitive $\text{PCBA}(\mathcal{C}) \in \text{PCBA}$ has the same consistency property as traditional BA but the validity property is defined with respect to the specific set $\mathcal{C}$. In fact, Broadcast (with sender p) and Consensus are special cases of PCBA primitives for $\mathcal{C} = \{p\}$ and $\mathcal{C} = \mathcal{P}$, respectively.

We prove general negative and positive results translating feasibility statements for different PCBA primitives (i.e., for $\text{PCBA}(\mathcal{C})$ with different choices of $\mathcal{C}$), in the pres-

[1] We point out that some works use the word "persistency" to refer to the validity property of Consensus; furthermore, in some works the term Byzantine agreement refers exclusively to Consensus.

[2] In fact feasibility of Broadcast for $t < n$ when a setup is available was also proved in [26], but the suggested protocol has exponential communication complexity.

ence of a mixed active/passive adversary. In particular, we show under which conditions we can construct PCBA($\mathcal{C}$) if we assume PCBA($\mathcal{C}'$) for $\mathcal{C}' \neq \mathcal{C}$. This characterization allows to translate feasibility results for PCBA($\mathcal{C}$) for specific choices of $\mathcal{C}$ to results about the traditional notions of BA (i.e., Broadcast and Consensus) and vice-versa. Furthermore, we provide exact feasibility bounds for PCBA, for an arbitrary choice of $\mathcal{C}$, tolerating a general adversary who might actively and passively corrupt players, simultaneously. Our results are for perfect security and, assuming a setup which allows for generation and verification of digital signatures, for statistical and computational security. Our characterization specifies the set of players who can securely broadcast their input. In fact, as we show, there are non-trivial adversary structures for which this set is neither empty nor the complete player set $\mathcal{P}$. To the best of our knowledge, this is the first work to explore this asymmetry of Broadcast. Note that in this model, with the exception of perfect security, exact bounds are not even known for traditional BA. All our protocols are efficient in the size of the player set and the representation of the inputs. Furthermore, unless some signature is forged, our protocols achieve perfect security.

As an extension of our results, we show how to define PCBA in a setting where the adversary can actively, passively, and fail corrupt players, simultaneously. For this setting, we give an exact feasibility bound for computationally secure PCBA($\mathcal{P}$) (aka Consensus), assuming a PKI. This result answers an open problem from ASIACRYPT 2008 that concerns feasibility of computationally secure multi-party computation (MPC) and secure function evaluation (SFE) in this model. In particular, in [23], a complete characterization of computationally secure MPC and SFE *assuming Broadcast* was proved. Because an exact bound for Consensus is trivially necessary for SFE (hence, also for MPC) and sufficient for Broadcast, our result fills the gap left open in that work.

Related Work. BA in the general adversary model was considered in [16,1], where exact feasibility bounds for an adversary who can actively corrupt and/or fail corrupt players were proved. However, these works consider the model without a setup and their impossibility results are *only* for Consensus. We point out that, in that model, adding passive corruption makes no difference for Consensus [5]. Feasibility of BA with active and passive corruption (and a trusted PKI) was previously studied in [21] for Consensus, and in [20] for Consensus and Broadcast. In both works, a threshold adversary is considered; the corresponding bound for Consensus is $2t_a + \min\{t_a, t_p\} < n$. In such a threshold world, constructing BA protocols for the corresponding bound turns out to be less involved than in the general-adversary setting, as one can consider the cases $t_a \leq t_p$ and $t_a > t_p$ separately, and construct one protocol for each. Finally, the intermediate ground between Broadcast and Consensus was partially explored in [13], where a variant of Consensus was considered with the property that if more than $n/2$ honest parties have the same input-value then the output is this value.

2 The Model

We consider a set $\mathcal{P} = \{p_1, \ldots, p_n\}$ of n players who can communicate with each other through a complete network of bilateral synchronous authenticated channels. Furthermore, we consider a *general active/passive adversary*, i.e., the adversary's corruption capability is characterized by an adversary structure which is a monotone set of pairs of

player sets, i.e., $\mathcal{Z} = \{(A_1, E_1), \ldots, (A_m, E_m)\}$ (for some m). The adversary chooses a class in $\mathcal{Z}$ non-adaptively i.e., before the beginning of the protocol; this class is denoted as $Z^\star = (A^\star, E^\star)$ and is called the *actual adversary class* or simply the actual adversary. The players in $A^\star$ and $E^\star$ are actively and passively corrupted, respectively. Note that $Z^\star$ is not known to the players and appears only in the security analysis. For notational simplicity we assume that $A \subseteq E$ for any $(A, E) \in \mathcal{Z}$ (intuitively, an actively corrupted player can behave as being passively corrupted). To simplify the description, we adopt the following convention: Whenever a player does not receive a message (when expecting one), or receives a message outside of the expected range, then the special symbol $\perp$ is taken for this message. Moreover, we say that a player is *correct* at a certain point of the protocol if he has followed the protocol instructions correctly up to that point.

Digital Signatures. For computational and statistical security, we assume a trusted setup which allows the players to generate and verify digital signatures, e.g., a PKI, with the respective security. We make the standard assumption on the security of the used signature-scheme, namely existential unforgeability under chosen-message attacks. In slight abuse of notation, we refer to signatures that unconditionally satisfy this definition, except with negligible probability, as *information theoretically (i.t.)*, or *statistically* secure. Note the no construction of such i.t. secure signatures is known. Nevertheless, for our construction i.t. pseudo-signatures of the type used in [28] would also be sufficient. We denote by $\text{sig}_i(x)$ the signature of player p_i (i.e., generated using p_i's private key) to message x. We say that some value σ_i is a *valid* signature with signer p_i (or simply p_i's valid signature) on a message x, if the signature-verification algorithm (given p_i's public key) accepts this signature as valid for the message x. Without loss of generality, we assume that every signature includes a unique signer ID, round ID, and message ID so that it can be linked to the signer and the specific round of the protocol in which it was generated.

Passive Corruption and Forgery. Passive corruption allows the adversary to see the internal state of the corrupted players. This includes the private (signing) keys of these players. Hence, for a passively corrupted player p_i, the adversary can trivially produce signatures with signer p_i on any message of her choice. Therefore, we will only use the term "forgery" for signatures of players who are not passively (or actively) corrupted.

3 Definition and Reductions

Consensus and Broadcast differ in their respective validity property. In particular, Broadcast defines validity with respect to the input of one specific player, the sender, whereas Consensus considers the inputs of *every* player in $\mathcal{P}$. A natural question which arises is: "why should one restrict the definition of BA primitives to these two extreme cases?" In fact, one can find real world scenarios where agreement on the inputs of a subset of parties is desirable, e.g., a network where a dedicated set of master-routers needs to agree on the status of certain links, in order to compute routing-paths. Furthermore, considering such intermediate cases might lead to more efficient protocols for BA and, more general, secure distributed computation in cases where only a subset of the

parties need to provide input. This leads naturally to the definition of a new class of BA primitives, called *player-centric BA* and denoted as PCBA $= \{\text{PCBA}(\mathcal{C})\}_{\mathcal{C}\subseteq\mathcal{P}}$, which is parametrized by non-empty subsets $\mathcal{C}$ of the player set $\mathcal{P}$. All the members of the class PCBA have the same consistency property as in the original definitions of BA, but the validity property of each PCBA$(\mathcal{C}) \in$ PCBA is defined on the inputs of the players in $\mathcal{C}$. More precisely, in PCBA$(\mathcal{C})$, every $p_i \in \mathcal{C}$ has an input x_i and the goal is that all players in $\mathcal{P}$ agree on an output-value y, such that if every non-actively corrupted player in $\mathcal{C}$ has input x, then $y = x$. More formally, we say that a protocol *perfectly $\mathcal{Z}$-securely realizes PCBA*$(\mathcal{C})$ among the players in $\mathcal{P}$, if it satisfies the following properties in the presence of a $\mathcal{Z}$-adversary:

- (Consistency) There exists some y such that every $p_j \in \mathcal{P} \setminus A^\star$ outputs y[3].
- ($\mathcal{C}$-Validity) If every $p_i \in \mathcal{C} \setminus A^\star$ has the same input $x_i = x$, then every $p_j \in \mathcal{P} \setminus A^\star$ outputs $y = x$.

If a protocol satisfies the above properties except with negligible probability in the presence of a computationally bounded (resp. unbounded) $\mathcal{Z}$-adversary, then we say that the protocol *computationally* (resp. *statistically*) *$\mathcal{Z}$-securely realizes* PCBA$(\mathcal{C})$. As in most of the synchronous BA literature, all the protocols presented in this work trivially satisfy the following termination property: For every $p_i \in \mathcal{P} \setminus A^\star$ the protocol terminates after a finite number of rounds; to save space we omit it in our security analysis. We point out that the above definition requires that the inputs of all non-actively corrupted players (even those that are passively corrupted) are considered. This is the most natural way of defining PCBA in the mixed active/passive model and it is consistent with the past literature on secure distributed computation tolerating a mixed adversary, e.g. [14, 22, 24, 5, 23], as well as the literature tolerating passive corruption only (semi-honest model), e.g., [30, 19, 7].

Remark 1 (Authenticated Channels). Consistently with the existing BA literature, our definition of PCBA$(\mathcal{C})$ requires that the inputs of all $p \in \mathcal{C} \setminus A^\star$ (even those that are passively corrupted) are considered. To meet this requirement, authenticated channels are necessary. Indeed, when such channels are not given and are simulated, e.g., by digital signatures or MACs, then this requirement can trivially be violated. For a detailed discussion on secure computation without authentication we refer to [3].

Note that Broadcast (with sender p) and Consensus are special cases of PCBA for $\mathcal{C} = \{p\}$ and $\mathcal{C} = \mathcal{P}$, respectively. However, most past results on feasibility of Broadcast, including the ones considering a general adversary [22, 1], are concerned with whether or not there exists a protocol which achieve PCBA$(\{p\})$ *for every* $p \in \mathcal{P}$. In the remaining of this section we prove results which allow us to translate statements about feasibility of PCBA$(\mathcal{C})$ for different choices of $\mathcal{C} \subseteq \mathcal{P}$. All results in the current section hold for all three security levels, i.e., perfect, statistical, and computational; furthermore, all the negative results hold even when a trusted key-setup allowing digital signatures is assumed. The proofs have been moved to the full version of this paper.

An Inherent Impossibility. As with Consensus, the definition of PCBA$(\mathcal{C})$ only makes sense if there are no two actively corruptible sets that cover the set $\mathcal{C}$. More precisely, for a player set $\mathcal{C} \subseteq \mathcal{P}$, let $\text{C}^{(2)}(\mathcal{P}, \mathcal{Z}, \mathcal{C})$ denote the following condition:

[3] Recall that $A^\star$ denotes the set of actively corrupted players.

$$\mathrm{C}^{(2)}(\mathcal{P},\mathcal{Z},\mathcal{C}) \Leftrightarrow \forall (A_1,E_1),(A_2,E_2) \in \mathcal{Z}:\ A_1 \cup A_2 \neq \mathcal{C}$$

One can verify that when $\mathrm{C}^{(2)}(\mathcal{P},\mathcal{Z},\mathcal{C})$ does not hold, then no protocol can achieve $\mathrm{PCBA}(\mathcal{C})$, as the parties would have to be able to distinguish the setting where the players in A_1 are corrupted from the setting where the players in A_2 are corrupted (even when the corrupted players behave correctly).

The following lemma states that if for a non-empty set $\mathcal{C}$ a $\mathcal{Z}$-secure protocol for $\mathrm{PCBA}(\mathcal{C})$ exists and the adversary cannot actively corrupt every $p_i \in \mathcal{C}$ simultaneously, then Broadcast with sender *any* player $p_i \in \mathcal{P}$ (i.e., any player in the complete player set) is possible.

Lemma 1 (Broadcast from PCBA($\mathcal{C}$)). *If for some (non-empty) set $\mathcal{C} \subseteq \mathcal{P}$ there exists a protocol for $\mathcal{Z}$-securely realizing PCBA($\mathcal{C}$) and the condition $\forall (A,E) \in \mathcal{Z}: \mathcal{C} \not\subseteq A$ holds, then for every $p \in \mathcal{P}$ there exists a protocol which $\mathcal{Z}$-securely realizes PCBA($\{p\}$) (i.e., Broadcast with sender p).*

The above lemma can be generalized to compare arbitrary subsets of $\mathcal{P}$ with respect to feasibility of PCBA as follows:

Lemma 2 (PCBA($\mathcal{C}'$) from PCBA($\mathcal{C}$)). *If for a (non-empty) set $\mathcal{C} \subseteq \mathcal{P}$, there exists a protocol for $\mathcal{Z}$-securely realizing PCBA($\mathcal{C}$) and the condition $\forall (A,E) \in \mathcal{Z}: \mathcal{C} \not\subseteq A$ holds, then for every (non-empty) set $\mathcal{C}' \subseteq \mathcal{P}$, for which the condition $\big((|\mathcal{C}'| = 1) \vee \mathrm{C}^{(2)}(\mathcal{P},\mathcal{Z},\mathcal{C}')\big)$ holds, there exists a protocol which $\mathcal{Z}$-securely realizes PCBA($\mathcal{C}'$).*

Corollary 1. *Assuming that for some (non-empty) set $\mathcal{C} \subseteq \mathcal{P}$ the condition $\mathrm{C}^{(2)}(\mathcal{P},\mathcal{Z},\mathcal{C})$ holds, there exists a protocol for $\mathcal{Z}$-securely realizing PCBA($\mathcal{C}$) if and only if for every (non-empty) $\mathcal{C}' \subseteq \mathcal{C}$ for which $\mathrm{C}^{(2)}(\mathcal{P},\mathcal{Z},\mathcal{C}')$ holds there exists a protocol which $\mathcal{Z}$-secure realizes PCBA($\mathcal{C}'$).*

4 Perfect Security

In this section we study the case of perfect security and prove an exact bound for player-centric BA tolerating a general active/passive adversary. The bound is stated in the following theorem:

Theorem 1. *Assuming $|\mathcal{P}| \geq 3$[4], there exists a perfectly $\mathcal{Z}$-secure PCBA($\mathcal{C}$) protocol for some $\mathcal{C} \subseteq \mathcal{P}$ if and only if the condition $\mathrm{C}^{\mathit{perf}}_{\mathrm{PCBA}}(\mathcal{P},\mathcal{Z},\mathcal{C})$ holds, where $\mathrm{C}^{\mathit{perf}}_{\mathrm{PCBA}}(\mathcal{P},\mathcal{Z},\mathcal{C}) \iff \mathrm{C}^{(3)}(\mathcal{P},\mathcal{Z},\mathcal{C}) \wedge \big(|\mathcal{C}| = 1 \vee \mathrm{C}^{(2)}(\mathcal{P},\mathcal{Z},\mathcal{C})\big)$, and $\mathrm{C}^{(3)}(\mathcal{P},\mathcal{Z},\mathcal{C}) \iff \forall (A_1,E_1),(A_2,E_2),(A_3,E_3) \in \mathcal{Z}:\ A_1 \cup A_2 \cup A_3 \neq \mathcal{P}$.*

The sufficiency of the above condition is straight-forward: In [15] a Consensus (i.e., $\mathrm{PCBA}(\mathcal{P})$) protocol was given which is $\mathcal{Z}$-secure when $\mathrm{C}^{(3)}(\mathcal{P},\mathcal{Z},\cdot)$ holds. Because the condition $\mathrm{C}^{(3)}(\mathcal{P},\mathcal{Z},\cdot)$ implies $\mathrm{C}^{(2)}(\mathcal{P},\mathcal{Z},\mathcal{P})$, if $|\mathcal{C}| = 1$ then the sufficiency follows from Lemma 1 (feasibility of broadcast with sender the player $p \in \mathcal{C}$), otherwise Lemma 2 implies that there exist a $\mathrm{PCBA}(\mathcal{C})$ protocol for every $\mathcal{C}$ for which $\mathrm{C}^{(2)}(\mathcal{P},\mathcal{Z},\mathcal{C})$ holds. The necessity of the condition $\mathrm{C}^{\mathit{perf}}_{\mathrm{PCBA}}(\mathcal{P},\mathcal{Z},\mathcal{C})$ for $\mathrm{PCBA}(\mathcal{C})$ is proved in the full version of this paper.

[4] The case $|\mathcal{P}| < 3$ is of no interest, as $\mathrm{PCBA}(\mathcal{C})$ is either impossible (which happens when $|\mathcal{C}| = 2$ and $\mathrm{C}^{(2)}(\mathcal{P},\mathcal{Z},\mathcal{C})$ is violated) or trivial.

5 Statistical and Computational Security (With Setup)

In this section we consider $\mathcal{Z}$-secure player-centric Byzantine Agreement in a setting where a setup allowing secure signatures. e.g., a Public-Key Infrastructure (PKI), is assumed. We point out that, unless some signature is forged, all the protocols in this section are perfectly secure. Therefore, our constructed protocols are as secure as the underlying signature scheme. The following theorem, states an exact bound for feasibility of PCBA($\mathcal{C}$) for an arbitrary set $\mathcal{C} \subseteq \mathcal{P}$, tolerating a $\mathcal{Z}$-adversary who can actively and passively corrupt players.

Theorem 2. *Assuming that* $|\mathcal{P}| \geq 3$ *and a setup which allows for generation/verification of computationally (resp. statistically) secure digital signatures is given, there exists a protocol which computationally (resp. statistically)* $\mathcal{Z}$*-secure realizes PCBA*($\mathcal{C}$) *for a non-empty set* $\mathcal{C} \subseteq \mathcal{P}$ *if and only if the condition* $\mathrm{C}^{\mathrm{c/s}}_{\mathrm{PCBA}}(\mathcal{P}, \mathcal{Z}, \mathcal{C})$ *holds, where* $\mathrm{C}^{\mathrm{c/s}}_{\mathrm{PCBA}}(\mathcal{P}, \mathcal{Z}, \mathcal{C}) \Leftrightarrow \mathrm{C}^{(5)}_{\mathrm{PCBA}}(\mathcal{P}, \mathcal{Z}, \mathcal{C}) \wedge \big((|\mathcal{C}| = 1) \vee \mathrm{C}^{(2)}(\mathcal{P}, \mathcal{Z}, \mathcal{C})\big)$*, and*

$$\mathrm{C}^{(5)}_{\mathrm{PCBA}}(\mathcal{P}, \mathcal{Z}, \mathcal{C}) \Longleftrightarrow \begin{cases} \forall (A_1, E_1), (A_2, E_2), (A_3, E_3) \in \mathcal{Z} : \\ A_1 \cup A_2 \cup \big(E_1 \cap E_2 \cap A_3\big) = \mathcal{P} \Rightarrow (E_1 \cap E_2 \cap A_3) \cap \mathcal{C} = \emptyset \end{cases}$$

The necessity of the condition is proved in the full version. In the remaining of this section, we prove the sufficiency of $\mathrm{C}^{\mathrm{c/s}}_{\mathrm{PCBA}}(\mathcal{P}, \mathcal{Z}, \mathcal{C})$ for the existence of $\mathcal{Z}$-secure PCBA($\mathcal{C}$). The proof proceeds in two steps: In a first step (Sub-section 5.1), we construct a PCBA($\{p\}$) protocol which is $\mathcal{Z}$-secure when the condition $\mathrm{C}^{(5)}_{\mathrm{PCBA}}(\mathcal{P}, \mathcal{Z}, \{p\})$ is satisfied. In a second step (Sub-section 5.2), we use this protocol to construct a protocol for PCBA($\mathcal{C}$) which is $\mathcal{Z}$-secure when $\mathrm{C}^{\mathrm{c/s}}_{\mathrm{PCBA}}(\mathcal{P}, \mathcal{Z}, \mathcal{C})$ is satisfied.

5.1 PCBA($\{p\}$) (Broadcast with Sender p)

For simplicity we construct a *bit*-broadcast protocol; using standard techniques, one can extend it to broadcast any message. The high-level idea is the following: first, in a distribution phase, the sender p sends his input and his signature on it to every player; in a second phase, all the players run a protocol to establish a consistent view on the sender's value. Although this sounds similar to the standard approach for constructing a Broadcast protocol [8,11] (i.e., first have p multi-send his input and then invoke Consensus), our second phase cannot be realized by a Consensus protocol as the condition $\mathrm{C}^{(5)}_{\mathrm{PCBA}}(\mathcal{P}, \mathcal{Z}, \{p\})$ is weaker than $\mathrm{C}^{\mathrm{c/s}}_{\mathrm{PCBA}}(\mathcal{P}, \mathcal{Z}, \mathcal{P})$ which is necessary for Consensus. Nevertheless, to realize the second phase we use an approach which is inspired by the methodology of [8] for achieving Consensus. More precisely, we build sub-protocols which achieve gradually stronger consistency properties, and compose them in a clever way to construct the Broadcast protocol. We denote these sub-protocols as MakeConsistent, GradeConsistency, and UseKing.

Because our protocols cannot achieve Consensus, the message which is distributed by the sender in the first phase plays a central role in the construction. In particular, in each sub-protocol, the players have input this message along with the sender's signature. To deal with a sender who never sends his signature to any player, we use the following technical trick: Each $p_i \in \mathcal{P}$ keeps a local bit (throughout the whole protocol) α_i which indicates whether or not, according to p_i's view, the sender p is actively corrupted.

Initially $\alpha_i = 0$. If p_i detects that the sender is misbehaving then he sets $\alpha_i := 1$. When some p_i has set $\alpha_i = 1$ then p_i will accept any value as p's correct signature on any bit (without invoking the signature verification algorithm). This trick makes sure that, when no player receives a signature from p then every p_i sets $\alpha_i := 1$ and hence any value is acceptable as p's signature on any bit. As syntactic sugar we say that some value σ is a (p, p_i)-*acceptable* signature on x if $p_i \in \mathcal{P}$ accepts σ as p's signature on x (i.e, σ is valid or $\alpha_i = 1$); for a player set $\mathcal{C}$ we say that σ is a $(p, \mathcal{C})$-*acceptable* signature on x if for every $p_i \in \mathcal{C}$ the value σ is a (p, p_i)-acceptable signature on x[5].

Remark 2 (The use of signatures). The player use signatures for detecting and exposing passive corruption. In particular, often in our sub-protocols a player p_i is instructed to send a message m to some p_j along with his signature $\text{sig}_i(m)$ on it, so that p_j has a proof that he indeed got this message from this sender in the corresponding round[6]. However, if p_i is passively corrupted, the adversary might introduce into the protocol arbitrary signatures with signer p_i. To cope with this behavior, we have p_j forward p_i's signature $\text{sig}_i(m)$ to every player as soon as he receives it. This way, if some party presents to p_j a (fake) signature with sender p_i and the same ID's (round and message ID) as $\text{sig}_i(m)$, then p_j can prove to every player that $p_i \in E^\star$.

In the following we sketch the sub-protocols MakeConsistent, GradeConsistency, and UseKing, and specify the achieved security properties. Due to space limitation, many of the protocols along with their security analysis have been removed, and will be included in the full version of this paper. We stress that the security properties are guaranteed *only when* at the beginning of the protocol the following two conditions hold: (1) Every $p_i \in \mathcal{P} \setminus A^\star$ holds as input a pair (x_i, σ_i) such that σ_i is a $(p, \mathcal{P})$-acceptable signature on x_i with round ID corresponding to the distribution phase, and (2) when $p \in \mathcal{P} \setminus A^\star$ then for some x and for every $p_i \in \mathcal{P} \setminus A^\star$: $x_i = x$ and $\alpha_i = 0$. To keep the description short, we introduce the following notation: We say that the input state is (p, x)-*well-formed* if it satisfies the above two conditions. The invariant in all sub-protocols is that if the input state is (p, x)-well-formed then the output state is also (p, x)-well-formed.

Distribution Phase. Before describing the three sub-protocols, we describe the protocol Send (see next page) used in the distribution phase for p to send his input along with his signature. The protocol achieves a (p, x)-well-formed state, where x is p's input.

Protocol Send($\mathcal{P}, \mathcal{Z}, p, x$)

1. p sends x along with his signature on it to every p_j who denotes the received value as x_j and the corresponding signature as σ_j; if p_j does not receive a message with a valid signature then he sets $\alpha_j := 1$.
2. Every $p_i \in \mathcal{P}$ forwards (x_i, σ_i) to every p_j. If p_j did not receive a consistent pair (x_j, σ_j) in Step 1 and receives one in Step 2 from some p_i then he adopts it. Otherwise p_j sets $(x_j, \sigma_j) := (0, \perp)$.

[5] Note that, for any $p_j \neq p$, only p_j's valid signature, i.e., the one matching p_j's public key, can be (p_j, p_i)-acceptable.

[6] Recall that every signature has a unique signer ID, message ID, and round ID.

Lemma 3. *Assuming that no signature is forged, protocol* $\mathsf{Send}(\mathcal{P}, \mathcal{Z}, p, x)$ *achieves a* (p, x)*-well-formed state.*

MakeConsistent. As the name suggests, protocol MakeConsistent ensures that there are no inconsistencies among the outputs of non-actively corrupted players (however, some of them might output a special symbol "n/v", denoting that they have no output-value)[7]. On a high level, the protocol works as follows: each p_i sends his input along with his signature on it to every party p_j; subsequently, every p_j forwards all the received values/signatures to every p_k, who uses the received signatures to detect passive corruption (see Remark 2). Each p_k checks if his view is consistent with pre-agreement on some value x; if this is the case, then he outputs x, otherwise he outputs "n/v".

GradeConsistency. In GradeConsistency each $p_i \in \mathcal{P}$ outputs a pair (y_i, g_i) (along with a $(p, \mathcal{P})$-acceptable signature on y_i), where y_i is p_i's actual output-value and $g_i \in \{0, 1\}$ is a bit, called p_i's *grade*. The grade g_i has the meaning of the confidence level of p_i on the fact that agreement on y_i has been reached. In particular, if $g_i = 1$ for some $p_i \in \mathcal{P} \setminus A^\star$ then (p_i knows that) $y_j = y_i$ for every $p_j \in \mathcal{P} \setminus A^\star$. Moreover, when the non-actively corrupted players pre-agree on a value x, then they all output x with grade 1. The protocol GradeConsistency is included in the full version.

UseKing. Here, there exists a distinguished player $p_k \in \mathcal{P}$, called the *king*. If $p_k \in \mathcal{P} \setminus A^\star$, then every $p_j \in \mathcal{P}$ outputs the same value $y_j = y$ along with a $(p, \mathcal{P})$-acceptable signature on it (consistency). Furthermore, independent of whether or not the king is correct, pre-agreement is preserved. The idea is simple: invoke GradeConsistency, and have the king forward his output to every party p_j, who adopts it when his grade (in GradeConsistency) was $g_j = 0$, and ignores it otherwise.

Broadcast. We next describe our $\mathrm{PCBA}(\{p\})$ (aka Broadcast) protocol: First, protocol Send is invoked. Subsequently, for $k = 1, \ldots, n$, UseKing is invoked with king $p_k \in \mathcal{P}$. The input to the first iteration of UseKing is the state which is output from Send, whereas for $k = 2, \ldots, n$ the input to the kth iteration of UseKing is the output of the $(k-1)$th iteration. If $p \in \mathcal{P} \setminus A^\star$ then the well-formness of the state (Lemma 3) ensures that from the first iteration of UseKing all $p_i \in \mathcal{P} \setminus A^\star$ have as input the input-bit of p, and the $\{p\}$-validity property of UseKing ensures that this agreement will be preserved in all iterations. In any case, $\mathrm{C}^{(5)}_{\mathrm{PCBA}}(\mathcal{P}, \mathcal{Z}, \{p\})$ ensures that there is at least one honest player $p_\ell \in \mathcal{P}$; at the latest during the iteration of UseKing with king p_ℓ, agreement on the output will be achieved (king consistency), which is maintained in all future iterations of UseKing ($\{p\}$-validity).

Lemma 4. *Assuming that no signature is forged and the condition* $\mathrm{C}^{(5)}_{\mathrm{PCBA}}(\mathcal{P}, \mathcal{Z}, \{p\})$ *holds, the protocol* $\mathsf{Broadcast}(\mathcal{P}, \mathcal{Z}, p, x)$ *$\mathcal{Z}$-securely realizes PCBA$(\{p\})$ (i.e, Broadcast with sender p).*

5.2 PCBA($\mathcal{C}$) for an Arbitrary $|\mathcal{C}| \geq 1$

Using our $\mathrm{PCBA}(\{p\})$ protocol, i.e., protocol Broadcast, we can achieve $\mathrm{PCBA}(\mathcal{C})$ for an arbitrary $\mathcal{C} \subseteq \mathcal{P}$. The corresponding protocol, denoted as $\mathsf{PCBA}_{\mathcal{C}}$, is described in the following; the input of each $p_i \in \mathcal{C}$ is denoted as x_i.

[7] Observe that "n/v" is not the same as $\perp$.

Protocol $\mathsf{PCBA}_{\mathcal{C}}(\mathcal{P}, \mathcal{Z}, x_1, \ldots, x_{|\mathcal{C}|})$

1. Every $p_i \in \mathcal{C}$ uses Broadcast to Broadcast x_i.
2. For each p_j: if $|\mathcal{C}| = 1$, then output the broadcasted value; otherwise, if there exists unique x s.t. $\exists (A, E) \in \mathcal{Z} : \{p_i \mid x_i \neq x\} \subseteq A$ then output x, otherwise output 0.

Lemma 5. *Assuming that no signature is forged, if the condition* $\mathrm{C}^{(5)}_{\mathrm{PCBA}}(\mathcal{P}, \mathcal{Z}, \mathcal{C}) \wedge \big((|\mathcal{C}| = 1) \vee \mathrm{C}^{(2)}(\mathcal{P}, \mathcal{Z}, \{p\})\big)$ *holds for a set* $\mathcal{C} \subseteq \mathcal{P}$*, then the protocol* $\mathsf{PCBA}_{\mathcal{C}}$ *perfectly* $\mathcal{Z}$*-securely realized PCBA*$(\mathcal{C})$.

6 Extension: Adding Fail Corruption

We extend the definition of PCBA to consider an adversary who can actively, passively and fail corrupt players, simultaneously. In this setting, a general adversary is described by a structure which is a collection of triples (instead of pairs) of player sets, $\mathcal{Z} = \{(A_1, E_1, F_1), \ldots, (A_m, E_m, F_m)\}$, where the adversary of class (A, E, F) actively corrupts the players in A, passively corrupts the players in E, and fail corrupts the players in F. Consistently with our previous notation, we denote by $(A^\star, E^\star, F^\star)$ the class corresponding to the adversary's actual corruption choice. To simplify the notation, we assume that $A \subseteq F$ (anyway, an actively corrupted player can behave as being fail corrupted). We say that a player is *alive* at a certain point of the protocol if he has not crashed until that point. Note that a fail corrupted player is both correct and alive until the point when he crashes. In the following we give the definition of player-centric BA in this extended model and prove an exact feasibility bound for PCBA$(\mathcal{P})$ (aka Consensus) for computational security assuming a trusted PKI.

A natural question which arises when fail corruption is considered in PCBA is how the inputs of fail corrupted players are accounted in the validity condition. Following the intuition that a fail corrupted player never gives a wrong input (but might give no input) we extend the definition of PCBA as follows: every p_i has input x_i and the goal is to agree on an output-value y, such that if every $p_i \in \mathcal{C} \setminus A^\star$ who is alive at the beginning has the same input $x_i = x$ then $y = x$. More formally, let $\mathcal{C} \subseteq \mathcal{P}$; we say that a protocol *perfectly* $\mathcal{Z}$*-securely realizes PCBA*$(\mathcal{C})$, if it satisfies the following properties in the presence of a $\mathcal{Z}$-adversary:

- (Consistency) There exists some y such that every player p_i who is correct until the end of the protocol outputs y.
- ($\mathcal{C}$-Validity) If every $p_i \in \mathcal{C} \setminus A^\star$ who is alive at the beginning of the protocol has the same input $x_i = x$, then every (alive) $p_j \in \mathcal{P} \setminus A^\star$ outputs $y \in \{x, \text{“n/v”}\}$, where $y = x$, unless all players in $\mathcal{C} \setminus A^\star$ have crashed during the protocol execution.
- (Termination) For $p_i \in \mathcal{P} \setminus A^\star$ the protocol terminates after a finite number of rounds.

When a protocol satisfies the above properties except with negligible probability in the presence of a computationally bounded (resp. unbounded) adversary, then we say that the protocol *computationally* (resp. *statistically*) $\mathcal{Z}$*-securely realizes* PCBA$(\mathcal{C})$.

PCBA$(\mathcal{P})$ (Consensus). We next give an complete characterization of tolerable adversaries for computationally secure PCBA$(\mathcal{P})$ (Consensus) in our model. Recall that the corresponding bound for (perfect) security without a setup can be derived in a

straight-forward manner from [1] (see [5] for details). The necessary and sufficient condition is stated in the following theorem which is proved in the full version of this paper:

Theorem 3. *Assuming $|\mathcal{P}| \geq 3$, if a setup allowing digital signatures is given, then a set of players $\mathcal{P}$ can computationally $\mathcal{Z}$-securely realize Consensus if and only if the following condition holds:* $\forall (A_1, E_1, F_1), (A_2, E_2, F_2), (A_3, E_3, F_3) \in \mathcal{Z} : A_1 \cup A_2 \cup \big(\big((E_1 \cap F_1) \cup (E_2 \cap F_2) \cup (E_1 \cap E_2)\big) \cap A_3\big) \cup (F_1 \cap F_2 \cap F_3) \neq \mathcal{P}$.

7 Conclusions and Open Problems

Most existing definitions of Byzantine Agreement are of an all-or-nothing type. Motivated by the above observation, we introduced a new class of player-centric BA primitives, denoted as PCBA $= \{\text{PCBA}(\mathcal{C})\}_{\mathcal{C} \subseteq \mathcal{P}}$, which is parametrized by non-empty subsets $\mathcal{C}$ of the player set. For each PCBA$(\mathcal{C}) \in$ PCBA, the validity condition depends on the inputs of the players in $\mathcal{C}$. We proved general negative and positive results, which associate feasibility of PCBA for different choices of the set $\mathcal{C}$. Furthermore, for a general active/passive adversary we proved exact feasibility bounds for PCBA for any choice of the set $\mathcal{C}$, for all three security levels, i.e., perfect, statistical, and computational. Moreover, we showed that there might be adversaries who can be tolerated for some specific sender p to broadcast his input, but not for any $p' \in \mathcal{P}$ being the sender.

As an extension of our results we provide a definition of PCBA tolerating an active/passive/fail adversary and prove an exact bound for PCBA$(\mathcal{P})$ (aka Consensus) in this model. A complete characterization of PCBA for an arbitrary set $\mathcal{C}$ in this extended model is an interesting research direction. However, the unexpectedly high complexity of the tight bound for Consensus gives some evidence that such a characterization might be too complicated, and raises the question whether one should look for a different trichotomy of corruption types.

References

1. Altmann, B., Fitzi, M., Maurer, U.: Byzantine agreement secure against general adversaries in the dual failure model. In: DISC 1999. LNCS, vol. 1693, pp. 123–139. Springer, Heidelberg (1999)
2. Bar-Noy, A., Dolev, D., Dwork, C., Strong, H.: Shifting gears: Changing algorithms on the fly to expedite Byzantine agreement. Inf. Comput. 97(2), 205–233 (1992)
3. Barak, B., Canetti, R., Lindell, Y., Pass, R., Rabin, T.: Secure computation without authentication. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 361–377. Springer, Heidelberg (2005)
4. Baum-Waidner, B., Pfitzmann, B., Waidner, M.: Unconditional Byzantine agreement with good majority. In: Jantzen, M., Choffrut, C. (eds.) STACS 1991. LNCS, vol. 480, pp. 285–295. Springer, Heidelberg (1991)
5. Beerliova-Trubiniova, Z., Fitzi, M., Hirt, M., Maurer, U., Zikas, V.: MPC vs. SFE: Perfect security in a unified corruption model. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 231–250. Springer, Heidelberg (2008)
6. Beerliová-Trubíniová, Z., Hirt, M., Riser, M.: Efficient Byzantine agreement with faulty minority. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 393–409. Springer, Heidelberg (2007)
7. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation. In: STOC 1988, pp. 1–10 (1988)

8. Berman, P., Garray, J., Perry, J.: Towards optimal distributed consensus. In: FOCS 1989, pp. 410–415 (1989)
9. Coan, B., Welch, J.: Modular Construction of Efficient Byzantine Agreement Protocols. In: PODC 1989, pp. 295–306 (1989)
10. Dolev, D., Fischer, M., Fowler, R., Lynch, N., Strong, H.: An efficient algorithm for Byzantine agreement without authentication. Information and Control 52(3), 257–274 (1982)
11. Dolev, D., Strong, H.: Polynomial algorithms for multiple processor agreement. In: STOC 1982, pp. 401–407 (1982)
12. Feldman, P., Micali, S.: Optimal algorithms for Byzantine agreement. In: STOC 1988, pp. 148–161 (1988)
13. Fitzi, M., Garray, J.: Efficient player-optimal protocols for strong and differential consensus. In: PODC 2003, pp. 211–220 (2003)
14. Fitzi, M., Hirt, M., Maurer, U.: Trading correctness for privacy in unconditional multi-party computation. In: Krawczyk, H. (ed.) CRYPTO 1998. LNCS, vol. 1462, pp. 121–136. Springer, Heidelberg (1998)
15. Fitzi, M., Hirt, M., Maurer, U.: General adversaries in unconditional multi-party computation. In: Lam, K.-Y., Okamoto, E., Xing, C. (eds.) ASIACRYPT 1999. LNCS, vol. 1716, pp. 232–246. Springer, Heidelberg (1999)
16. Fitzi, M., Maurer, U.: Efficient Byzantine agreement secure against general adversaries. In: Kutten, S. (ed.) DISC 1998. LNCS, vol. 1499, pp. 134–148. Springer, Heidelberg (1998)
17. Garay, J., Moses, Y.: Fully polynomial Byzantine agreement in t+1 rounds. In: STOC 1993, pp. 31–41 (1993)
18. Garay, J., Perry, K.: A continuum of failure models for distributed computing. In: Segall, A., Zaks, S. (eds.) WDAG 1992. LNCS, vol. 647, pp. 153–165. Springer, Heidelberg (1992)
19. Goldreich, O., Micali, S., Wigderson, A.: How to play any mental game — a completeness theorem for protocols with honest majority. In: STOC 1987, pp. 218–229 (1987)
20. Gordon, S., Katz, J., Kumaresan, R., Yerukhimovich, A.: Authenticated broadcast with a partially compromised public-key infrastructure. In: Dolev, S., Cobb, J., Fischer, M., Yung, M. (eds.) SSS 2010. LNCS, vol. 6366, pp. 144–158. Springer, Heidelberg (2010)
21. Gupta, A., Gopal, P., Bansal, P., Srinathan, K.: Authenticated Byzantine generals in dual failure model. In: Kant, K., Pemmaraju, S.V., Sivalingam, K.M., Wu, J. (eds.) ICDCN 2010. LNCS, vol. 5935, pp. 79–91. Springer, Heidelberg (2010)
22. Hirt, M., Maurer, U.: Complete characterization of adversaries tolerable in secure multi-party computation. In: PODC 1997, pp. 25–34 (1997)
23. Hirt, M., Maurer, U., Zikas, V.: MPC vs. SFE: Unconditional and computational security. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 1–18. Springer, Heidelberg (2008)
24. Ishai, Y., Kushilevitz, E., Lindell, Y., Petrank, E.: On combining privacy with guaranteed output delivery in secure multiparty computation. In: Dwork, C. (ed.) CRYPTO 2006. LNCS, vol. 4117, pp. 483–500. Springer, Heidelberg (2006)
25. Lamport, L., Fischer, M.: Byzantine generals and transaction commit protocols. Technical Report Opus 62, SRI International (Menlo Park CA), TR (1982)
26. Lamport, L., Shostak, R., Pease, M.: The Byzantine generals problem. ACM Transactions on Programming Languages and Systems 4(3), 382–401 (1982)
27. Pease, M., Lamport, L.: Reaching agreement in the presence of faults. Journal of the ACM 27, 228–234 (1980)
28. Pfitzmann, B., Waidner, M.: Unconditional Byzantine agreement for any number of faulty processors. In: Finkel, A., Jantzen, M. (eds.) STACS 1992. LNCS, vol. 577, pp. 337–350. Springer, Heidelberg (1992)
29. Toueg, S., Perry, K., Srikanth, T.: Fast distributed agreement. SIAM J. Comput. 16(3), 445–457 (1987)
30. Yao, A.: Protocols for secure computations. In: FOCS 1982, pp. 160–164 (1982)

Limits on the Computational Power of Random Strings

Eric Allender[1], Luke Friedman[1], and William Gasarch[2]

[1] Department of Computer Science, Rutgers University, Piscataway, NJ 08855, USA
{allender,lbfried}@cs.rutgers.edu
[2] Dept. of Computer Science, University of Maryland, College Park, MD, 20742
gasarch@cs.umd.edu

Abstract. How powerful is the set of random strings? What can one say about a set A that is efficiently reducible to R, the set of Kolmogorov-random strings? We present the first *upper bound* on the class of computable sets in P^R and NP^R.

The two most widely-studied notions of Kolmogorov complexity are the "plain" complexity $C(x)$ and "prefix" complexity $K(x)$; this gives two ways to define the set "R": R_C and R_K. (Of course, each different choice of universal Turing machine U in the definition of C and K yields another variant R_{C_U} or R_{K_U}.) Previous work on the power of "R" (for *any* of these variants [1,2,9]) has shown

- BPP $\subseteq \{A : A \leq^{\mathrm{p}}_{tt} R\}$.
- PSPACE $\subseteq \mathrm{P}^R$.
- NEXP $\subseteq \mathrm{NP}^R$.

Since these inclusions hold irrespective of low-level details of how "R" is defined, we have e.g.: NEXP $\subseteq \Delta_1^0 \cap \bigcap_U \mathrm{NP}^{R_{K_U}}$. ($\Delta_1^0$ is the class of computable sets.)

Our main contribution is to present the *first* upper bounds on the complexity of sets that are efficiently reducible to R_{K_U}. We show:

- BPP $\subseteq \Delta_1^0 \cap \bigcap_U \{A : A \leq^{\mathrm{p}}_{tt} R_{K_U}\} \subseteq$ PSPACE.
- NEXP $\subseteq \Delta_1^0 \cap \bigcap_U \mathrm{NP}^{R_{K_U}} \subseteq$ EXPSPACE.

Hence, in particular, PSPACE is sandwiched between the class of sets Turing- and truth-table-reducible to R.

As a side-product, we obtain new insight into the limits of techniques for derandomization from uniform hardness assumptions.

1 Introduction

In this paper, we take a significant step toward providing characterizations of some important complexity classes in terms of efficient reductions to *non-computable* sets. Along the way, we obtain new insight into the limits of techniques for derandomization from uniform hardness assumptions.

Our attention will focus on the set of Kolmogorov random strings:

Definition 1. *Let $K(x)$ be the prefix Kolmogorov complexity of the string x. Then*

$$R_K = \{x : K(x) \geq |x|\}.$$

(More complete definitions of Kolmogorov complexity can be found in Section 2. Each universal prefix Turing machine U gives rise to a slightly different measure K_U, and hence to various closely-related sets R_{K_U}.)

The first steps toward characterizing complexity classes in terms of efficient reductions to R_K came in the form of the following curious inclusions:

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 293–304, 2011.

Theorem 1. *The following inclusions hold:*

- BPP $\subseteq \{A : A \leq^{\mathrm{p}}_{tt} R_K\}$ *[9].*
- PSPACE $\subseteq \mathrm{P}^{R_K}$ *[2].*
- NEXP $\subseteq \mathrm{NP}^{R_K}$ *[1].*

We call these inclusions "curious" because the upper bounds that they provide for the complexity of problems in BPP, PSPACE and NEXP are not even computable; thus at first glance these inclusions may seem either trivial or nonsensical.

A key step toward understanding these inclusions in terms of standard complexity classes is to invoke one of the guiding principles in the study of Kolmogorov complexity: The choice of universal machine should be irrelevant. Theorem 1 actually shows that problems in certain complexity classes are *always* reducible to R_K, no matter *which* universal machine is used to define $K(x)$. That is, we have

- BPP $\subseteq \Delta_1^0 \cap \bigcap_U \{A : A \leq^{\mathrm{p}}_{tt} R_{K_U}\}$.
- PSPACE $\subseteq \Delta_1^0 \cap \bigcap_U \mathrm{P}^{R_{K_U}}$.
- NEXP $\subseteq \Delta_1^0 \cap \bigcap_U \mathrm{NP}^{R_{K_U}}$.

(Recall that Δ_1^0 is the class of computable sets.) The question arises as to how powerful the set $\Delta_1^0 \cap \bigcap_U \{A : A \leq_r R_{K_U}\}$ is, for various notions of reducibility $\leq_r$. Until now, no computable upper bound was known for the complexity of any of these classes. (Earlier work [1] did give an upper bound for a related class defined in terms of $\leq^{\mathrm{p}}_{dtt}$ reductions – but this only provided a characterization of P in terms of a class of polynomial-time reductions, which is much less compelling than giving a characterization where the set R_K is actually providing some useful computational power.)

Our main results show that the class of problems reducible to R_K in this way *does* have bounded complexity; hence it is at least plausible to conjecture that some complexity classes can be characterized in this way:

Main Results

- $\Delta_1^0 \cap \bigcap_U \{A : A \leq^{\mathrm{p}}_{tt} R_{K_U}\} \subseteq$ PSPACE.
- $\Delta_1^0 \cap \bigcap_U \mathrm{NP}^{R_{K_U}} \subseteq$ EXPSPACE.

A stronger inclusion is possible for "monotone" truth-table reductions ($\leq^{\mathrm{p}}_{mtt}$). We show that $\Delta_1^0 \cap \bigcap_U \{A : A \leq^{\mathrm{p}}_{mtt} R_{K_U}\} \subseteq \mathrm{coNP} \cap \mathrm{P/poly}$.

Combining our results with Theorem 1 we now have:

- BPP $\subseteq \Delta_1^0 \cap \bigcap_U \{A : A \leq^{\mathrm{p}}_{tt} R_{K_U}\} \subseteq$ PSPACE $\subseteq \Delta_1^0 \cap \bigcap_U \mathrm{P}^{R_{K_U}}$.
- NEXP $\subseteq \Delta_1^0 \cap \bigcap_U \mathrm{NP}^{R_{K_U}} \subseteq$ EXPSPACE.

In particular, note that PSPACE is sandwiched in between the classes of computable problems that are reducible to R_K via truth-table and Turing reductions.

Our results bear a superficial resemblence to results of Book *et al.* [6,7,8], who also studied decidable sets that are reducible in some sense to algorithmically random sets. However, there is really not much similarity at all. Book *et al.* studied the class **ALMOST-R**

Definition 2. *Let* **R** *be a reducibility (e.g,* $\leq^p_m$ *or* $\leq^p_T$*). Then* **ALMOST-R** *is the class of all* B *such that* $\{A : B$ *is* **R***-reducible to* $A\}$ *has measure 1.*

Book *et al.* showed that **ALMOST-R** can be characterized as the class of decidable sets that are **R**-reducible to sets whose characteristic sequences are (for example) Martin-Löf random. Thus using such sets as oracles is roughly the same as providing access to unbiased independent coin flips. But a set whose characteristic sequence is Martin-Löf random will contain many strings of low Kolmogorov complexity, and a set such as R_K is very far from being Martin-Löf random. Another contrast is provided by noting that **ALMOST**-NP $=$ AM, whereas NP^{R_K} contains NEXP.

1.1 Derandomization from Uniform Hardness Assumptions

Papers in derandomization that follow the "hardness vs. randomness" paradigm generally fall into two categories: those that proceed from uniform hardness assumptions, and those that rely on nonuniform hardness. The nonuniform approach yields the more general and widely-applicable tools. For instance, the work of Babai, Fortnow, Nisan, and Wigderson [5] shows that, given *any* function f, one can construct a pseudorandom generator[1] G_f such that, given *any* test T that distinguishes the pseudorandom distribution generated by G_f from the uniform distribution, one can conclude that f has polynomial-size circuits that have access to T as an oracle. (Or, in terms of the contrapositive, if f does *not* have small circuits relative to T, then the generator G_f is secure against T.) As discussed by Gutfreund and Vadhan [10], invoking the terminology introduced by Reingold et al. [16], this is an example of a *fully black-box* construction.

In contrast, there has been much more modest success in providing derandomization tools from *uniform* hardness assumptions. The canonical example here comes from Impagliazzo and Wigderson [12] as extended by Trevisan and Vadhan [17]. They show that for certain functions f in PSPACE (including some PSPACE-complete problems), one can build a pseudorandom generator G_f such that, given *any* test T that distinguishes the pseudorandom distribution generated by G_f from the uniform distribution, one can conclude that f can be computed in BPP^T. (Or, in terms of the contrapositive, if f is *not* in BPP^T, then the generator G_f is secure against T.) As discussed by Gutfreund and Vadhan [10], this is still an example of a black-box reduction (although it is not *fully* black box), since it works for *every* test T that distinguishes random from pseudorandom.

Furthermore, Trevisan and Vadhan showed that a fully black-box construction, such as is used in the nonuniform setting, can *not* yield derandomizations from uniform assumptions [17]. Their argument is nonconstructive, in the sense that they derive a contradiction from the assumption that a fully black-box construction exists, as opposed to presenting a concrete function f for which no fully-black-box construction will work.

Gutfreund and Vadhan considered the question in more detail, and showed that there is no function f outside of $\mathrm{BPP}^{\mathrm{NP}}$ for which there is a uniform *non-adaptive* black-box reduction from computing f to distinguishing random from pseudorandom [10]. (That is, if the BPP^T algorithm computing f is non-adaptive in the sense that the list of oracle queries is computed before any queries are asked, then $f \in \mathrm{BPP}^{\mathrm{NP}}$.) Gutfreund and Vadhan also considered a more general class of black-box reductions, still maintaining

[1] The generator G_f in general is computable in PSPACE^f, but in many important instances it is computable in P^f.

some restrictions on "adaptiveness", and showed that there is no function f outside of PSPACE for which there is a uniform black-box reduction of this restricted class, from computing f to distinguishing random from pseudorandom [10]. It appears that no limits were known at all, for general BPP reductions to distinguishing random from pseudorandom.

We state an easy consequence of our main theorems:

Theorem 2. *If there is a uniform black-box reduction from a computable function f to distinguishing random from pseudorandom, then $f \in$* EXPSPACE.

In addition to shedding light on the limitations of black-box reductions to distinguishing random from pseudorandom, Theorem 2 also provides some insight about the completeness of problems related to the Minimum Circuit Size Problem:

The Minimum Circuit Size Problem (MCSP) (given the truth table of a Boolean function f, and a number s, does f have a circuit of size s?) is a well-known example of a problem in NP that is believed to lie outside of P, but is not known to be NP-complete [13,2,4].

For a complexity class $\mathcal{C}$, let MCSP$^{\mathcal{C}}$ denote an analog of MCSP for $\mathcal{C}$: given the truth table of a Boolean function f, and a number s, does f have an *oracle* circuit of size s, where the oracle is for the standard complete set for $\mathcal{C}$?

It is known [2,4] that MCSP$^{\mathcal{C}}$ is complete for $\mathcal{C}$ under P/poly reductions, where $\mathcal{C}$ is any of $\{$PSPACE, EXP, NEXP, EXPSPACE, doubly- or triply-exponential time and space, etc. $\ldots\}$. Completeness under *uniform* reductions is known only for two cases: MCSP$^{\text{PSPACE}}$ is complete for PSPACE under ZPP reductions, and MCSP$^{\text{EXP}}$ is complete for EXP under NP reductions [2]. In the former case, completeness is proved via the Impagliazzo-Wigderson generator [12]; in the latter case completeness is proved via a uniform black-box NP-reduction to distinguishing random from pseudorandom.

Now consider doubly-exponential space EEXPSPACE. Is MCSP$^{\text{EEXPSPACE}}$ complete for EEXPSPACE under ZPP or even NP reductions? As a consequence of Theorem 2, this question cannot be answered using the techniques that were used to resolve the analagous questions for PSPACE and EXP, which were black-box reductions to distinguishing random from pseudorandom.

2 Background and Definitions

For a fixed universal Turing machine U, the plain Kolmogorov complexity of a string x, $C(x)$, is the size of the shortest s such that $U(s) = x$. This paper is concerned much more with *prefix complexity*:

1. A *prefix Turing machine* is a Turing machine M such that, for all x, if $M(x)$ halts then, for all $y \neq \lambda$, $M(xy)$ does not halt. That is, the domain of M is a prefix code.
2. Let M be a prefix Turing machine. Define $K_M(x)$ to be the size of the shortest s such that $M(s) = x$.
3. A *universal prefix Turing machine* is a prefix Turing machine U such that, for any prefix Turing machine M, there is a constant c such that for all $x, K_U(x) \leq K_M(x) + c$.

We select some universal prefix Turing machine U and call $K_U(x)$ the *prefix complexity of* x. As usual, we delete the subscript in this case, and let $K(x)$ denote the prefix complexity of x. The arbitrary choice of U affects $K(x)$ by at most an additive constant, and in most instances where prefix complexity is studied, the particular choice of U is deemed to be irrelevant. Note however, that in this paper it is important to consider K_U for various machines U.

If f is a function mapping some domain to the naturals $\mathbb{N}$, then $ov(f)$, the overgraph of f, is $\{(x, y) : f(x) \leq y\}$. (For instance, $ov(K_U) = \{(x, y) :$ there is an s, $|s| \leq y$, such that $U(s) = x\}$).

The following definition was used implicitly by Muchnik and Positselsky [15]: A *Prefix Free Entropy Function* f is a function from $\{0,1\}^*$ to $\mathbb{N}$ such that

- $\sum_{x\in\{0,1\}^*} 2^{-f(x)} \leq 1$ and
- $ov(f)$ is computably enumerable (c.e.)

The canonical example of a prefix free entropy function is $K(x)$. (That K is a prefix free entropy function follows from the Kraft Inequality; see e.g. [14, Theorem 1.11.1].)

Note that if f is a prefix free entropy function, then 2^{-f} is a special case of what Li and Vitányi call a *Lower Semicomputable Discrete Semimeasure* [14, Definition 4.2.2]. We recall the *Coding Theorem* (see [14, Theorem 4.3.3]), the proof of which yields the following important relationship between prefix free entropy functions and prefix complexity.

Theorem 3. *Let f be a prefix free entropy function. Given a machine computing f, one can construct a prefix machine M such that $f(x) = K_M(x) - 1$.*

We will make use of the following easy propositions.

Proposition 1. *Let U and U' be prefix Turing machines. Then there is a prefix machine U'' such that $K_{U''}(x) = \min(K_U(x), K_{U'}(x)) + 1$.*

Proposition 2. *Given any machine U and constant c, there is a machine U' such that $K_U(x) + c = K_{U'}(x)$*

In this paper we consider four types of reductions: truth table reductions, monotone truth table reductions, anti-monotone reductions, and Turing reductions.

- *Truth-table reductions*. For a complexity class $\mathcal{R}$ and languages A and B, we say that A *$\mathcal{R}$-truth-table-reduces* to B ($A \leq_{tt}^{\mathcal{R}} B$) if there is a function q computable in $\mathcal{R}$, such that, on an input $x \in \{0,1\}^*$, q produces an encoding of a circuit λ and a list of queries $q_1, q_2, \ldots q_m$ so that for $a_1, a_2, \ldots, a_m \in \{0,1\}$ where $a_i = 1$ if and only if $q_i \in B$, it holds that $x \in A$ if and only if $\lambda(a_1 a_2 \cdots a_m) = 1$. If the function q is polynomial time computable, we say that A *polynomial-time-truth-table-reduces* to B ($A \leq_{tt}^{p} B$).
- *Monotone truth-table reductions*. In the scenario above, if the circuit λ computes a monotone function (i.e. changing any input bit of the function from 0 to 1 cannot change the output of the function from 1 to 0), then we say that A *$\mathcal{R}$-monotone-truth-table-reduces* to B ($A \leq_{mtt}^{\mathcal{R}} B$). If the function q is polynomial time computable, we say that A *polynomial-time-monotone-truth-table-reduces* to B ($A \leq_{mtt}^{\mathrm{p}} B$).

- *Anti-monotone truth-table reductions*. In the scenario above, if the circuit λ computes an anti-monotone function (i.e. $\neg\lambda$ is monotone), then we say that A *$\mathcal{R}$-anti-monotone-truth-table-reduces* to B ($A \leq^{\mathcal{R}}_{amtt} B$). If the function q is polynomial time computable, we say that A *polynomial-time-anti-monotone-truth-table-reduces* to B ($A \leq^{\mathrm{p}}_{amtt} B$).
- *Turing reductions*. We say that A *$\mathcal{R}$-Turing reduces* to B ($A \leq^{\mathcal{R}}_{T} B$) if there is an oracle Turing machine in class $\mathcal{R}$ that accepts A when given B as an oracle.

3 Main Results

Theorem 4. $\Delta_1^0 \cap \bigcap_U \{A : A \leq^p_{tt} R_{K_U}\} \subseteq$ PSPACE

Proof. Due to space limitations, we provide an overview of the proof here; details can be found in a more complete version of the paper [3]. The main idea of the proof can be seen as a blending of the approach of [1] with the techniques that are used to prove Theorem 2.7 of [15].

We will actually prove the statement

$$\Delta_1^0 \cap \bigcap_U \{A : A \leq^p_{tt} ov(K_U)\} \subseteq \text{PSPACE}. \tag{1}$$

The theorem follows, since any query "$x \in R_{K_U}$?" can always be modified to the equivalent query "$(x, |x| - 1) \notin ov(K_U)$?", so

$$\Delta_1^0 \cap \bigcap_U \{A : A \leq^p_{tt} R_{K_U}\} \subseteq \Delta_1^0 \cap \bigcap_U \{A : A \leq^p_{tt} ov(K_U)\}.$$

To prove the statement (1) it suffices to show that

$$L \notin \text{PSPACE} \Rightarrow \exists \text{ a universal prefix machine } U \text{ s.t. } L \not\leq^p_{tt} ov(K_U). \tag{2}$$

Let $L \notin$ PSPACE be given. Our strategy will be to use a diagonalization technique to carefully construct a universal prefix machine U such that $L \not\leq^p_{tt} ov(K_U)$. To do this we will use the standard prefix complexity function K, together with a function $F : \{0,1\}^* \to \mathbb{N}$ that we will construct, to form a function $H : \{0,1\}^* \to \mathbb{N}$ with the following properties.

1. F is a total function and $ov(F)$ is c.e.
2. $H(x) = \min(K(x) + 4, F(x) + 2)$.
3. $\sum_{x \in \{0,1\}^*} 2^{-H(x)} \leq \frac{1}{4}$.
4. $L \not\leq^p_{tt} ov(H)$.

Claim 1: Given the above properties, $H = K_{U'}$ for some universal prefix machine U' (which by Property 4 ensures that (2) holds).

It remains to show that for a given $L \notin$ PSPACE we can always construct an H with the desired properties. Let us first informally discuss the ideas before providing the formal construction.

Our control over H comes from our freedom in constructing the function F. The construction will occur in stages – at any given time in the construction there will be a "current" version of F which we will denote by F^*. Similarly, there will be a "current" version of K denoted by K^*, which represents our knowledge of K at a given stage. At all times, H^*, our "current" version of H, will be defined as $\min(K^*(x)+4, F^*(x)+2)$.

Originally we set $F^*(x) = 2|x| + 2$ and K^* as the empty function. At each stage of the construction we will assume that a new element (x, y) is enumerated into $ov(K)$ according to some fixed enumeration of $ov(K)$. (This is possible since $ov(K)$ is c.e.) When this occurs K^* is updated by setting $K^*(x) = \min(K^*(x), y)$. (Since K^* is a partial function, it is possible that $K^*(x)$ was previously undefined. In this case we set $K^*(x) = y$.) Similarly, during the construction at times we will modify F by enumerating elements into $ov(F)$. Whenever we enumerate an element (x, y) into $ov(F)$, F^* is updated by setting $F^*(x) = \min(F^*(x), y)$.

Let $\gamma_1, \gamma_2, \ldots$ be a list of all possible polynomial time truth table reductions from L to $ov(H)$. This is formed in the usual way: we take a list of all Turing Machines and put a clock of $n^i + i$ on the ith one and we will interpret the output as an encoding of a Boolean circuit on atoms of the form "$(z, r) \in ov(H)$".

We need to ensure that $L \not\leq^p_{tt} ov(H)$. We break this requirement up into an infinite number of requirements:

$$R_e : \gamma_e \text{ is not a polynomial-time tt-reduction of } L \text{ to } ov(H)$$

At stage e of the construction we will begin to attempt to satisfy the requirement R_e. For a particular input x, let $\gamma_e(x)$ be an encoding of a circuit $\lambda_{e,x}$. The output of the circuit $\lambda_{e,x}$ is determined by the truth values of the atoms "$z \in ov(H)$" that label the inputs to the circuit. Define $\lambda_{e,x}[H']$ to be the truth value obtained by taking the circuit $\lambda_{e,x}$ and for each atom "$(z, r) \in ov(H)$" using the truth value of "$(z, r) \in ov(H')$ in its place. In order to satisfy the requirement R_e, we would like to find some x such that $\lambda_{e,x}[H] \neq L(x)$, where $L(x)$ is the characteristic function of L. The problem is that at a given stage s we can "guess" at the value of $\lambda_{e,x}[H]$ by computing $\lambda_{e,x}[H^*]$, but in general we cannot know the value of $\lambda_{e,x}[H]$ for sure, because as H^* evolves the value of $\lambda_{e,x}[H^*]$ may change. The main difficulty is that the function K is out of our control and determining whether $(z, r) \in ov(K)$ is in general an uncomputable task.

We do have *some* influence over the situation though due to our control of F. Indeed, for any atom "$(z, r) \in ov(H)$", we can ensure that the truth value of the atom is 1 by enumerating $(z, r - 2)$ into $ov(F)$. (Note that for all x, the value of $H^*(x)$ can only decrease over time). We have to be careful about making these types of changes though; if we are too liberal in modifying F we may violate the condition $\sum_{x\in\{0,1\}^*} 2^{-H(x)} \leq 1/4$ in the process. Thus the construction becomes a balancing act – we will try to use F to satisfy R_e while at the same time maintaining the invariant that $\sum_{x\in\{0,1\}^*} 2^{-H^*(x)} \leq 1/4$. (In particular, if F_s is the function F^* at the beginning of stage s, for all x we will not want $\lim_{s\to\infty} F_s(x)$ to be very much smaller than $K(x)$).

As part of our solution, for each R_e we will find a suitable witness x and set up a game $\mathcal{G}_{e,x}$ played between us (making moves by enumerating elements into $ov(F)$), and K, who makes moves by enumerating elements into $ov(K)$. (Even though

elements are obliviously enumerated into $ov(K)$ according to some fixed enumeration we will treat K as if it is a willful adversary). The witness x will be chosen so that we have a winning strategy; as long as K continues to make legal moves we can respond with changes to F (our own legal moves) that both assure that R_e is satisfied and that $\sum_{x\in\{0,1\}^*} 2^{-H^*(x)} \leq 1/4$. Our ability to find such a witness x follows from our assumption that the language L is not in PSPACE; if no such witness exists, then membership in L reduces to finding which player has a winning strategy in one of these games, which can be done in PSPACE.

It is possible that K will cheat by enumerating elements into $ov(K)$ in such a way that it plays an illegal move. In this case we will simply destroy the game $\mathcal{G}_{e,x}$ and start all over again with a new game $\mathcal{G}_{e,x'}$, using a different witness x'. However we will be able to show that if K cheats infinitely often on games associated with a particular requirement R_e, then $\sum_{x\in\{0,1\}^*} 2^{-K(x)}$ diverges. This contradicts K being a prefix complexity function. Hence K can only cheat finitely often[2].

The requirements $R_1, R_2, R_3, \ldots$ are listed in priority ordering. If during stage s a move is played on a game $\mathcal{G}_{e,x}$, we say that R_e is "acting". In this case for all $e < e' \leq s$, if $\mathcal{G}_{e',y}$ is the game associated with $R_{e'}$ currently being played, we destroy this game and start a new game $\mathcal{G}_{e',y'}$ with some new witness y'. When this happens we say that each of the $R_{e'}$ has been "injured" by R_e. The reason this works in the end is that at some point $R_1, R_2, \ldots, R_{e-1}$ have stopped acting, so R_e will no longer ever be injured by some higher priority requirement.

3.1 Description of the Game

Now let us describe one of the games $\mathcal{G}_{e,x}$ in more depth and provide some analysis of the game. Let the inputs to the Boolean circuit $\lambda_{e,x}$ (encoded by $\gamma_e(x)$) be labeled by the atoms $\{(z_1, r_1), \ldots, (z_k, r_k)\}$. Let $X_e = \{z_1, \ldots, z_k\}$. Note that the queries in this reduction are of the form: "Is $H(z_i) \leq r_i$?". If $H^*(z_i) \leq r_i$ then we already know $H(z_i) \leq r_i$, so we can replace that input to the circuit with the value *TRUE* and simplify the circuit accordingly. Renumber the z's, rename k to again be the number of questions, and rename X_e to be the set of all z's being asked about. When we are done we have atoms $\{(z_1, r_1), \ldots, (z_k, r_k)\}$ and we know that $(\forall z_i \in X_e)[H^*(z_i) > r_i]$.

We make one more change to X_e. If there exists an element z_i such that $z_i \in X_e$ and $z_i \in X_{e'}$ for some $e' < e$, then changing H^* on the value z_i during the game $\mathcal{G}_{e,x}$ could affect the game associated with the requirement $R_{e'}$, which would upset our priority ordering. Hence we will take

$$X_e = X_e - \bigcup_{e'<e} X_{e'}.$$

This will ensure that R_e cannot injure any $R_{e'}$ with $e' < e$.

[2] This reliance on the convergence of $\sum_{x\in\{0,1\}^*} 2^{-K(x)}$ is the key reason why our proof does not go through in the case where we consider the plain complexity $C(x)$ as opposed to the prefix complexity.

Let $H^*_{e,x}$ be the function H^* when the game $\mathcal{G}_{e,x}$ is first constructed. Let $\epsilon = 2^{-e-i_e-5}$. (How i_e is determined will be explained later). The game $\mathcal{G}_{e,x}$ is played on a labeled DAG. The label of each node of the DAG has the following two parts:

1. A function h that maps X_e to $\mathbb{N}$. The function h provides conjectured values for H restricted to X_e. The function h will be consistent with $H^*_{e,x}$ in that $(\forall i)[h(z_i) \leq H^*_{e,x}(z_i)]$.
2. A truth value *VAL*, which is the value of $\lambda_{e,x}$ assuming that $(\forall z \in X_e)[H(z) = h(z)]$. Note that this will be either YES or NO indicating that either, under assumption $(\forall z \in X_e)[H(z) = h(z)]$, $\lambda_{e,x}$ thinks $x \in L$ or thinks $x \notin L$.

There is a separate node in the DAG for every possible such function h.

Let us place an upper bound on the size of this DAG. The set X_e contains at most $|x|^e$ queries. For any query z_i, $H(z_i)$ can take at most $2|z_i| + 4$ values (since it is always bounded by $F^*(z_i) + 2$). Note also that $|z_i| \leq |x|^e$. Thus there are at most $(2|x|^e + 4)^{|x|^e}$ possible choices for h. For all large x this is bounded by $2^{|x|^{2e}}$, so note that we can represent a *particular* node in the DAG with $|x|^{2e} + 1$ bits.

We now describe the start node and how to determine the edges of the DAG.

1. There is a node (h, VAL) where $h = H^*_{e,x}$ restricted to X_e. This is the start node and has indegree 0.
2. There is an edge from (h, VAL) to (h', VAL') if for all $z_i \in X_e$, $h(z_i) \geq h'(z_i)$ (so it is possible that H^* could at some point evolve from $H^*_{e,x}$ to h, and then at a later point evolve from h to h'.)

The game $\mathcal{G}_{e,x}$ is played between two players, the YES player and the NO player. Each player has a score, which originally is zero, and represents how much the player has been penalized so far in the game. (In other words a high score is bad). The game starts with a token placed on the start node. The YES player goes first (although this choice is arbitrary), after which the players alternate moves.

On a given turn a player can either leave the token where it is or move the token to a new node in the DAG. Suppose a player moves the token from a node t to a node t', where h is the function labeling t and h' is the function labeling t'. In this case we add $\sum_{z_i \in X_e} 2^{-h'(z_i)} - 2^{-h(z_i)}$ to the player's score.

A player can legally move the token from node t to t' if

1. There is an edge from t to t' in the game DAG.
2. The score of the player after making the move does not exceed ϵ.

The YES player wins if the token ends up on a node such that *VAL* = YES, and the NO player wins if the token ends up on a node such that *VAL* = NO. Note that because the game is entirely deterministic, for a given game $\mathcal{G}_{e,x}$, either the YES player has a winning strategy or the NO player has a winning strategy. Let $val(\mathcal{G}_{e,x}) = 1$ if the YES player has a winning strategy on the game $\mathcal{G}_{e,x}$ and $val(\mathcal{G}_{e,x}) = 0$ otherwise.

During the actual construction the games will be played between us (the construction) trying to make the computation go one way, and K (which we do not control) trying to make it go (perhaps) another way. We will always ensure that we play the side of the player who has the winning strategy in the game. We will effect our moves by

enumerating elements into $ov(F)$, which changes F^* and hence H^*. (To move the token to a node labeled with the function h, we modify H^* so that h equals H^* restricted to the set X_e) The K moves will occur when a new element is enumerated into $ov(K)$ at the beginning of each stage, which changes K^* and hence H^*. (In this case K is moving the token to the node in the game DAG labeled by the new H^*).

The key is that the players' scores measure how much the sum $\sum_{x\in\{0,1\}^*} 2^{-H^*(x)}$ has gone up, which we bound by not allowing a player's score to exceed ϵ. (Of course K is oblivious to the rules of the game and will at times cheat – we take this into account as part of our analysis.) One final note: it is possible that K will simply stop playing a game in the middle and never make another move. This will not matter to us in the construction; what is important is that we have a winning strategy and if K does move we always have a winning response.

Theorem 5. $\Delta_1^0 \cap \bigcap_U \{A : A \leq^{\mathrm{p}}_{mtt} R_{K_U}\} \subseteq \mathrm{coNP} \cap \mathrm{P/poly}$

Proof. The containment in P/Poly comes from [1].

Note that a reduction showing $L \leq^{\mathrm{p}}_{mtt} R_{K_U}$ corresponds to an *anti-monotone* reduction to $ov(K_U)$ (where the only queries are of the form "Is $K_U(z) < |z|$?") Thus this same reduction is an anti-monotone reduction from the complement of L to the complement of R_{K_U}. If we replace each Boolean function in this anti-monotone reduction with its complement, we obtain a *monotone* reduction of L to $ov(K_U)$.

Thus it suffices to show that any set that is $\leq^{\mathrm{p}}_{mtt}$-reducible to the overgraph $ov(K_U)$ for every U is in NP.

The proof of this containment is almost identical to the proof of Theorem 4. The only difference is now we consider an arbitrary language $L \notin \mathrm{NP}$, and must show that when a game $\mathcal{G}_{e,x}$ is constructed corresponding to a polynomial time *monotone* truth table reduction γ_e, determining whether $val(\mathcal{G}_{e,x}) = 1$ can be computed in NP. Note that in the monotone case, the NO player of the game has no incentive to ever make a move, as doing so could only change the value of the circuit $\lambda_{e,x}$ from NO to YES. Therefore whether the YES player has a winning strategy in the game depends solely on whether the YES player can legally move the token from the start node to a node u in the game DAG labeled by YES. This is an NP question – the certificate is the node u, which as we have seen can be represented by a polynomial number of bits in $|x|$.

Theorem 6. $\Delta_1^0 \cap \bigcap_U \mathrm{NP}^{R_{K_U}} \subseteq \mathrm{EXPSPACE}$

Proof. An NP-Turing reduction can be simulated by a truth-table reduction computable in exponential time, where all queries have length bounded by a polynomial in the input length. Carrying out the same analysis as in the proof of Theorem 4, but changing the time bound on the truth-table reductions from polynomial to exponential, immediately yields the EXPSPACE upper bound.

4 Perspective and Open Problems

How should one interpret the theorems presented here?

Prior to this work, the inclusion NEXP $\subseteq$ NP^{R_K} was just a curiosity, since it was not clear that it was even meaningful to speak about efficient reductions to an undecidable set. Here, we show that if we view R_K not as merely a single undecidable set, but as a *class* of closely-related undecidable sets (differing only by the "insignificant" choice of the universal Turing machine U), then the computable sets that are always in NP^{R_K} *is* a complexity class sandwiched between NEXP and EXPSPACE. The obvious question is whether this class is actually *equal* to NEXP (or to EXPSPACE). Any characterization of a complexity class in terms of efficient reductions to a class of undecidable sets would raise the possibility of applying techniques from computability theory to questions in complexity theory, where they had seemed inapplicable previously.

One possible objection to the theorems presented here is that they make use of universal Turing machines U that are far from "natural". However, we see little to be gained in trying to formulate a definition of a "natural" universal Turing machine. Even basic questions such as whether there is a truth-table reduction from the Halting Problem to R_K depend on the choice of the universal Turing machine U [15,1], and the only machines for which the answer is known (positive and negative) are all decidedly "unnatural". All of the positive results, showing that problems *are* efficiently reducible to R_K hold using a quite general notion of "universal Turing machine", and we believe that the approach used here and in [1] to "factor out" the idiosyncrasies of individual universal machines is a more productive route to follow.

It is natural to conjecture that our main theorems hold, even if "$\Delta_1^0\cap$" is erased from the statement of the theorems. For instance, if A is in $\bigcap_U$ NP$^{R_{K_U}}$, we conjecture that A is computable.

We also conjecture that all of our main theorems hold if the prefix complexity function $K(x)$ is replaced everywhere by $C(x)$, although the techniques that we use do not seem sufficient to prove this.

Is R_K just as useful as an oracle as $ov(K)$? All of the upper bounds that we have on the classes of sets reducible to R_K hold also for the classes of sets reducible to $ov(K)$. However, it seems much easier to make use of $ov(K)$ than to use R_K. For instance, for *any* computable set A, there is a universal prefix machine U such that $A \leq_{tt}^{\mathrm{p}} ov(K_U)$ (by creating a machine U such that $x \in A$ if and only if $K_U(x)$ is odd). (Details will appear in a more complete version of this work.) In contrast, some of us would conjecture that, for any machine U, if $A \leq_{tt}^{\mathrm{p}} R_{K_U}$, then $A \in$ P/poly. (Some partial results in this direction can be found in [1]; see also [11].)

The theorems presented here all relativize. For instance, for any computable oracle B, if $A \notin$ PSPACE^B, then there is a universal prefix Turing machine U such that $A \not\leq_{tt}^{\mathrm{P^B}} R_{K_U}$. (Note that, for computable oracles B, there is no need to "relativize" R_{K_U}. A similar restatement is possible for noncomputable oracles B, too.) However, it seems quite possible to us that, say, if it were possible to *characterize* NEXP in terms of NP^{R_K}, that this might proceed via nonrelativizable techniques. The types of characterizations of complexity classes that we are investigating are quite different than those that have been studied in the past, and we hope that new insights will result as new connections are explored.

Acknowledgments

We thank Adam Day, Bruno Loff, Russell Impagliazzo, and Danny Gutfreund for helpful comments. The first two authors were supported in part by NSF Grants CCF-0830133 and CCF-0832787.

References

1. Allender, E., Buhrman, H., Koucký, M.: What can be efficiently reduced to the Kolmogorov-random strings? Annals of Pure and Applied Logic 138, 2–19 (2006)
2. Allender, E., Buhrman, H., Koucký, M., van Melkebeek, D., Ronneburger, D.: Power from random strings. SIAM Journal on Computing 35, 1467–1493 (2006)
3. Allender, E., Friedman, L., Gasarch, W.: Limits on the computational power of random strings. Technical Report TR10-139, Electronic Colloquium on Computational Complexity (2010)
4. Allender, E., Koucký, M., Ronneburger, D., Roy, S.: The pervasive reach of resource-bounded Kolmogorov complexity in computational complexity theory. Journal of Computer and System Sciences 77, 14–40 (2010)
5. Babai, L., Fortnow, L., Nisan, N., Wigderson, A.: BPP has subexponential time simulations unless EXPTIME has publishable proofs. Computational Complexity 3, 307–318 (1993)
6. Book, R.V.: On languages reducible to algorithmically random languages. SIAM Journal on Computing 23(6), 1275–1282 (1994)
7. Book, R.V., Lutz, J., Wagner, K.W.: An observation on probability versus randomness with applications to complexity classes. Mathematical Systems Theory 27(3), 201–209 (1994)
8. Book, R.V., Mayordomo, E.: On the robustness of ALMOST-*r*. RAIRO Informatique Théorique et Applications 30(2), 123–133 (1996)
9. Buhrman, H., Fortnow, L., Koucky, M., Loff, B.: Derandomizing from random strings. In: 25th IEEE Conference on Computational Complexity (CCC), pp. 58–63. IEEE Computer Society Press, Los Alamitos (2010)
10. Gutfreund, D., Vadhan, S.P.: Limitations of hardness vs. Randomness under uniform reductions. In: Goel, A., Jansen, K., Rolim, J.D.P., Rubinfeld, R. (eds.) APPROX and RANDOM 2008. LNCS, vol. 5171, pp. 469–482. Springer, Heidelberg (2008)
11. Hitchcock, J.M.: Lower bounds for reducibility to the kolmogorov random strings. In: Ferreira, F., Löwe, B., Mayordomo, E., Mendes Gomes, L. (eds.) CiE 2010. LNCS, vol. 6158, pp. 195–200. Springer, Heidelberg (2010)
12. Impagliazzo, R., Wigderson, A.: Randomness vs. time: de-randomization under a uniform assumption. J. Comput. Syst. Sci. 63(4), 672–688 (2001)
13. Kabanets, V., Cai, J.-Y.: Circuit minimization problem. In: Proc. ACM Symp. on Theory of Computing (STOC), pp. 73–79 (2000)
14. Li, M., Vitanyi, P.: Introduction to Kolmogorov Complexity and its Applications, 3rd edn. Springer, Heidelberg (2008)
15. Muchnik, A.A., Positselsky, S.: Kolmogorov entropy in the context of computability theory. Theoretical Computer Science 271, 15–35 (2002)
16. Reingold, O., Trevisan, L., Vadhan, S.P.: Notions of reducibility between cryptographic primitives. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 1–20. Springer, Heidelberg (2004)
17. Trevisan, L., Vadhan, S.P.: Pseudorandomness and average-case complexity via uniform reductions. Computational Complexity 16(4), 331–364 (2007)

The Decimation Process in Random k-SAT

Amin Coja-Oghlan and Angelica Y. Pachon-Pinzon*

University of Warwick, Mathematics and Computer Science,
Coventry CV4 7AL, UK
{a.coja-oghlan,a.y.pachon-pinzon}@warwick.ac.uk

Abstract. Non-rigorous statistical mechanics ideas have inspired a message passing algorithm called *Belief propagation guided decimation* for finding satisfying assignments of random k-SAT instances. This algorithm can be viewed as an attempt at implementing a certain thought experiment that we call the *decimation process*. In this paper we identify a variety of phase transitions in the decimation process and link these phase transitions to the performance of the algorithm.

1 Introduction

Let $k \geq 3$ and $n > 1$ be integers, let $r > 0$ be a real, and set $m = \lceil rn \rceil$. Let $\boldsymbol{\Phi} = \boldsymbol{\Phi}_k(n, m)$ be a propositional formula obtained by choosing a set of m clauses of length k over the variables $V = \{x_1, \ldots, x_n\}$ uniformly at random. For k, r fixed we say that $\boldsymbol{\Phi}$ has some property $\mathcal{P}$ *with high probability* ('w.h.p.') if $\lim_{n\to\infty} \mathrm{P}[\boldsymbol{\Phi} \in \mathcal{P}] = 1$.

The interest in random k-SAT originates from the experimental observation that for certain densities r the random formula $\boldsymbol{\Phi}$ is satisfiable w.h.p. while a large class of algorithms, including and particularly the workhorses of practical SAT solving such as sophisticated DPLL-based solvers, fail to find a satisfying assignment efficiently [14]. Over the past decade, a fundamentally new class of algorithms have been proposed on the basis of ideas from statistical physics [6,13]. Experiments performed for $k = 3, 4, 5$ indicate that these new 'message passing algorithms', namely *Belief Propagation guided decimation* and *Survey Propagation guided decimation* ('BP/SP decimation'), excel on random k-SAT instances [10]. Indeed, the experiments indicate that BP/SP decimation find satisfying assignments for r close to the threshold where $\boldsymbol{\Phi}$ becomes unsatisfiable w.h.p. Generally, SP is deemed conceptually superior to BP.

For example, in the case $k = 4$ the threshold for the existence of satisfying assignments is conjectured to be $m/n \sim r_4 \approx 9.93$ [12]. According to experiments from [10], SP decimation finds satisfying assignments for densities up to $r = 9.73$. Experiments from [16] suggest that the "vanilla" version of BP decimation succeeds up to $r = 9.05$. Another version of BP decimation (with a different decimation strategy from [6]) succeeds up to $r = 9.24$, again according to experimental data from [10]. By comparison, the currently best rigorously

* Supported by EPSRC grant EP/G039070/2.

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 305–316, 2011.

analyzed algorithm is efficient up to $r = 5.54$ [9], while `zChaff`, a prominent practical SAT solver, becomes ineffective beyond $r = 5.35$ [10].

Since random k-SAT instances have widely been deemed extremely challenging benchmarks, the stellar experimental performance of the physicists' message passing algorithms has stirred considerable excitement. However, the statistical mechanics ideas that BP/SP decimation are based on are highly non-rigorous, and thus a rigorous analysis of these message passing algorithms is an important but challenging open problem. A first step was made in [7], where it was shown that BP decimation does not outperform far simpler combinatorial algorithms for sufficiently large clause lengths k. More precisely, the main result of [7] is that there is a constant $\rho_0 > 0$ (independent of k) such that the 'vanilla' version of BP decimation fails to find satisfying assignments w.h.p. if $r > \rho_0 2^k/k$. By comparison, non-constructive arguments show that w.h.p. $\boldsymbol{\Phi}$ is satisfiable if $r < r_k = 2^k \ln 2 - k$, and unsatisfiable if $r > 2^k \ln 2$ [3,4]. This means that for $k \gg \rho_0$ sufficiently large, BP decimation fails to find satisfying assignments w.h.p. already for densities a factor of (almost) k below the threshold for satisfiability.

The analysis performed in [7] is based on an intricate method for directly tracking the execution of BP decimation. Unfortunately this argument does little to illuminate the conceptual reasons for the algorithms' demise. In particular, [7] does not provide a link to the statistical mechanics ideas that inspired the algorithm. The present paper aims to remedy these defects. Here we study the *decimation process*, an idealized thought experiment that the BP decimation algorithm aims to implement. We show that this experiment undergoes a variety of phase transitions that explain the failure of BP decimation for densities $r > \rho_0 \cdot 2^k/k$. Our results identify phase transitions jointly in terms of the clause/variable density r and with respect to the time parameter of the decimation process. The latter dimension was ignored in the original statistical mechanics work on BP [6,13] but turns out to have a crucial impact on the performance of the algorithm. On a non-rigorous basis, this has been pointed out recently by Ricci-Tersenghi and Semerjian [16], and our results can be viewed as providing a rigorous version of (parts of) their main results. The results of this paper can also be seen as a generalization of the ones obtained in [1] for random k-SAT, and indeed our proofs build heavily upon the techniques developed in that paper.

2 Results

BP decimation is a polynomial-time algorithm that aims to (heuristically) implement the 'thought experiment' shown in Fig. 1 [15,16], which we call the *decimation process*.[1] A moment's reflection reveals that, given a satisfiable input formula Φ, the decimation process outputs a uniform sample from the set of all satisfying assignments of Φ. The obvious obstacle to actually implementing this

[1] Several different versions of BP decimation have been suggested. In this paper we refer to the simplest but arguably most natural one, also considered in [7,15,16]. Other versions decimate the variables in a different order, allowing for slightly better experimental results [6,10].

experiment is the computation of the marginal probability $M_{x_t}(\Phi_{t-1})$ that x_t takes the value 'true' in a random satisfying assignment of Φ_{t-1}, a $\#P$-hard problem in the worst case. Yet the key hypothesis underlying BP decimation is that these marginals can be computed efficiently on *random* formulas by means of a message passing algorithm. We will return to the discussion of BP decimation and its connection to Experiment 1 below.

Experiment 1 ('decimation process'). *Input:* A satisfiable k-CNF Φ.
Result: A satisfying assignment $\sigma : V \to \{0,1\}$ (with 0/1 representing 'false'/'true').

0. Let $\Phi_0 = \Phi$.
1. For $t = 1, \ldots, n$ do
2. Compute the fraction $M_{x_t}(\Phi_{t-1})$ of all satisfying assignments of Φ_{t-1} in which the variable x_t takes the value 1.
3. Assign $\sigma(x_t) = 1$ with probability $M_{x_t}(\Phi_{t-1})$, and let $\sigma(x_t) = 0$ otherwise.
4. Obtain the formula Φ_t from Φ_{t-1} by substituting the value $\sigma(x_t)$ for x_t and simplifying (i.e., delete all clauses that got satisfied by assigning x_t, and omit x_t from all other clauses).
5. Return the assignment σ.

Fig. 1. The decimation process

We are going to study the decimation process when applied to a random formula $\boldsymbol{\Phi}$ for densities $r < 2^k \ln 2 - k$, i.e., in the regime where $\boldsymbol{\Phi}$ is satisfiable w.h.p. More precisely, conditioning on $\boldsymbol{\Phi}$ being satisfiable, we let $\boldsymbol{\Phi}_t$ be the (random) formula obtained after running the first t iterations of Experiment 1. The variable set of this formula is $V_t = \{x_{t+1}, \ldots, x_n\}$, and each clause of $\boldsymbol{\Phi}_t$ consists of *at most* k literals. Let $\mathcal{S}(\boldsymbol{\Phi}_t) \subset \{0,1\}^{V_t}$ be the set of all satisfying assignments of $\boldsymbol{\Phi}_t$. We say that *almost all* $\sigma \in \mathcal{S}(\boldsymbol{\Phi}_t)$ have a certain property $\mathcal{A}$ if $|\mathcal{A} \cap \mathcal{S}(\boldsymbol{\Phi}_t)| = (1 - o(1))|\mathcal{S}(\boldsymbol{\Phi}_t)|$.

We will identify various phase transition that the formulas $\boldsymbol{\Phi}_t$ undergo as t grows from 1 to n. As it turns out, these can be characterized via two simple parameters. The first one is the clauses density $r \sim m/n$. Actually, it will be most convenient to work in terms of

$$\rho = kr/2^k \qquad \text{and} \qquad \theta = 1 - t/n,$$

so that $m/n \sim \rho \cdot 2^k/k$. We will be interested in the regime $\rho_0 \le \rho \le k \ln 2$, where ρ_0 is a constant (independent of k). The upper bound $k \ln 2$ marks the point where satisfying assignments cease to exist [4]. The second parameter θ is the fraction of 'free' variables (i.e., variables not yet assigned by time t).

The symmetric phase. Let Φ be a k-CNF on V, let $1 \le t < n$, let Φ_t be the formula obtained after t steps of the decimation process, and suppose that $\sigma \in \mathcal{S}(\Phi_t)$. A variable $x \in V_t$ is *loose* if there is $\tau \in \mathcal{S}(\Phi_t)$ such that $\sigma(x) \neq \tau(x)$ and $d(\sigma, \tau) \le \ln n$, where $d(\cdot, \cdot)$ denotes the Hamming distance. For any $x \in V_t$ we let $M_x(\Phi_t) = |\{\sigma \in \mathcal{S}(\Phi_t) : \sigma(x) = 1\}| / |\mathcal{S}(\Phi_t)|$ be the marginal probability that x takes the value 'true' in a random satisfying assignment of Φ_t.

Theorem 2. *There are constants $k_0, \rho_0 > 0$ such that for $k \geq k_0$, $\rho_0 \leq \rho \leq k \ln 2 - 2 \ln k$, and*

$$k \cdot \theta > \exp\left[\rho\left(1 + \frac{\ln\ln\rho}{\rho} + \frac{10}{\rho}\right)\right]$$

the random formula $\boldsymbol{\Phi}_t$ has the following properties w.h.p.

1. *In almost all $\sigma \in \mathcal{S}(\boldsymbol{\Phi}_t)$ at least $0.99\theta n$ variables are loose.*
2. *At least $\theta n/3$ variables $x \in V_t$ satisfy $M_x(\boldsymbol{\Phi}_t) \in [0.01, 0.99]$.*
3. *The average distance of two random satisfying assignments is $\geq 0.49\theta n$.*

Intuitively, Theorem 2 can be summarized as follows. In the early stages of the decimation process (while θ is 'big'), most variables in a typical $\sigma \in \mathcal{S}(\boldsymbol{\Phi}_t)$ are loose. Hence, the correlations amongst the variables are mostly local: if we 'flip' one variable in σ, then we can 'repair' the unsatisfied clauses that this may cause by simply flipping another $\ln n$ variables. Furthermore, for at least a good fraction of the variables, the marginals $M_x(\boldsymbol{\Phi}_t)$ are bounded away from 0/1. Finally, as the distance between satisfying assignments is large on average, the set $\mathcal{S}(\boldsymbol{\Phi}_t)$ is 'well spread' over the Hamming cube $\{0,1\}^{V_t}$.

Shattering and rigidity. Let Φ be a k-CNF and let $\sigma \in \mathcal{S}(\Phi_t)$. For an integer $\omega \geq 1$ we call a variable $x \in V_t$ *ω-rigid* if any $\tau \in \mathcal{S}(\Phi_t)$ with $\sigma(x) \neq \tau(x)$ satisfies $d(\sigma, \tau) \geq \omega$. Furthermore, we say that a set $S \subset \{0,1\}^{V_t}$ is *(α, β)-shattered* if it admits a decomposition $S = \bigcup_{i=1}^{N} R_i$ into pairwise disjoint subsets such that the following two conditions are satisfied.

SH1. We have $|R_i| \leq \exp(-\alpha\theta n)|S|$ for all $1 \leq i \leq N$.
SH2. If $1 \leq i < j \leq N$ and $\sigma \in R_i$, $\tau \in R_j$, then $\mathrm{dist}(\sigma, \tau) \geq \beta\theta n$.

Theorem 3. *There are constants $k_0, \rho_0 > 0$ such that for $k \geq k_0$, $\rho_0 \leq \rho \leq k \ln 2 - 2 \ln k$, and*

$$\frac{\rho}{\ln 2}(1 + 2\rho^{-2}) \leq k\theta \leq \exp\left[\rho\left(1 - \frac{\ln\rho}{\rho} - \frac{2}{\rho}\right)\right] \tag{1}$$

the random formula $\boldsymbol{\Phi}_t$ has the following properties w.h.p.

1. *In almost all $\sigma \in \mathcal{S}(\boldsymbol{\Phi}_t)$ at least $0.99\theta n$ variables are $\Omega(n)$-rigid.*
2. *There exist $\alpha = \alpha(k, \rho) > 0, \beta = \beta(k, \rho) > 0$ such that $\mathcal{S}(\boldsymbol{\Phi}_t)$ is (α, β)-shattered.*
3. *At least $\theta n/3$ variables $x \in V_t$ satisfy $M_x(\boldsymbol{\Phi}_t) \in [0.01, 0.99]$.*
4. *The average distance of two random satisfying assignments is at least $0.49\theta n$.*

Thus, if the fraction θ of free variables lies in the regime (1), then in most satisfying $\sigma \in \mathcal{S}(\boldsymbol{\Phi}_t)$ the values assigned to 99% of the variables are linked via long-range correlations: to 'repair' the damage done by flipping a single rigid variable it is inevitable to reassign a *constant fraction* of all variables. This is mirrored in the geometry of the set $\mathcal{S}(\boldsymbol{\Phi}_t)$: it decomposes into exponentially many exponentially tiny subsets, which are mutually separated by a linear Hamming distance $\Omega(n)$. Yet as in the symmetric phase, the marginals of a good fraction of the free variables remain bounded away from 0/1, and the set $\mathcal{S}(\boldsymbol{\Phi}_t)$ remains 'well spread' over the Hamming cube $\{0,1\}^{V_t}$.

The ferromagnetic phase. Let $\alpha > 0$. We say that a set $S \subset \{0,1\}^{\theta n}$ is α-*ferromagnetic* if for any $\sigma, \tau \in S$ we have $\mathrm{dist}(\sigma, \tau) \leq \alpha n$.

Theorem 4. *There are constants $k_0, \rho_0 > 0$ such that for $k \geq k_0$, $\rho_0 \leq \rho \leq k \ln 2 - 2 \ln k$, and*

$$\ln \rho < k \cdot \theta < (1 - \rho^{-2}) \cdot \rho / (\ln 2) \tag{2}$$

the random formula $\boldsymbol{\Phi}_t$ has the following properties w.h.p.

1. *In almost all $\sigma \in \mathcal{S}(\boldsymbol{\Phi}_t)$ at least $0.99\theta n$ variables are $\Omega(n)$-rigid.*
2. *The set $\mathcal{S}(\boldsymbol{\Phi}_t)$ is $\exp(2-\rho)/k$-ferromagnetic.*
3. *At least $0.99\theta n$ variables $x \in V_t$ satisfy $M_x(\boldsymbol{\Phi}_t) \in \left[0, 2^{-k/2}\right] \cup \left[1 - 2^{-k/2}, 1\right]$.*
4. *There is a set $R \subset V_t$ of size $|R| \geq 0.99\theta n$ such that for any $\sigma, \tau \in \mathcal{S}(\boldsymbol{\Phi}_t)$ we have $|\{x \in R : \sigma(x) \neq \tau(x)\}| \leq k 2^{-k} n$.*

In other words, as the decimation process progresses to a point that the fraction θ of free variables satisfies (2), the set of satisfying assignments shrinks into a ferromagnetic subset of $\{0,1\}^{V_t}$ of tiny diameter, in contrast to a well-spread shattered set as in Theorem 3. Furthermore, most marginals $M_x(\boldsymbol{\Phi}_t)$ are either extremely close to 0 or extremely close to 1. In fact, there is a large set R of variables on which all satisfying assignments virtually agree (more precisely: any two can't disagree on more than $k2^{-k}n$ variables in R).

The forced phase. We call a variable x *forced* in the formula Φ_t if Φ_t has a clause that only contains the variable x (a 'unit clause'). Clearly, in any satisfying assignment x must be assigned so as to satisfy this clause.

Theorem 5. *There are constants $k_0, \rho_0 > 0$ such that for $k \geq k_0$, $\rho_0 \leq \rho \leq k \ln 2 - 2 \ln k$, and*

$$1/n \ll k \cdot \theta < \ln(\rho)(1 - 10/\ln \rho) \tag{3}$$

the random formula $\boldsymbol{\Phi}_t$ has the following properties w.h.p.

1. *At least $0.99\theta n$ variables are forced.*
2. *The set $\mathcal{S}(\boldsymbol{\Phi}_t)$ is $\exp(2-\rho)/k$-ferromagnetic.*

Belief Propagation. As mentioned earlier, the BP decimation algorithm is an attempt at implementing the decimation process by means of an efficient algorithm. The key issue with this is the computation of the marginals $M_{x_t}(\Phi_{t-1})$ in step 2 of the decimation process. Indeed, the problem of computing these marginals is $\#P$-hard in the worst case. Thus, instead of working with the 'true' marginals, BP decimation uses certain numbers $\mu_{x_t}(\Phi_{t-1}, \omega)$ that can be computed efficiently, where $\omega \geq 1$ is an integer parameter. The precise definition of the $\mu_{x_t}(\Phi_{t-1}, \omega)$ can be found in [6]. Basically, they are the result of a 'local' dynamic programming algorithm ('Belief Propagation') that depends upon the assumption of a certain correlation decay property. For given k, ρ, the key hypothesis underpinning the BP decimation algorithm is

Hypothesis 6. *For any $\varepsilon > 0$ there is $\omega = \omega(\varepsilon, k, \rho, n) \geq 1$ such that w.h.p. for all $1 \leq t \leq n$ we have $|\mu_{x_t}(\boldsymbol{\Phi}_{t-1}, \omega) - M_{x_t}(\boldsymbol{\Phi}_{t-1})| < \varepsilon$.*

In other words, Hypothesis 6 states that throughout the decimation process, the 'BP marginals' $\mu_{x_t}(\Phi_{t-1}, \omega)$ are a good approximation to the true marginals $M_{x_t}(\Phi_{t-1})$.

Theorem 7. *There exist constants $c_0, k_0, \rho_0 > 0$ such that for all $k \geq k_0$, and $\rho_0 \leq \rho \leq k \ln 2 - 2 \ln k$ the following is true for any integer $\omega = \omega(k, \rho, n) \geq 1$. Suppose that*

$$c_0 \ln(\rho) < k \cdot \theta < \rho / \ln 2. \tag{4}$$

Then for at least $0.99\theta n$ variables $x \in V_t$ we have $\mu_x(\boldsymbol{\Phi}_t, \omega) \in [0.49, 0.51]$.

The proof is based on the techniques developed in [7]; the details are omitted from this extended abstract. Comparing Theorem 4 with Theorem 7, we see that w.h.p. for θ satisfying (4) most of the 'true' marginals $M_x(\boldsymbol{\Phi}_t)$ are very close to either 0 or 1, whereas the 'BP marginals' lie in $[0.49, 0.51]$. Thus, in the regime described by (4) the BP marginals do *not* provide a good approximation to the actual marginals.

Corollary 1. *There exist constants $c_0, k_0, \rho_0 > 0$ such that for all $k \geq k_0$, $\rho_0 \leq \rho \leq k \ln 2 - 3 \ln k$ Hypothesis 6 is untrue.*

Summary and discussion. Fix $k \geq k_0$ and $\rho \geq \rho_0$. Theorems 2–5 show how the space of satisfying assignments of $\boldsymbol{\Phi}_t$ evolves as the decimation process progresses. In the *symmetric phase* $k\theta \geq \exp((1 + o_\rho(1))\rho)$ where there still is a large number of free variables, the correlations amongst the free variables are purely local ('loose variables'). As the number of free variables enters the regime $(1 + o_\rho(1))\rho / \ln 2 \leq k\theta \leq \exp((1 - o_\rho(1))\rho)$, the set $\mathcal{S}(\boldsymbol{\Phi}_t)$ of satisfying assignments shatters into exponentially many tiny 'clusters', each of which comprises only an exponentially small fraction of all satisfying assignments. Most satisfying assignments exhibit long-range correlations amongst the possible values that can be assigned to the individual variables ('rigid variables'). This phenomenon goes by the name of *dynamic replica symmetry breaking* in statistical mechanics [11].

While in the previous phases the set of satisfying assignments is scattered all over the Hamming cube (as witnessed by the average Hamming distance of two satisfying assignments), in the *ferromagnetic phase* $(1 - o_\rho(1)) \ln \rho \leq k\theta \leq (1 - o_\rho(1))\rho / \ln 2$ the set of satisfying assignments has a tiny diameter. This is mirrored by the fact that the marginals of most variables are extremely close to either 0 or 1. Furthermore, in (most of) this phase the estimates of the marginals resulting from Belief Propagation are off (Theorem 7). As part 4 of Theorem 4 shows, the mistaken estimates of the Belief Propagation computation would make it impossible for BP decimation to penetrate the ferromagnetic phase. More precisely, even if BP decimation would emulate the decimation process perfectly up until the ferromagnetic phase commences, with probability $1 - \exp(-\Omega(n))$ BP decimation would then assign at least $k2^{-k}n$ variables in the set R from part 4 of Theorem 4 'wrongly' (i.e., differently than they are assigned in any satisfying assignment). In effect, BP decimation would fail to find a satisfying assignment, regardless of its subsequent decisions. Finally, in the forced phase

$k\theta \leq (1 - o_\rho(1)) \ln \rho$ there is an abundance of unit clauses that make it easy to read off the values of most variables. However, getting stuck in the ferromagnetic phase, BP decimation won't reach this regime.

3 Related Work

BP/SP decimation are inspired by a generic but highly non-rigorous analysis technique from statistical mechanics called the *cavity method* [6]. This technique is primarily destined for the *analysis* of phase transitions. In [6,11] the cavity method was used to study the structure of the set $\mathcal{S}(\boldsymbol{\Phi})$ of satisfying assignments (or, more accurately, properties of the Gibbs measure) of the *undecimated* random formula $\boldsymbol{\Phi}$. Thus, the results obtained in that (non-rigorous) work identify phase transitions solely in terms of the formula density ρ. On the basis of these results, it was hypothesized that (certain versions of) BP decimation should find satisfying assignments up to $\rho \sim \ln k$ or even up to $\rho \sim k \ln 2$ [11]. The argument given for the latter scenario in [11] is that the key obstacle for BP to approximate the true marginals is *condensation*, a phenomenon that from the viewpoint of BP is very similar to ferromagnetism. In terms of the parameter ρ, the condensation threshold was (non-rigorously) estimated to occur at $\rho = k \ln 2 - 3k2^{-k-1} \ln 2$. However, [7] shows that (the basic version of) BP decimation fails to find satisfying assignments already for $\rho \geq \rho_0$, with ρ_0 a constant independent of k.

The explanation for this discrepancy is that [6,11] neglect the time parameter $\theta = 1 - t/n$ of the decimation process. As Theorem 4 shows, even for *fixed* $\rho \geq \rho_0$ (independent of k) ferromagnetism occurs as the decimation process proceeds to θ in the regime (2). This means that decimating variables has a similar effect on the geometry of the set of satisfying assignments as increasing the clause/variable density. On a non-rigorous basis an analysis both in terms of the formula density ρ and the time parameter θ was carried out in [16]. Thus, our results can be viewed as a rigorous version of parts of [16] (with proofs based on completely different techniques). In addition, Theorem 7 confirms rigorously that for ρ, θ in the ferromagnetic phase, BP does not yield the correct marginals.

The present results have no immediate bearing on the conceptually more sophisticated SP decimation algorithm. However, we conjecture that SP undergoes a similar sequence of phase transitions and that the algorithm will not find satisfying assignments for densities $\rho \geq \rho_0$, with ρ_0 a constant independent of k.

Theorem 3 can be viewed as a generalization of the results on random k-SAT obtained in [1] (which additionally deals with further problems such as random graph/hypergraph coloring). In [1] we rigorously proved a substantial part of the results hypothesized in [11] on shattering and rigidity in terms of the clause/variable density ρ; this improved prior work [2,5,8]. The new aspect of the present work is that we identify not only a transition for shattering/rigidity, but also for ferromagnetism and forcing in terms of *both* the density ρ and the time parameter θ of the decimation process. As explained in the previous paragraph, the time parameter is crucial to link these phase transitions to the performance of algorithms such as BP decimation.

In particular, from Theorem 3 we can recover the main result of [1] on random k-SAT. Namely, if $\rho \geq \ln k + 2\ln\ln k + 2$, then (11) is satisfied even for $\theta = 1$, i.e., the *undecimated* random formula $\boldsymbol{\Phi}$ has the properties 1.–4. stated in Theorem 3. Technically, the present paper builds upon the methods developed in [1].

4 Analyzing the Decimation Process

In the rest of the paper, we are going to sketch the proofs of the main results. In this section we perform some groundwork to facilitate a rigorous analysis of the decimation process. The key problem is to get a handle on the following experiment:

D1. Generate a random formula $\boldsymbol{\Phi}$, conditioned on $\boldsymbol{\Phi}$ being satisfiable.
D2. Run the decimation process for t steps to obtain $\boldsymbol{\Phi}_t$.
D3. Choose a satisfying assignment $\boldsymbol{\sigma}_t \in \mathcal{S}(\boldsymbol{\Phi}_t)$ uniformly at random.
D4. The result is the pair $(\boldsymbol{\Phi}_t, \boldsymbol{\sigma}_t)$.

As throughout the paper we only work with densities m/n where $\boldsymbol{\Phi}$ is satisfiable w.h.p., the conditioning in step **D1** is essentially void. Recalling that the outcome of the decimation process is a uniformly random satisfying assignment of $\boldsymbol{\Phi}$, we see that the following experiment is equivalent to **D1**–**D4**:

U1. Generate a random formula $\boldsymbol{\Phi}$, conditioned on $\boldsymbol{\Phi}$ being satisfiable.
U2. Choose $\boldsymbol{\sigma} \in \mathcal{S}(\boldsymbol{\Phi})$ uniformly at random.
U3. Substitute $\boldsymbol{\sigma}(x_i)$ for x_i for $1 \leq i \leq t$ and simplify to obtain a formula $\boldsymbol{\Phi}_t$.
U4. The result is the pair $(\boldsymbol{\Phi}_t, \boldsymbol{\sigma}_t)$, where $\boldsymbol{\sigma}_t : V_t \to \{0,1\},\ x \mapsto \boldsymbol{\sigma}(x)$.

Fact 8. *The two probability distributions induced on formula/assignment pairs by the two experiments* **D1**–**D4** *and* **U1**–**U4** *are identical.*

Still, an analysis of **U1**–**U4** seems difficult because of **U2**: it is unclear how to analyze (or implement) this step directly. Following [1], we will surmount this problem by considering yet another experiment.

P1. Choose an assignment $\boldsymbol{\sigma}' \in \{0,1\}^V$ uniformly at random.
P2. Choose a formula $\boldsymbol{\Phi}'$ with m clauses that is satisfied by $\boldsymbol{\sigma}'$ uniformly at random.
P3. Substitute $\boldsymbol{\sigma}'(x_i)$ for x_i for $1 \leq i \leq t$ and simplify to obtain a formula $\boldsymbol{\Phi}'_t$.
P4. The result is the pair $(\boldsymbol{\Phi}'_t, \boldsymbol{\sigma}'_t)$, where $\boldsymbol{\sigma}'_t : V_t \to \{0,1\},\ x \mapsto \boldsymbol{\sigma}'(x)$.

The experiment **P1**–**P4** is easy to implement and, in effect, also amenable to a rigorous analysis. For given the assignment $\boldsymbol{\sigma}'$, there are $(2^k - 1)\binom{n}{k}$ clauses in total that evaluate to 'true' under $\boldsymbol{\sigma}'$, and to generate $\boldsymbol{\Phi}'$ we merely choose m out of these uniformly and independently. Unfortunately, it is *not* true that the experiment **P1**–**P4** is equivalent to **U1**–**U4**. However, we will employ a result from [1] that establishes a connection between these two experiments that is strong enough to extend many results from **P1**–**P4** to **U1**–**U4**.

To state this result, observe that **P1**–**P4** and **U1**–**U4** essentially only differ in their first two steps. Thus, let $\Lambda_k(n,m)$ denote the set of all pairs (Φ,σ), where Φ is a k-CNF on $V=\{x_1,\ldots,x_n\}$ with m clauses, and $\sigma\in\mathcal{S}(\Phi)$. Let $\mathcal{U}_k(n,m)$ denote the probability distribution induced on $\Lambda_k(n,m)$ by **U1**–**U2**, and let $\mathcal{P}_k(n,m)$ signify the distribution induced by **P1**–**P2**; this distribution is sometimes called the *planted model*.

Theorem 9 ([1]). *Suppose $k\geq 4$ and $0<\rho<k\ln 2-k^2/2^k$. Let $\mathcal{E}\subset\Lambda_k(n,m)$. If $\mathrm{P}_{\mathcal{P}_k(n,m)}[\mathcal{E}]\geq 1-\exp(-\rho n/2^k)$ then $\mathrm{P}_{\mathcal{U}_k(n,m)}[\mathcal{E}]=1-o(1)$.*

5 Shattering, Pairwise Distances, and Ferromagnetism

To prove shattering and ferromagnetism, we adapt arguments from [1,2,8] to the situation where we have the *two* parameters θ,ρ (rather than just ρ). Let $(\boldsymbol{\Phi}_t,\boldsymbol{\sigma}_t)$ be the (random) outcome of the experiment **U1**–**U4**. For $0\leq\alpha\leq 1$ let $X_\alpha(\boldsymbol{\Phi}_t,\boldsymbol{\sigma}_t)$ denote the number of satisfying assignments $\tau\in\mathcal{S}(\boldsymbol{\Phi}_t)$ with Hamming distance $d(\boldsymbol{\sigma}_t,\tau)=\alpha\theta n$. To establish the 'shattering' part of Theorem 3, we are going to prove the following

Claim 10. *Under the assumptions of Theorem 3 there exist $a_1<a_2<0.49$, $a_3>0$ depending only on k,ρ such that w.h.p. we have*

$$X_\alpha(\boldsymbol{\Phi}_t,\boldsymbol{\sigma}_t)=0 \quad \text{for all } a_1<\alpha<a_2, \text{ and} \tag{5}$$

$$\max_{\alpha\leq 0.49} X_\alpha(\boldsymbol{\Phi}_t,\boldsymbol{\sigma}_t)<\exp(-a_3 n)\cdot|\mathcal{S}(\boldsymbol{\Phi}_t)|\,. \tag{6}$$

Claim 10 implies that for the outcome $\boldsymbol{\Phi}_t$ of the first t steps of the decimation process the set $\mathcal{S}(\boldsymbol{\Phi}_t)$ shatters w.h.p. For by Fact 8 Claim 10 implies that w.h.p. almost all $\sigma_t\in\mathcal{S}(\boldsymbol{\Phi}_t)$ are such that (5) and (6) hold. Choose any such $\sigma_{t,1}\in\mathcal{S}(\boldsymbol{\Phi}_t)$ and let $R_1=\{\tau\in\mathcal{S}(\boldsymbol{\Phi}_t):d(\tau,\sigma_{t,1})\leq a_1 n\}$. Then, choose $\sigma_{t,2}\in\mathcal{S}(\boldsymbol{\Phi}_t)\setminus R_1$ satisfying (5) and (6), let $R_2=\{\tau\in\mathcal{S}(\boldsymbol{\Phi}_t)\setminus R_1:d(\tau,\sigma_{t,2})\leq a_1 n\}$, and proceed inductively until all remaining satisfying assignments violate either (5) or (6). Let $R_1,\ldots,R_N$ be the classes constructed in this way and let $R_0=\mathcal{S}(\boldsymbol{\Phi}_t)\setminus\bigcup_{i=1}^N R_i$. An additional (simple) argument is needed to show that $|R_0|\leq\exp(-\Omega(n))|\mathcal{S}(\boldsymbol{\Phi}_t)|$ w.h.p. The decomposition $R_0,\ldots,R_N$ witnesses that $\mathcal{S}(\boldsymbol{\Phi}_t)$ shatters.

With respect to pairwise distances of satisfying assignments, (6) implies that w.h.p. only an exponentially small fraction of all satisfying assignments of $\boldsymbol{\Phi}_t$ lies within distance $\leq 0.49\theta n$ of $\boldsymbol{\sigma}_t$. It is not difficult to derive the statement made in Theorem 3 on the average pairwise distance from this. In addition, the fact that the average pairwise distance of satisfying assignments is $\geq 0.49\theta n$ w.h.p. implies in combination with a double counting argument the claim about the marginals $M_x(\boldsymbol{\Phi}_t)$ in Theorems 2 and 3.

To establish Claim 10 we will work with the experiment **P1**–**P4** and use Theorem 9 to transfer the result to the experiment **U1**–**U4**. Thus, let $(\boldsymbol{\Phi}'_t, \boldsymbol{\sigma}'_t)$ be the (random) outcome of experiment **P1**–**P4**, and assume that k, ρ, θ are as in Theorem 3. To prove (5) we need to bound $X_\alpha(\boldsymbol{\Phi}'_t, \boldsymbol{\sigma}'_t)$ from above, for which we use the 'first moment method'. Indeed, by standard arguments (similar to those used in [2]) the expectation of $X_\alpha(\boldsymbol{\Phi}'_t, \boldsymbol{\sigma}'_t)$ satisfies $\frac{1}{n}\ln \mathrm{E}X_\alpha(\boldsymbol{\Phi}'_t, \boldsymbol{\sigma}'_t) \leq \psi(\alpha)$, with

$$\psi(\alpha) = -\alpha\theta\ln\alpha - (1-\alpha)\theta\ln(1-\alpha) + \frac{2^k\rho}{k}\ln\left(1 - \frac{1-(1-\alpha\theta)^k}{2^k-1}\right).$$

Thus, in order to prove that $\max_{a_1<\alpha<a_2} X_\alpha(\boldsymbol{\Phi}'_t, \boldsymbol{\sigma}'_t) = 0$ w.h.p. we would just have to prove that $\max_{a_1<\alpha<a_2}\psi(\alpha) < 0$ (so that Markov's inequality implies that $X_\alpha = 0$ w.h.p.). But as our goal is to prove a result about the $X_\alpha(\boldsymbol{\Phi}_t, \boldsymbol{\sigma}_t)$ (i.e., the experiment **U1**–**U4**), we need to prove a slightly stronger bound, namely $\max_{a_1<\alpha<a_2}\psi(\alpha) < -\rho/2^k$. Then Markov's inequality and Theorem 9 imply the first part of Claim 10. Via elementary calculus, one can show that the aforementioned bound holds with $a_1 = \exp(2-\rho) - \varepsilon$ and $a_2 = \exp(2-\rho) + \varepsilon$ for a sufficiently small $\varepsilon > 0$.

To prove (6) we bound $\mathrm{E}X_\alpha$ from above by a similar first moment argument. But in addition, we need a lower bound on $|\mathcal{S}(\boldsymbol{\Phi}_t)|$. To derive this, we need

Theorem 11 ([2]). *Assume $k \geq 4$ and $\rho \leq k\ln 2 - k^2/2^k$. Then w.h.p. $\frac{1}{n}\ln|\mathcal{S}(\boldsymbol{\Phi})| \geq \ln 2 + 2^k\frac{\rho}{k}\ln(1-2^{-k}) - 0.99\rho/2^k$.*

Together with a double counting argument, Theorem 11 implies the part 2 of Claim 10. The 'ferromagnetism' bit of Theorem 4 follows from similar arguments.

6 Rigid Variables

Assume that k, ρ, θ satisfy the assumptions of Theorem 3. Let $(\boldsymbol{\Phi}_t, \boldsymbol{\sigma}_t)$ be the (random) outcome of **U1**–**U4**. Our goal is to show that w.h.p. most variables $x \in V_t$ are rigid.

What is the basic obstacle that makes it difficult to 'flip' the value of x? Observe that we can simply assign x the opposite value $1 - \boldsymbol{\sigma}_t(x)$, unless $\boldsymbol{\Phi}_t$ has a clause $\mathcal{C}$ in which either x or $\bar{x}$ is the *only* literal that is true under $\boldsymbol{\sigma}_t$. If there is such a clause, we say that x *supports* $\mathcal{C}$. But even if x supports a clause $\mathcal{C}$ it might be easy to flip. For instance, if $\mathcal{C}$ features some variable $y \neq x$ that does not support a clause, then we could just flip both x, y simultaneously. Thus, to establish the existence of $\Omega(n)$-rigid variables we need to analyze the distribution of the number of clauses that a variable supports, the probability that these clauses only consists of variables that support further clauses, the probability that the same is true of those clauses, etc.

This analysis can be performed fairly neatly for the outcome $(\boldsymbol{\Phi}'_t, \boldsymbol{\sigma}'_t)$ of the experiment **P1**–**P4**. Let us sketch how this works, and why rigidity occurs at $k\theta = \exp((1+o(1))\rho)$ (cf. (1)). For a variable $x \in V_t$ we let S_x be the number of

clauses supported by x. Given the assignment $\boldsymbol{\sigma}'$ chosen in step **P1**, there are a total of $\binom{n-1}{k-1}$ possible clauses that x supports. Since in step **P2** we include m out of the $(2^k-1)\binom{n}{k}$ possible clauses satisfied under $\boldsymbol{\sigma}'$ uniformly and independently, we get $\mathrm{E}[S_x] = m\binom{n-1}{k-1}/\left((2^k-1)\binom{n}{k}\right) = \rho/(1-2^{-k}) \geq \rho$. In fact, S_x is binomially distributed. Hence, $\mathrm{P}[S_x = 0] \leq \exp(-\rho)$. Thus, the *expected* number of variables $x \in V_t$ with $S_x = 0$ is $\leq \theta n \exp(-\rho)$. Furthermore, if we condition on $S_x = j \geq 1$, then the actual *clauses* $\mathcal{C}_1, \ldots, \mathcal{C}_j$ supported by x are just independently uniformly distributed over the set of all $\binom{n-1}{k-1}$ possible clauses that x supports. Therefore, the *expected* number of variables $y \in V_t$ with $S_y = 0$ occurring in one of these clauses $\mathcal{C}_i$ is $(1+o(1))(k-1)\cdot\theta\exp(-\rho) \leq k\theta\exp(-\rho)$. Hence, if θ is as in (11), then this number is $\leq \exp(-2)/\rho$, i.e., 'small' for $\rho \geq \rho_0$ sufficiently big. Thus, we would expect that *most* clauses supported by x indeed consist exclusively of variables that support other clauses. Hence, for θ as in (11) we can expect most variables to be rigid.

Let us now indicate how this argument can be carried out in detail. Analyzing the distribution of the variables S_x in the experiment **P1**–**P4** and extending the result to the experiment **U1**–**U4** via Theorem 9, and setting $\zeta = \rho^2/\exp(\rho)$, we obtain the following.

Proposition 1. *Suppose that k, ρ, θ satisfy the assumptions of Theorem 3. Then w.h.p. in a random pair $(\boldsymbol{\Phi}_t, \boldsymbol{\sigma}_t)$ generated by the experiment* **U1**–**U4** *no more than $2\zeta\theta n$ variables in V_t support fewer than three clauses,*

To establish rigidity, we need to show that most variables support clauses in which only variables occur that support other clauses. To express this, we say that $S \subset V_t$ is *t-self-contained* if each $x \in S$ supports at least two clauses of $\boldsymbol{\Phi}_t$ that contain variables from S only. From Proposition 1 we can derive

Proposition 2. *Suppose that k, ρ, θ satisfy the assumptions of Theorem 3. The outcome $(\boldsymbol{\Phi}_t, \boldsymbol{\sigma}_t)$ of* **U1**–**U4** *has a t-self-contained set of size $(1-3\zeta)\theta n$ w.h.p.*

Suppose that $(\boldsymbol{\Phi}_t, \boldsymbol{\sigma}_t)$ has a self-contained set S of size $(1-3\zeta)\theta n$. To flip the value of a variable $x \in S$ we need to also flip one other variable from each of the (at least two) clauses that x supports and that consist of variables from S only. As each of these two variables, in turn, supports at least two clauses comprised of variables from S only, we need to also flip further variables in those. But these variables are again contained in S. This suggests that attempting to flip x will entail an avalanche of further flips. Indeed, the expansion properties of the random formula $\boldsymbol{\Phi}_t$ imply the following.

Proposition 3. *With k, ρ, θ as in Theorem 3 there is $\chi = \chi(k, \rho) > 0$ such that the outcome $(\boldsymbol{\Phi}_t, \boldsymbol{\sigma}_t)$ of* **U1**–**U4** *has the following property w.h.p.: all variables that are contained in a t-self-contained set are χn-rigid.*

Propositions 2 and 3 directly imply part 1 of Theorem 3. Self-contained sets also play a key role in the proof of Theorem 4 (details omitted).

References

1. Achlioptas, D., Coja-Oghlan, A.: Algorithmic barriers from phase transitions. In: Proc. 49th FOCS, pp. 793–802 (2008)
2. Achlioptas, D., Coja-Oghlan, A., Ricci-Tersenghi, F.: On the solution space geometry of random formulas. Random Structures and Algorithms 38, 251–268 (2011)
3. Achlioptas, D., Moore, C.: Random k-SAT: two moments suffice to cross a sharp threshold. SIAM Journal on Computing 36, 740–762 (2006)
4. Achlioptas, D., Peres, Y.: The threshold for random k-SAT is $2^k \ln 2 - O(k)$. Journal of the AMS 17, 947–973 (2004)
5. Achlioptas, D., Ricci-Tersenghi, F.: Random formulas have frozen variables. SIAM J. Comput. 39, 260–280 (2009)
6. Braunstein, A., Mézard, M., Zecchina, R.: Survey propagation: an algorithm for satisfiability. Random Structures and Algorithms 27, 201–226 (2005)
7. Coja-Oghlan, A.: On belief propagation guided decimation for random k-SAT. In: Proc. 22nd SODA, pp. 957–966 (2011)
8. Daudé, H., Mézard, M., Mora, T., Zecchina, R.: Pairs of SAT-assignments in random Boolean formulae. Theoretical Computer Science 393, 260–279 (2008)
9. Frieze, A., Suen, S.: Analysis of two simple heuristics on a random instance of k-SAT. Journal of Algorithms 20, 312–355 (1996)
10. Kroc, L., Sabharwal, A., Selman, B.: Message-passing and local heuristics as decimation strategies for satisfiability. In: Proc. 24th SAC, pp. 1408–1414 (2009)
11. Krzakala, F., Montanari, A., Ricci-Tersenghi, F., Semerjian, G., Zdeborova, L.: Gibbs states and the set of solutions of random constraint satisfaction problems. Proc. National Academy of Sciences 104, 10318–10323 (2007)
12. Mertens, S., Mézard, M., Zecchina, R.: Threshold values of random K-SAT from the cavity method. Random Struct. Alg. 28, 340–373 (2006)
13. Mézard, M., Parisi, G., Zecchina, R.: Analytic and algorithmic solution of random satisfiability problems. Science 297, 812–815 (2002)
14. Mitchell, D., Selman, B., Levesque, H.: Hard and easy distribution of SAT problems. In: Proc. 10th AAAI, pp. 459–465 (1992)
15. Montanari, A., Ricci-Tersenghi, F., Semerjian, G.: Solving constraint satisfaction problems through Belief Propagation-guided decimation. In: Proc. 45th Allerton (2007)
16. Ricci-Tersenghi, F., Semerjian, G.: On the cavity method for decimated random constraint satisfaction problems and the analysis of belief propagation guided decimation algorithms. J. Stat. Mech., 09001 (2009)

Improved Bounds for the Randomized Decision Tree Complexity of Recursive Majority*

Frédéric Magniez[1], Ashwin Nayak[2,**], Miklos Santha[1,3,***], and David Xiao[1,4]

[1] LIAFA, Univ. Paris 7, CNRS; Paris, France
magniez@liafa.jussieu.fr
[2] C&O and IQC, U. Waterloo; and Perimeter Institute; Waterloo, ON, Canada
ashwin.nayak@uwaterloo.ca
[3] Centre for Quantum Technologies, National U. of Singapore
santha@lri.fr
[4] Univ. Paris-Sud; Orsay, France
dxiao@lri.fr

Abstract. We consider the randomized decision tree complexity of the recursive 3-majority function. For evaluating height h formulae, we prove a lower bound for the δ-two-sided-error randomized decision tree complexity of $(1-2\delta)(5/2)^h$, improving the lower bound of $(1-2\delta)(7/3)^h$ given by Jayram, Kumar, and Sivakumar (STOC '03). Second, we improve the upper bound by giving a new zero-error randomized decision tree algorithm that has complexity at most $(1.007)\cdot 2.64946^h$. The previous best known algorithm achieved complexity $(1.004)\cdot 2.65622^h$. The new lower bound follows from a better analysis of the base case of the recursion of Jayram *et al.* The new algorithm uses a novel "interleaving" of two recursive algorithms.

1 Introduction

Decision trees form a simple model for computing boolean functions by successively reading the input bits until the value of the function can be determined. In this model, the only cost is the number of input bits queried. Formally, a *deterministic decision tree algorithm* A on n variables is a binary tree in which each internal node is labeled with an input variable x_i, and the leaves of the tree are labeled by either 0 or 1. Each internal node has two outgoing edges, labeled either by 0 or 1. For every input $x = x_1 \dots x_n$, there is a unique path in the tree leading from the root to a leaf: if an internal node is labeled by x_i, we follow either the 0 or the 1 outgoing edge according to the value of x_i. The value of the algorithm A on input x, denoted by $A(x)$, is the label of the leaf on this unique

* Partially supported by the French ANR Defis project ANR-08-EMER-012 (QRAC) and the European Commission IST STREP project 25596 (QSC).
** Partially supported by NSERC Canada. Research at PI is supported by the Gvt of Canada through Industry Canada and by the Province of Ontario through MRI.
*** Research at the Centre for Quantum Technologies is funded by the Singapore Ministry of Education and the National Research Foundation.

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 317–329, 2011.

path. The algorithm A *computes* a boolean function $f : \{0,1\}^n \to \{0,1\}$ if for every input x, we have $A(x) = f(x)$.

We define the *cost* $\mathrm{C}(A,x)$ of a deterministic decision tree algorithm A on input x as the number of input bits queried by A on x. Let $\mathcal{P}_f$ be the set of all deterministic decision tree algorithms which compute f. The *deterministic complexity* of f is $\mathrm{D}(f) = \min_{A\in\mathcal{P}_f} \max_{x\in\{0,1\}^n} \mathrm{C}(A,x)$. Since every function can be evaluated after reading all the input variables, $\mathrm{D}(f) \le n$. In an extension of the deterministic model, we can also permit randomization in the computation.

A *randomized decision tree algorithm* A on n variables is a distribution over all deterministic decision tree algorithms on n variables. Given an input x, the algorithm first samples a deterministic tree $B \in_{\mathrm{R}} A$, then evaluates $B(x)$. The error probability of A in computing f is given by $\max_{x\in\{0,1\}^n} \Pr_{B\in_{\mathrm{R}}A}[B(x) \neq f(x)]$. The *cost* of a randomized algorithm A on input x, denoted also by $\mathrm{C}(A,x)$, is the expected number of input bits queried by A on x. Let $\mathcal{P}_f^\delta$ be the set of randomized decision tree algorithms computing f with error at most δ. The two-sided bounded error *randomized complexity* of f with error $\delta \in [0,1/2)$ is $\mathrm{R}_\delta(f) = \min_{A\in\mathcal{P}_f^\delta} \max_{x\in\{0,1\}^n} \mathrm{C}(A,x)$.

We write $\mathrm{R}(f)$ for $\mathrm{R}_0(f)$. By definition, for all $0 \le \delta < 1/2$, it holds that $\mathrm{R}_\delta(f) \le \mathrm{R}(f) \le \mathrm{D}(f)$, and it is also known [1, 2, 12] that $\mathrm{D}(f) \le \mathrm{R}(f)^2$, and that for all constant $\delta \in (0,1/2)$, $\mathrm{D}(f) \in \mathrm{O}(\mathrm{R}_\delta(f)^3)$ [7].

Considerable attention in the literature has been given to the randomized complexity of functions computable by read-once formulae, which are boolean formulae in which every input variable appears only once. For a large class of well balanced formulae with NAND gates the exact randomized complexity is known. In particular, let NAND_h denote the *complete* binary tree of height h with NAND gates, where the inputs are at the $n = 2^h$ leaves. Snir [11] has shown that $\mathrm{R}(\mathsf{NAND}_h) \in \mathrm{O}(n^c)$ where $c = \log_2\left(\frac{1+\sqrt{33}}{4}\right) \approx 0.753$. A matching $\Omega(n^c)$ lower bound was obtained by Saks and Wigderson [9], and extended to Monte-Carlo algorithms by Santha [10]. Since $\mathrm{D}(\mathsf{NAND}_h) = 2^h = n$ this implies that $\mathrm{R}(\mathsf{NAND}_h) \in \Theta(\mathrm{D}(\mathsf{NAND}_h)^c)$. Saks and Wigderson conjectured that *for every* boolean function f and constant $\delta \in [0,1/2)$, $\mathrm{R}_\delta(f) \in \Omega(\mathrm{D}(f)^c)$.

After further progress due to Heiman, Newman, and Wigderson [3] and Heiman and Wigderson [4], one would have hoped that the simple model of decision tree algorithms might shed more light on the power of randomness. But surprisingly, we know the exact randomized complexity of very few boolean functions. In particular, the randomized complexity of the recursive 3-majority function ($\mathsf{3\text{-}MAJ}_h$) is still open. This function, proposed by Boppana, was one of the earliest examples where randomized algorithms were found to be more powerful than deterministic decision trees [9]. It is a read-once formula on 3^h variables given by the complete ternary tree of height h whose internal vertices are majority gates. It is easy to check that $\mathrm{D}(\mathsf{3\text{-}MAJ}_h) = 3^h$, but there is a naive randomized recursive algorithm for $\mathsf{3\text{-}MAJ}_h$ that performs better: pick two random children of the root and recursively evaluate them, then evaluate the third child iff the value is not yet determined. This has zero-error randomized complexity $(8/3)^h$. However, it was already observed by Saks and Wigderson [9]

that one can do even better than this naive algorithm. As for lower bounds, that reading 2^h variables is necessary for zero-error algorithms is easy to show. In spite of some similarities with the NAND_h function, no progress was reported on the randomized complexity of 3-MAJ for 17 years. In 2003, Jayram, Kumar, and Sivakumar [5] proposed an explicit randomized algorithm that achieves complexity $(1.004) \cdot 2.65622^h$, and beats the naive recursion. (Note, however, that the recurrence they derive in [5, Appendix B] is incorrect.) They also prove a $(1-2\delta)(7/3)^h$ lower bound for the δ-error randomized decision tree complexity of $\mathsf{3\text{-}MAJ}_h$. In doing so, they introduce a powerful combinatorial technique for proving decision tree lower bounds.

In this paper, we considerably improve the lower bound obtained in [5], by proving that $\mathrm{R}_\delta(\mathsf{3\text{-}MAJ}_h) \geq (1-2\delta)(5/2)^h$. We also improve the upper bound by giving a new zero-error randomized decision tree algorithm that has complexity at most $(1.007)2.64946^h$.

Theorem 1. *For all* $\delta \in [0, 1/2]$, *we have* $(1-2\delta)(5/2)^h \leq \mathrm{R}_\delta(\mathsf{3\text{-}MAJ}_h) \leq (1.007)2.64946^h$.

In contrast to the randomized case, the bounded-error *quantum* query complexity of $\mathsf{3\text{-}MAJ}_h$ is known more precisely; it is in $\Theta(2^h)$ [8].

New lower bound. For the lower bound they give, Jayram *et al.* consider a complexity measure related to the distributional complexity of $\mathsf{3\text{-}MAJ}_h$ with respect to a specific hard distribution (cf. Sect. 2.3). The focus of the proof is a relationship between the complexity of evaluating formulae of height h to that of evaluating formulae of height $h-1$. They derive a sophisticated recurrence relation between these two quantities, that finally implies that $\mathrm{R}_\delta(\mathsf{3\text{-}MAJ}_h) \geq (1-2\delta)(2+q)^h$, where $(1-2\delta)q^h$ is a lower bound on the probability p_h^δ that a randomized algorithm with error at most δ queries a special variable, called the "absolute minority", on inputs drawn from the hard distribution. They observe that any randomized decision tree with error at most δ must query at least one variable with probability $1-2\delta$. This variable has probability 3^{-h} of being the absolute minority, so $q \geq 1/3$, and the above lower bound follows.

We obtain the new lower bound by proving that $p_h^\delta \geq (1-2\delta)2^{-h}$, i.e., $q \geq 1/2$, which immediately implies the improved lower bound for $\mathrm{R}_\delta(\mathsf{3\text{-}MAJ}_h)$. We examine the relationship between p_h^δ and p_{h-1}^δ, by encoding a height $h-1$ instance into a height h instance, and using an algorithm for the latter. Analyzing our encoding requires understanding the behavior of all decision trees on 3 variables, and this can be done by exhaustively considering all such trees.

One can ask whether this is the best possible recurrence, and it may be possible to improve it by, say, encoding height $h-2$ instances into height h instances. Unfortunately we are unable to prove a claim analogous to Claim 3 in the case of such a recurrence, as the number of possible decision trees for 9 variables is too large to check exhaustively by hand. We nevertheless conjecture that such a claim exists for a two-level encoding scheme. More details will be provided in the journal version.

New algorithm. The naive algorithm and the algorithm of Jayram *et al.* are examples of *depth-k* recursive algorithms for $\mathsf{3\text{-}MAJ}_h$, for $k = 1, 2$, respectively. A depth-k recursive algorithm is a collection of subroutines, where each subroutine evaluates a node (possibly using information about other previously evaluated nodes), satisfying the following constraint: when a subroutine evaluates a node v, it is only allowed to call other subroutines to evaluate children of v at depth at most k, but is not allowed to call subroutines or otherwise evaluate children that are deeper than k. (Our notion of depth-1 is identical to the terminology "directional" that appears in the literature. In particular, the naive recursive algorithm is a directional algorithm.)

We present an improved depth-two recursive algorithm. To evaluate the root of the majority formula, we recursively evaluate one grandchild from each of two distinct children of the root. The grandchildren "give an opinion" about the values of their parents. The opinion guides the remaining computation in a natural manner: if the opinion indicates that the children are likely to agree, we evaluate the two children in sequence to confirm the opinion, otherwise we evaluate the third child. If at any point the opinion of the nodes evaluated so far changes, we modify our future computations accordingly. A key innovation is the use of an algorithm optimized to compute the value of a *partially evaluated* formula. In our analysis, we recognize when incorrect opinions are formed, and take advantage of the fact that this happens with smaller probability.

We do not believe that the algorithm we present here is optimal. Indeed, we conjecture that even better algorithms exist that follow the same high level intuition applied for depth-k recursion for $k > 2$. However, it seems new insights are required to analyze the performance of deeper recursions, as the formulas describing their complexity become unmanageable for $k > 2$.

Organization. We prepare the background for our main results Sect. 2. In Sect. 3 we prove the new lower bound for 3-MAJ. The new algorithm for the problem is described and analyzed in Sect. 4.

2 Preliminaries

We write $u \in_{\mathrm{R}} D$ to state that u is sampled from the distribution D. If X is a finite set, we identify X with the uniform distribution over X, and so, for instance, $u \in_{\mathrm{R}} X$ denotes a uniform element of X.

2.1 Distributional Complexity

A variant of the randomized complexity we use is *distributional* complexity. Let $\mathcal{D}_n$ be the set of distributions over $\{0,1\}^n$. The *cost* $\mathrm{C}(A, D)$ of a randomized decision tree algorithm A on n variables with respect to a distribution $D \in \mathcal{D}_n$ is the expected number of bits queried by A when x is sampled from D and over the random coins of A. The *distributional complexity* of a function f on n variables for δ two-sided error is $\Delta_\delta(f) = \max_{D \in \mathcal{D}_n} \min_{A \in \mathcal{P}_f^\delta} \mathrm{C}(A, D)$. The following observation is a well established route to proving lower bounds on worst case complexity.

Proposition 2. $\mathrm{R}_\delta(f) \geq \Delta_\delta(f)$.

2.2 The 3-MAJ$_h$ Function and the Hard Distribution

Let $\mathsf{MAJ}(x)$ denote the boolean majority function of its input bits. The ternary majority function 3-MAJ_h is defined recursively on $n = 3^h$ variables, for every $h \geq 0$. We omit the height h when it is obvious from context. For $h = 0$ it is the identity function. For $h > 0$, let x be an input of length n and let $x^{(1)}, x^{(2)}, x^{(3)}$ be the first, second, and third $n/3$ variables of x. Then

$$\text{3-}\mathsf{MAJ}(x) = \mathsf{MAJ}(\text{3-}\mathsf{MAJ}(x^{(1)}),\ \text{3-}\mathsf{MAJ}(x^{(2)}),\ \text{3-}\mathsf{MAJ}(x^{(3)})).$$

In other terms, 3-MAJ_h is defined by the read-once formula on the complete ternary tree T_h of height h in which every internal node is a majority gate and the leaves are the input variables. For every node v in T_h different from the root, let $P(v)$ denote the parent of v. We say that v and w are siblings if $P(v) = P(w)$. For any node v in T_h, let $Z(v)$ denote the set of variables associated with the leaves in the subtree rooted at v. We say that a node v is at depth d in T_h if the distance between v and the root is d. The root is therefore at depth 0, and the leaves are at depth h.

We now define recursively, for every $h \geq 0$, the set $\mathcal{H}_h$ of *hard inputs* of height h. The hard inputs consist of instances for which at each node v in the ternary tree, one child of v has value different from the value of v. For $b \in \{0,1\}$, let $\mathcal{H}_h^b = \{x \in \mathcal{H}_h : \text{3-}\mathsf{MAJ}_h(x) = b\}$. The *hard distribution* on inputs of height h is defined to be the uniform distribution over $\mathcal{H}_h$.

For an $x \in \mathcal{H}_h$, the *minority path* $M(x)$ is the path, starting at the root, obtained by following the child whose value disagrees with its parent. For $0 \leq d \leq h$, the node of $M(x)$ at depth d is called the depth d minority node, and is denoted by $M(x)_d$. We call the leaf $M(x)_h$ of the minority path the *absolute minority* of x, and denote it by $m(x)$.

2.3 The Jayram-Kumar-Sivakumar Lower Bound

For a deterministic decision tree algorithm B computing 3-MAJ_h, let $L_B(x)$ denote the set of variables queried by B on input x. Recall that $\mathcal{P}^\delta_{\text{3-}\mathsf{MAJ}_h}$ is the set of all randomized decision tree algorithms that compute 3-MAJ_h with two-sided error at most δ. Jayram *et al.* define the function $I^\delta(h,d)$, for $d \leq h$:

$$I^\delta(h,d) = \min_{A \in \mathcal{P}^\delta_{\text{3-}\mathsf{MAJ}_h}} \mathbb{E}_{x \in_{\mathrm{R}} \mathcal{H}_h, B \in_{\mathrm{R}} A}[|Z(M(x)_d) \cap L_B(x)|].$$

In words, it is the minimum over algorithms computing 3-MAJ_h, of the expected number of queries below the dth level minority node, over inputs from the hard distribution. Note that $I^\delta(h,0) = \min_{A \in \mathcal{P}^\delta_{\text{3-}\mathsf{MAJ}_h}} \mathrm{C}(A, \mathcal{H}_h)$, and therefore by Proposition 2, $\mathrm{R}_\delta(\text{3-}\mathsf{MAJ}_h) \geq I^\delta(h,0)$.

We define $p^\delta_h = I^\delta(h,h)$, which is the minimal probability that a δ-error algorithm A queries the absolute minority of a random hard x of height h.

Jayram *et al.* prove a recursive lower bound for $I^\delta(h,d)$ using information theoretic arguments. A more elementary proof can be found in Ref. [6].

Theorem 3 (Jayram, Kumar, Sivakumar [5]). *For all* $0 \le d < h$*:*

$$I^\delta(h,d) \ge I^\delta(h,d+1) + 2I^\delta(h-1,d).$$

A simple computation using their recursion gives $I^\delta(h,0) \ge \sum_{i=0}^{h} \binom{h}{i} 2^{h-i} p_i^\delta$. Putting this together with the fact that $\mathrm{R}_\delta(\mathsf{3\text{-}MAJ}_h) \ge I^\delta(h,0)$, we get the following corollary:

Corollary 4. *Let* $q, a > 0$ *such that* $p_i^\delta \ge a \cdot q^i$ *for all* $i \in \{0,1,2,\ldots,h\}$*. Then* $\mathrm{R}_\delta(\mathsf{3\text{-}MAJ}_h) \ge a(2+q)^h$*.*

As mentioned in Sect. 1, Jayram *et al.* obtain the $(1-2\delta)(7/3)^h$ lower bound from this corollary by observing that $p_h^\delta \ge (1-2\delta)(1/3)^h$.

3 Improved Lower Bound

Theorem 5. *For every error* $\delta > 0$ *and height* $h \ge 0$*, we have* $p_h^\delta \ge (1-2\delta)2^{-h}$*.*

Proof. We prove this theorem by induction. Clearly, $p_0^\delta \ge 1-2\delta$. It then suffices to show that $2p_h^\delta \ge p_{h-1}^\delta$ for $h \ge 1$. We do so by reduction as follows: let A be a randomized algorithm that achieves the minimal probability p_h^δ for height h formulae. We construct a randomized algorithm A' for height $h-1$ formulae such that the probability that A' errs is at most δ, and A' queries the absolute minority with probability at most $2p_h^\delta$. Since p_{h-1}^δ is the minimum probability of querying the absolute minority over all randomized algorithms on inputs of height $h-1$ with error at most δ, this implies that $2p_h^\delta \ge p_{h-1}^\delta$.

We now specify the reduction. For the sake of simplicity, we omit the error δ in the notation. We use the following definition:

Definition 6 (One level encoding scheme). *A* one level encoding scheme *is a bijection* $\psi : \mathcal{H}_{h-1} \times \{1,2,3\}^{3^{h-1}} \to \mathcal{H}_h$*, such that for all* (y,r) *in the domain,* $\mathsf{3\text{-}MAJ}_{h-1}(y) = \mathsf{3\text{-}MAJ}_h(\psi(y,r))$*.*

Let $c : \{0,1\} \times \{1,2,3\} \to \mathcal{H}_1$ *satisfying* $b = \mathsf{MAJ}(c(b,s))$ *for all inputs* (b,s)*. Define the one level encoding scheme* ψ induced *by* c *as follows:* $\psi(y,r) = x \in \mathcal{H}_h$ *such that for all* $1 \le i \le 3^{h-1}$*,* $(x_{3i-2}, x_{3i-1}, x_{3i}) = c(y_i, r_i)$*.*

To define A', we use the one level encoding scheme ψ induced by the following function: $c(y,1) = y01$, $c(y,2) = 1y0$, and $c(y,3) = 01y$.

On input y, algorithm A' picks a uniformly random string $r \in \{1,2,3\}^{3^{h-1}}$, and runs A on $x = \psi(y,r)$. Observe that A' has error at most δ as $\mathsf{3\text{-}MAJ}_{h-1}(y) = \mathsf{3\text{-}MAJ}_h(\psi(y,r))$ for all r, and A has error at most δ. We claim now:

$$2 \Pr_{A,\, x \in_{\mathrm{R}} \mathcal{H}_h}[A(x) \text{ queries } x_{m(x)}] \quad \ge \quad \Pr_{A',\, (y,r) \in_{\mathrm{R}} \mathcal{H}'_h}[A'(y,r) \text{ queries } y_{m(y)}] \qquad (1)$$

where $\mathcal{H}'_h$ is the uniform distribution over $\mathcal{H}_{h-1} \times \{1,2,3\}^{3^{h-1}}$.

We prove this inequality by taking an appropriate partition of the probabilistic space of hard inputs $\mathcal{H}_h$, and prove Eq. 1 separately, on each set in the partition. For $h = 1$, the two classes of the partition are $\mathcal{H}_1^0$ and $\mathcal{H}_1^1$. For $h > 1$, the partition consists of the equivalence classes of the relation $\sim$ defined by $x \sim x'$ if $x_i = x'_i$ for all i such $P(i) \neq P(m(x))$ in the tree T.

Because ψ is a bijection, observe that this also induces a partition of (y, r), where $(y, r) \sim (y', r')$ iff $\psi(y, r) \sim \psi(y', r')$. Also observe that every equivalence class contains three elements. Let S be an equivalence class of $\sim$. Then Eq. 1 follows from the following stronger statement: for every S, and for all B in the support of A, it holds that

$$\begin{aligned} 2 \Pr_{x \in_R \mathcal{H}_h}[B(x) \text{ queries } x_{m(x)} \mid x \in S] \\ \geq \Pr_{(y,r) \in_R \mathcal{H}'_h}[B'(y, r) \text{ queries } y_{m(y)} \mid \psi(y, r) \in S] \ , \end{aligned} \tag{2}$$

where B' is the algorithm that computes $x = \psi(y, r)$ and then evaluates $B(x)$.

The same proof applies to all sets S, but to simplify the notation, we consider a set S that satisfies the following: for $x \in S$, we have $m(x) \in \{1, 2, 3\}$ and that $x_{m(x)} = 1$. Observe that for each $j > 3$, the jth bits of all three elements in S coincide. Therefore, the restriction of B to the variables (x_1, x_2, x_3), when looking only at the three inputs in S, is a well-defined decision tree on three variables. We call this restriction B_1, and formally it is defined as follows: for each query x_j made by B for $j > 3$, B_1 simply uses the value of x_j that is shared by all $x \in S$ and that we hard-wire into B_1; for each query x_j made by B where $j \in \{1, 2, 3\}$, B_1 actually queries x_j. Note that the restriction B_1 does not necessarily compute $\mathsf{3\text{-}MAJ}_1(x_1x_2x_3)$, for two reasons. Firstly, B_1 is derived from B, which may err on particular inputs. But even if $B(x)$ correctly computes $\mathsf{3\text{-}MAJ}_h(x)$, it might happen that B never queries any of x_1, x_2, x_3, or it might query one and never query a second one, etc.

For any $x \in S$, recall that we write $(y, r) = \psi^{-1}(x)$. It holds for our choice of S that $m(y) = 1$ because we assumed $m(x) \in \{1, 2, 3\}$ and also $y_1 = y_{m(y)} = 0$ because we assumed $x_{m(x)} = 1$.

Observe that, for inputs $x \in S$, B queries $x_{m(x)}$ iff B_1 queries the minority among x_1, x_2, x_3. Also, $B'(y, r)$ queries $y_{m(y)}$ iff $B_1(\psi(0, r_1))$ queries x_{r_1} (cf. definition of c used by A'). Furthermore, the distribution of $x_1x_2x_3$ when $x \in_R S$ is uniform over $\mathcal{H}_1^0$. Similarly, the distribution of r_1 over uniform (y, r) conditioned on $\psi(y, r) \in S$ is identical to that of $(0, r_1) = \psi^{-1}(x_1x_2x_3)$ for $x_1x_2x_3 \in_R \mathcal{H}_1^0$. Thus Eq. 2 is equivalent to:

$$\begin{aligned} 2 \Pr_{x \in_R \mathcal{H}_1^0}[B_1(x) \text{ queries } x_{m(x)}] \\ \geq \Pr_{x \in_R \mathcal{H}_1^0}[B_1(x) \text{ queries } x_{r_1} \text{ where } (0, r_1) = \psi^{-1}(x)] \ . \end{aligned} \tag{3}$$

Observe that Eq. 3 holds trivially if B_1 makes no queries, since then both sides equal 0. Therefore it is enough to consider only the case where B_1 makes at least one query. For any decision tree algorithm Q on three bits, which makes

at least one query, we define the number ρ_Q as:

$$\rho_Q = \frac{\Pr_{x\in_R\mathcal{H}_1^0}[Q(x) \text{ queries } x_{m(x)}]}{\Pr_{x\in_R\mathcal{H}_1^0}[Q(x) \text{ queries } x_{r_1} \text{ where } (0,r_1)=\psi^{-1}(x)]} \ .$$

Note that the denominator is at least $1/3$, since Q queries x_{r_1} when x is such that r_1 is the index of the first query. We prove that ρ_Q is always at least $1/2$, by describing a decision tree algorithm Q' which minimizes ρ_Q. The algorithm Q' is defined as follows: first query x_1, if $x_1 = 0$, stop, else if $x_1 = 1$, query x_2 and stop.

Claim. The algorithm Q' gives $\rho_{Q'} = 1/2$, and this is the minimal possible ρ_Q among all deterministic decision tree algorithms making at least one query.

To prove the claim we first evaluate $\rho_{Q'}$. The numerator equals $1/3$ since the minority is queried only when $x = 100$, while the denominator equals $2/3$ since x_{r_1} is queried when x is 001 or 100.

Let now be Q any algorithm which makes at least one query, we prove that $\rho_Q \geq 1/2$. Without loss of generality, we may suppose that the first query is x_1. We distinguish two cases.

If Q makes a second query when the first query is evaluated to 0 then the numerator is at least $2/3$ since for the second query there is also an x for which $m(x)$ is the index of this query. But the denominator is at most 1, and therefore in that case $\rho_Q \geq 2/3$. If Q does not make a second query when the first query is evaluated to 0 then the denominator is at most $2/3$ since for $x = 010$, we have $r_1 = 3$, but x_3 is not queried. Since the numerator is at least $1/3$, we have in that case $\rho_Q \geq 1/2$.

To handle a general S, we replace $\{1, 2, 3\}$ with $m(x)$ and its two siblings. For S such that $x \in S$ satisfies $x_{m(x)} = 0$, the optimal algorithm Q' is the same as the one described above, except that each 0 is changed to 1 and vice versa.

Therefore Eq. 3 holds for every B_1, which implies the theorem. □

Combining Corollary 4 and Theorem 5, we obtain the following.

Corollary 7. $\mathrm{R}_\delta(\text{3-MAJ}_h) \geq (1 - 2\delta)(5/2)^h$.

4 Improved Depth-Two Algorithm

In this section, we present a new zero-error algorithm for computing 3-MAJ_h. For the key ideas behind it, we refer the reader to Sect. 1.

As before, we identify the formula 3-MAJ_h with a complete ternary tree of height h. In the description of the algorithm we adopt the following convention. Once the algorithm has determined the value b of the subformula rooted at a node v of the formula 3-MAJ_h, we also use v to denote this bit value b.

The algorithm is a combination of two depth-2 recursive algorithms. The first one, EVALUATE, takes a node v of height $h(v)$, and evaluates the subformula rooted at v. The interesting case, when $h(v) > 1$, is depicted in Fig. 1. The first

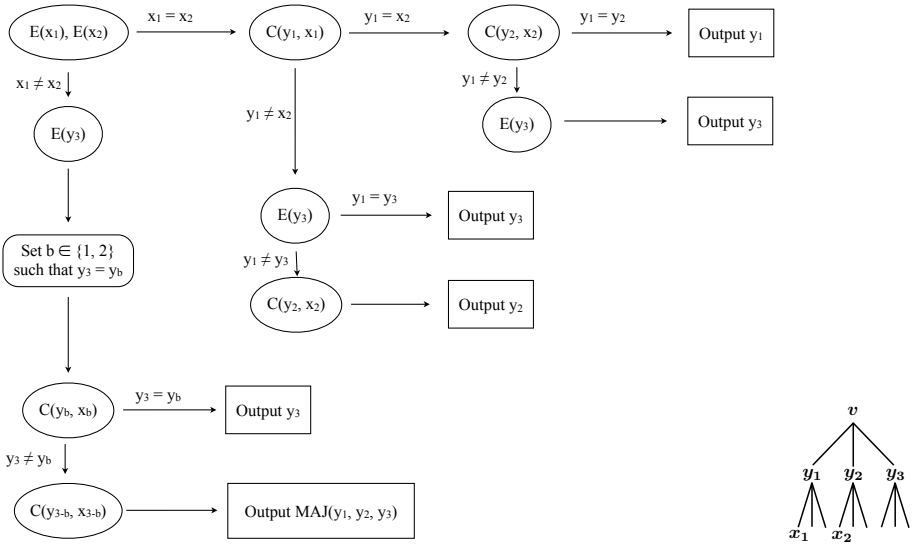


Fig. 1. Pictorial representation of algorithm EVALUATE on a subformula of height $h(v) \geq 2$ rooted at v. It is abbreviated by the letter 'E' when called recursively on descendants of v. The letter 'C' abbreviates the second algorithm COMPLETE.

step, permuting the input, means applying a random permutation to the children y_1, y_2, y_3 of v and independent random permutations to each of the three sets of grandchildren.

The second algorithm, COMPLETE, is depicted in Fig. 2. It takes two arguments v, y_1, and completes the evaluation of the subformula $\mathsf{3\text{-}MAJ}_h$ rooted at node v, where $h(v) \geq 1$, and y_1 is a child of v whose value has already been evaluated. The first step, permuting the input, means applying a random permutation to the children y_2, y_3 of v and independent random permutations to each of the two sets of grandchildren of y_2, y_3. Note that this is similar in form to the depth 2 algorithm of [5].

To evaluate an input of height h, we invoke EVALUATE(r), where r is the root. The correctness of the two algorithms follows by inspection—they determine the values of as many children of the node v as is required to compute the value of v.

For the complexity analysis, we study the expected number of queries they make for a worst-case input of fixed height h. (*A priori*, we do not know if such an input is a hard input as defined in Section 2.2.) Let $T(h)$ be the worst-case complexity of EVALUATE(v) for v of height h. For COMPLETE(v, y_1), we distinguish between two cases. Let y_1 be the child of node v that has already been evaluated. The complexity given that y_1 is the minority child of v is denoted by S^{m}, and the complexity given that it is a majority child is denoted by S^{M}.

The heart of our analysis is the following set of recurrences that relate T, S^{M} and S^{m} to each other.

Lemma 8. *It holds that* $S^{\mathrm{m}}(1) = 2$, $S^{\mathrm{M}}(1) = \frac{3}{2}$, $T(0) = 1$, *and* $T(1) = \frac{8}{3}$.

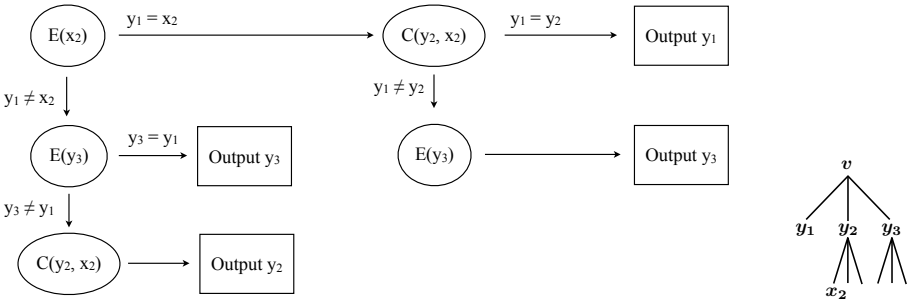


Fig. 2. Pictorial representation of algorithm COMPLETE on a subformula of height $h \geq 1$ rooted at v one child y_1 of which has already been evaluated. It is abbreviated by the letter 'C' when called recursively on descendants of v. Calls to EVALUATE are denoted 'E'.

For all $h \geq 1$, it holds that

$$S^{\mathrm{M}}(h) \leq S^{\mathrm{m}}(h) \qquad \textit{and} \qquad S^{\mathrm{M}}(h) \leq T(h) \ . \tag{4}$$

Finally, for all $h \geq 2$, it holds that

$$S^{\mathrm{m}}(h) = T(h-2) + T(h-1) + \frac{2}{3}\,S^{\mathrm{M}}(h-1) + \frac{1}{3}\,S^{\mathrm{m}}(h-1) \ , \tag{5}$$

$$S^{\mathrm{M}}(h) = T(h-2) + \frac{2}{3}\,T(h-1) + \frac{1}{3}\,S^{\mathrm{M}}(h-1) + \frac{1}{3}\,S^{\mathrm{m}}(h-1) \ , \quad \textit{and} \tag{6}$$

$$T(h) = 2\,T(h-2) + \frac{23}{27}\,T(h-1) + \frac{26}{27}\,S^{\mathrm{M}}(h-1) + \frac{18}{27}\,S^{\mathrm{m}}(h-1) \ . \tag{7}$$

Proof. We prove these relations by induction. The bounds for $h \in \{0, 1\}$ follow immediately by inspection of the algorithms. To prove the statement for $h \geq 2$, we assume the recurrences hold for all $l < h$. Observe that it suffices to prove Equations (5), (6), (7) for height h, since the values of the coefficients immediately imply that Inequalities (4) holds for h as well.

Equation (5). Since COMPLETE(v, y_1) always starts by computing the value of a grandchild x_2 of v, we get the first term $T(h-2)$ in Eq. (5). It remains to show that the worst-case complexity of the remaining queries is $T(h-1) + (2/3)S^{\mathrm{M}}(h-1) + (1/3)S^{\mathrm{m}}(h-1)$.

Since y_1 is the minority child of v, we have that $y_1 \neq y_2 = y_3$. The complexity of the remaining steps is summarized in the next table in the case that the three children of node y_2 are not all equal. In each line of the table, the worst case complexity is computed given the event in the first cell of the line. The second cell in the line is the probability of the event in the first cell over the random permutation of the children of y_2. This gives a contribution of $T(h-1) + (2/3)S^{\mathrm{M}}(h-1) + (1/3)S^{\mathrm{m}}(h-1)$.

$S^{\mathrm{m}}(h)$ (we have $y_1 \neq y_2 = y_3$)		
event	probability	complexity
$y_2 = x_2$	$2/3$	$T(h-1) + S^{\mathrm{M}}(h-1)$
$y_2 \neq x_2$	$1/3$	$T(h-1) + S^{\mathrm{m}}(h-1)$

This table corresponds to the worst case, as the only other case is when all children of y_2 are equal, in which the cost is $T(h-1) + S^{\mathrm{M}}(h-1)$. Applying Inequality (4) for $h-1$, this is a smaller contribution than the case where the children are not all equal.

Therefore the worst case complexity for S^{m} is given by Eq. (5). We follow the same convention and appeal to this kind of argument also while deriving the other two recurrence relations.

Equation (6). Since COMPLETE(v, y_1) always starts by computing the value of a grandchild x_2 of v, we get the first term $T(h-2)$ in Eq. (6). There are then two possible patterns, depending on whether the three children y_1, y_2, y_3 of v are all equal. If $y_1 = y_2 = y_3$, we have in the case that all children of y_2 are not equal that:

$S^{\mathrm{M}}(h)$ if $y_1 = y_2 = y_3$		
event	probability	complexity
$y_2 = x_2$	$2/3$	$S^{\mathrm{M}}(h-1)$
$y_2 \neq x_2$	$1/3$	$T(h-1)$

As in the above analysis of Eq. (5), applying Inequalities (4) for height $h-1$ implies that the complexity in the case when all children of y_2 are equal can only be smaller, therefore the above table describes the worst-case complexity for the case when $y_1 = y_2 = y_3$.

If y_1, y_2, y_3 are not all equal, we have two events $y_1 = y_2 \neq y_3$ or $y_1 = y_3 \neq y_2$ of equal probability as y_1 is a majority child of v. This leads to the following tables for the case where the children of y_2 are not all equal

$S^{\mathrm{M}}(h)$ given $y_1 = y_2 \neq y_3$		
event	prob.	complexity
$y_2 = x_2$	$2/3$	$S^{\mathrm{M}}(h-1)$
$y_2 \neq x_2$	$1/3$	$T(h-1) + S^{\mathrm{m}}(h-1)$

$S^{\mathrm{M}}(h)$ given $y_1 = y_3 \neq y_2$		
event	prob.	complexity
$y_2 = x_2$	$2/3$	$T(h-1)$
$y_2 \neq x_2$	$1/3$	$T(h-1) + S^{\mathrm{m}}(h-1)$

As before, one can apply Inequalities (4) for height $h-1$ to see that the worst case occurs when the children of y_2 are not all equal.

From the above tables, we deduce that the worst-case complexity occurs on inputs where y_1, y_2, y_3 are not all equal. This is because one can apply Inequalities (4) for height $h-1$ to see that, line by line, the complexities in the table for the case $y_1 = y_2 = y_3$ are upper bounded by the corresponding entries in each of the latter two tables. To conclude Eq. (6), recall that the two events $y_1 = y_2 \neq y_3$ and $y_1 = y_3 \neq y_2$ occur with probability $1/2$ each:

$$S^{\mathrm{M}}(h) = T(h-2) + \frac{1}{2}\left[\frac{2}{3}\,S^{\mathrm{M}}(h-1) + \frac{1}{3}\,(T(h-1) + S^{\mathrm{m}}(h-1))\right] + \frac{1}{2}\left[\frac{2}{3}\,T(h-1) + \frac{1}{3}\,(T(h-1) + S^{\mathrm{m}}(h-1))\right] \quad .$$

Equation (7). Since EVALUATE(v) starts with two calls to itself to compute x_1, x_2, we get the first term $2\,T(h-2)$ on the right hand side. The full analysis of Eq. (7) is similar to those of Eq. (5) and Eq. (6); we defer it to the journal article. □

Theorem 9. *$T(h), S^{\mathrm{M}}(h)$, and $S^{\mathrm{m}}(h)$ are all in $\mathrm{O}(\alpha^h)$, where $\alpha \le 2.64946$.*

Proof. We make an ansatz $T(h) \le a\,\alpha^h$, $S^{\mathrm{M}}(h) \le b\,\alpha^h$, and $S^{\mathrm{m}}(h) \le c\,\alpha^h$, and find constants a, b, c, α for which we may prove these inequalities by induction.

The base cases tell us that $2 \le c\alpha$, $\frac{3}{2} \le b\alpha$, $1 \le a$, and $\frac{8}{3} \le a\alpha$.

Assuming we have constants that satisfy these conditions, and that the inequalities hold for all appropriate $l < h$, for some $h \ge 2$, we derive sufficient conditions for the inductive step to go through.

By the induction hypothesis, Lemma 8, and our ansatz, it suffices to show

$$a + \tfrac{3a+2b+c}{3}\alpha \le c\,\alpha^2 \qquad a + \tfrac{2a+b+c}{3}\alpha \le b\,\alpha^2 \qquad 2a + \tfrac{23a+26b+18c}{27}\alpha \le a\,\alpha^2 \quad (8)$$

The choice $\alpha = 2.64946$, $a = 1.007$, $b = 0.55958\,a$, and $c = 0.75582\,a$ satisfies the base case as well as all the Inequalities (8), so the induction holds. □

References

[1] Blum, M., Impagliazzo, R.: General oracle and oracle classes. In: Proc. FOCS 1987, pp. 118–126 (1987)

[2] Hartmanis, J., Hemachandra, L.: One-way functions, robustness, and non-isomorphism of NP-complete sets. In: Proc. Struc. in Complexity Th. 1987, pp. 160–173 (1987)

[3] Heiman, R., Newman, I., Wigderson, A.: On read-once threshold formulae and their randomized decision tree complexity. In: Proc. Struc. in Complexity Th. 1990, pp. 78–87 (1990)

[4] Heiman, R., Wigderson, A.: Randomized versus deterministic decision tree complexity for read-once boolean functions. In: Proc. Struc. in Complexity Th. 1991, pp. 172–179 (1991)

[5] Jayram, T., Kumar, R., Sivakumar, D.: Two applications of information complexity. In: Proc. STOC 2003, pp. 673–682 (2003)

[6] Landau, I., Nachmias, A., Peres, Y., Vanniasegaram, S.: The lower bound for evaluating a recursive ternary majority function: an entropy-free proof. Tech. rep., Dep. of Stat., UC Berkeley (2006) (undergraduate Research Report), http://www.stat.berkeley.edu/110

[7] Nisan, N.: CREW PRAMs and decision trees. In: Proc. STOC 1989, pp. 327–335. ACM, New York (1989)

[8] Reichardt, B.W., Špalek, R.: Span-program-based quantum algorithm for evaluating formulas. In: Proc. 40th STOC, pp. 103–112. ACM, New York (2008)
[9] Saks, M., Wigderson, A.: Probabilistic boolean decision trees and the complexity of evaluating game trees. In: Proc. FOCS 1986, pp. 29–38 (1986)
[10] Santha, M.: On the Monte Carlo boolean decision tree complexity of read-once formulae. Random Structures and Algorithms 6(1), 75–87 (1995)
[11] Snir, M.: Lower bounds for probabilistic linear decision trees. Combinatorica 9, 385–392 (1990)
[12] Tardos, G.: Query complexity or why is it difficult to separate $\mathbf{NP}^A \cap \mathbf{coNP}^A$ from $\mathbf{P}^A$ by a random oracle. Combinatorica 9, 385–392 (1990)

The Fourier Entropy–Influence Conjecture for Certain Classes of Boolean Functions*

Ryan O'Donnell, John Wright, and Yuan Zhou

Department of Computer Science,
Carnegie Mellon University
{odonnell,jswright,yuanzhou}@cs.cmu.edu

Abstract. In 1996, Friedgut and Kalai made the *Fourier Entropy–Influence Conjecture*: For every Boolean function $f\colon \{-1,1\}^n \to \{-1,1\}$ it holds that $\boldsymbol{H}[\widehat{f^2}] \leq C \cdot \mathbf{I}[f]$, where $\boldsymbol{H}[\widehat{f^2}]$ is the spectral entropy of f, $\mathbf{I}[f]$ is the total influence of f, and C is a universal constant. In this work we verify the conjecture for symmetric functions. More generally, we verify it for functions with symmetry group $S_{n_1} \times \cdots \times S_{n_d}$ where d is constant. We also verify the conjecture for functions computable by read-once decision trees.

1 Introduction

The field of *Fourier analysis of Boolean functions* $f\colon \{-1,1\}^n \to \{-1,1\}$ plays an important role in many areas of mathematics and computer science, including complexity theory, learning theory, random graphs, social choice, inapproximability, arithmetic combinatorics, coding theory, metric spaces, etc. For a survey, see e.g. [17]. One of the most longstanding and important open problems in the field is the *Fourier Entropy–Influence (FEI) Conjecture* made by Friedgut and Kalai in 1996 [6]:

Fourier Entropy–Influence (FEI) Conjecture. $\exists C\ \forall f$, $\boldsymbol{H}[\widehat{f^2}] \leq C \cdot \mathbf{I}[f]$. That is,

$$\sum_{S \subseteq [n]} \widehat{f}(S)^2 \log_2 \frac{1}{\widehat{f}(S)^2} \leq C \cdot \sum_{S \subseteq [n]} \widehat{f}(S)^2 |S|.$$

The quantity $\boldsymbol{H}[\widehat{f^2}] = \sum \widehat{f}(S)^2 \log \frac{1}{\widehat{f}(S)^2}$ on the left is the *spectral entropy* or *Fourier entropy* of f. It ranges between 0 and n and measures how "spread out" f's Fourier spectrum is. The quantity $\mathbf{I}[f] = \sum \widehat{f}(S)^2 |S|$ appearing on the right is the *total influence* or *average sensitivity* of f. It also ranges between 0 and n and measures how "high up" f's Fourier spectrum is. (For definitions of the terms used in this introduction, see Section 2.)

* This research performed while the first author was a member of the School of Mathematics, Institute for Advanced Study. Supported by NSF grants CCF-0747250 and CCF-0915893, BSF grant 2008477, and Sloan and Okawa fellowships.

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 330–341, 2011.

The FEI Conjecture is superficially similar to the well-known *Logarithmic Sobolev Inequality* [9] for the Boolean cube which states that $\mathbf{Ent}[f^2] \leq 2 \cdot \mathbf{I}[f]$ holds for any $f\colon \{-1,1\}^n \to \mathbb{R}$, where $\mathbf{Ent}[g] = \mathbf{E}[g \ln g] - \mathbf{E}[g] \ln \mathbf{E}[g]$. However note that the FEI Conjecture requires $f\colon \{-1,1\}^n \to \{-1,1\}$ to be Boolean-*valued*, and it definitely fails for real-valued f.

1.1 Applications of the Conjecture

Friedgut and Kalai's original motivation for the FEI Conjecture came from the theory of random graphs. Suppose f represents a monotone graph property with $\mathbf{Pr}[f(G) = 1] = 1/2$ when $G \sim G(v, 1/2)$ (here $n = \binom{v}{2}$). If we also consider $\mathbf{Pr}[f(G) = 1]$ for $G \sim G(v,p)$, then the total influence $\mathbf{I}[f]$ equals the reciprocal of the derivative of this quantity at $p = 1/2$. Hence the property has "sharp threshold" if and only if $\mathbf{I}[f]$ is large. Friedgut and Kalai sought general conditions on f which would force $\mathbf{I}[f]$ to be large. They conjectured that having significant symmetry — and hence, a spread-out Fourier spectrum — was such a property.

The FEI Conjecture also easily implies the famed KKL Theorem [11]. To see this, first note that $\boldsymbol{H}[\widehat{f^2}] \geq \boldsymbol{H}_\infty[\widehat{f^2}] = \min_S\{\log \frac{1}{\widehat{f}(S)^2}\}$, the *min-entropy* of $\widehat{f^2}$. Thus the FEI Conjecture is strictly stronger than the following:

Fourier Min-Entropy–Influence Conjecture $\exists C\ \forall f \quad \boldsymbol{H}_\infty[\widehat{f^2}] \leq C \cdot \mathbf{I}[f]$. *That is,* $\exists S \subseteq [n]$ *such that* $\widehat{f}(S)^2 \geq 2^{-C \cdot \mathbf{I}[f]}$.

In particular, for balanced f (i.e., $\mathbf{E}[f] = 0 = \widehat{f}(\emptyset)^2$) the above conjecture implies there is a nonempty S with $\widehat{f}(S)^2 \geq 2^{-C \cdot \mathbf{I}[f]}$. Since $\mathbf{Inf}_j[f] \geq \widehat{f}(S)^2$ for each $j \in S$ we conclude $\max\{\mathbf{Inf}_i[f]\} \geq 2^{-C \cdot n \cdot \max\{\mathbf{Inf}_i[f]\}}$ whence $\max\{\mathbf{Inf}_i[f]\} \geq \Omega(\frac{1}{C}) \cdot \frac{\log n}{n}$, which is KKL's conclusion. Indeed, by applying the above deduction just to the *nonempty* Fourier coefficients it is straightforward to deduce $\mathbf{I}[f] \geq \frac{1}{C} \mathbf{Var}[f] \log \frac{1}{\max_i\{\mathbf{Inf}_i[f]\}}$, a strengthening of the KKL Theorem due to Talagrand [20]. We also remark that since $\mathbf{Inf}_i[f] = \widehat{f}(\{i\})$ for monotone f, the KKL Theorem implies the Fourier Min-Entropy–Influence Conjecture holds for monotone functions.

Finally, as emphasized by Klivans and coauthors [8,14], the FEI Conjecture is also important because it implies a version of *Mansour's Conjecture* from 1994.

Mansour's Conjecture [15]. $\forall \epsilon > 0\ \exists K$ *such that if* $f\colon \{-1,1\}^n \to \{-1,1\}$ *is computable by a DNF formula with* $m \geq 2$ *terms then by taking* $\mathcal{S}$ *to be the* m^K *sets* S *for which* $\widehat{f}(S)^2$ *is largest,* $\sum_{S \notin \mathcal{S}} \widehat{f}(S)^2 \leq \epsilon$.

In fact, Mansour conjectured more strongly that one may take $K = O(\log \frac{1}{\epsilon})$. It is well known [3] that if f is computed by an m-term DNF then $\mathbf{I}[f] \leq O(\log m)$. Thus the Fourier Entropy–Influence Conjecture would imply $\boldsymbol{H}[\widehat{f^2}] \leq C \cdot O(\log m)$, from which it easily follows that one may take $K = O(C/\epsilon)$ in Mansour's Conjecture. Mansour's Conjecture is important because if it is true then the query algorithm of Gopalan, Kalai, and Klivans [7] would agnostically learn DNF formulas under the uniform distribution to any constant accuracy in

polynomial time. Establishing such a result is a major open problem in computational learning theory [8]. Further, sufficiently strong versions of Mansour's Conjecture would yield improved pseudorandom generators for DNF formulas; see, e.g., [4,14] for more on this important open problem in pseudorandomness.

1.2 Prior Work

As far as we are aware, the result in [14] showing that the FEI Conjecture holds for random DNFs is the only published progress on the FEI Conjecture since it was posed. In this subsection we collect some observations related to the conjecture, all of which were presumably known to Friedgut and Kalai and should be considered folklore. See also [12] for additional recent discussion of the conjecture.

The FEI Conjecture holds for "the usual examples" that arise in analysis of Boolean functions — Parities (for which the conjecture is trivial), ANDs and ORs, Majority, Tribes [1], and Inner-Product-mod-2. This may be established by direct calculation based on the known Fourier coefficient formulas for these functions (see [21] for Majority and [16] for Tribes). By considering the AND and OR functions it is easy to show that the constant C must be at least 4. We can show that $C = 4$ is necessary and sufficient for the Tribes functions as well; smaller constants suffice for Inner-Product-mod-2 and Majority. The authors are also aware of an explicit family of functions which show the necessity of $C \geq 60/13 \approx 4.615$. For a gross upper bound, it is not too hard to show that $\boldsymbol{H}[\widehat{f}^2] \leq (1 + \log n) \cdot \mathbf{I}[f] + 1$; indeed, this will be shown in the course of the present paper.

The FEI Conjecture "tensorizes" in the following sense: For $f\colon \{-1,1\}^n \to \{-1,1\}$ and $M \in \mathbb{Z}_+$, define $f^{\oplus M} \to \{-1,1\}^{Mn} \to \{-1,1\}$ by $f(x^{(1)}, \dots, x^{(M)}) = f(x^{(1)})f(x^{(2)}) \cdots f(x^{(M)})$. Then it's easy to check that $\boldsymbol{H}[\widehat{f^{\oplus M}}^2] = M \cdot \boldsymbol{H}[\widehat{f}^2]$ and $\mathbf{I}[f] = M \cdot \mathbf{I}[f]$. This is of course consistent with the FEI Conjecture; it also implies that the following weaker-looking conjecture is actually equivalent to FEI:

$$\text{for all } f\colon \{-1,1\}^n \to \{-1,1\}, \quad \boldsymbol{H}[\widehat{f}^2] \leq C \cdot \mathbf{I}[f] + o(n).$$

To see the equivalence, given $f\colon \{-1,1\}^n \to \{-1,1\}$, apply the above to $f^{\oplus M}$, divide by M, and take the limit as $M \to \infty$.

1.3 Our Results and Approach

In this work, we prove the FEI Conjecture for some classes of Boolean functions.

Theorem 1. *The FEI Conjecture holds for symmetric functions, with* $C = 12.04$.

Although the class of symmetric functions is fairly small, there was sentiment that it might be a difficult case for FEI: for symmetric functions, $\widehat{f}(S) = \widehat{f}(S')$ whenever $|S| = |S'|$ and hence their Fourier spectrum is maximally spread out on each level.

Our proof of Theorem 1 uses the same high-level idea as in the well-known KKL Theorem [11]: namely, prove a certain inequality for the *discrete derivatives* $D_i f$ of f, and then sum over i. In our case, the key inequality we need for the derivatives is that they are very noise-sensitive:

Theorem. Let g be a discrete derivative of a symmetric function $f\colon \{-1,1\}^{n+1} \to \{-1,1\}$. Then for all real $1 \le c \le n$ it holds that $\mathbf{Stab}_{1-\frac{c}{n}}[g] \le \frac{2/\sqrt{\pi}}{\sqrt{c}}\,\mathbf{E}[g^2]$.

(For the notation used here, see Section 2.) Having established Theorem 1, it is not too hard to generalize it as follows:

Theorem 2. *The FEI Conjecture holds for* d-part-symmetric *functions, with* $C = 12.04 + \log_2 d$.

A d-part-symmetric function is essentially a function with symmetry group of the form $S_{n_1} \times \cdots \times S_{n_d}$. This theorem also generalizes the folklore fact that FEI holds (up to an additive constant) with $C = O(\log n)$, since *every* function is n-part-symmetric.

Finally, with an unrelated, direct inductive argument we can also prove:

Theorem 3. *The FEI Conjecture holds for functions computable by read-once decision trees, with* $C = 4.88$.

The notion of read-once decision tree is defined as follows. We first recall the definition of *decition tree*. We say that $f\colon \{-1,1\}^n \to \{-1,1\}$ is computable as a depth-0 decision tree if it is constantly -1 or 1. We inductively say that it is computable as a depth-d decision tree if there is a coordinate $i \in [n]$ such that $f(x) = f_0(x)$ when $x_i = 1$ and $f(x) = f_1(x)$ when $x_i = 0$, where f_0 and f_1 are computable by depth-$(d-1)$ decision trees. We further say that the decision-tree computation is *read-once* if f_0 and f_1 depend on disjoint sets of coordinates and are themselves inductively read-once.

Remark: We can extend the proof of Theorem 3 to handle the more general class of "recursively read-once functions" (as defined in [18]), with $C = 19$.

Remark: In independent and concurrent work, the FEI Conjecture was verified for monotone symmetric functions (a special case of Theorem 1) and decision lists (a special case of Theorem 3) using different methods of proof [22].

1.4 Outline for the Rest of the Paper

In Section 2, we introduce relevant definitions and notations. In Section 3, we prove Theorem 1 and Theorem 2. In Section 4, we put several remarks on future directions towards resolving the Fourier Entropy–Influence Conjecture.

Due to space reasons, we are not able to show the proof of Theorem 3 in this version of the paper.

2 Definitions and Notation

We use the notation $\mathbb{N} = \{0, 1, 2, \dots\}$, $\mathbb{Z}_+ = \mathbb{N} \setminus \{0\}$, and $[n] = \{1, 2, \cdots, n\}$. Throughout we write log for $\log_2$; for the natural logarithm we write ln. The expressions $0 \log 0$ and $0 \log \frac{1}{0}$ are to be interpreted as 0.

2.1 Basics of Boolean Fourier Analysis

This paper is concerned with *Boolean functions* $f\colon \{-1,1\}^n \to \mathbb{R}$, especially *Boolean-valued functions* $f\colon \{-1,1\}^n \to \{-1,1\}$. Every Boolean function f has a unique multilinear polynomial representation over $\mathbb{R}$,

$$f(x) = \sum_{S \subseteq [n]} \widehat{f}(S) \prod_{i \in S} x_i.$$

This is known as the *Fourier expansion* of f, and the real numbers $\widehat{f}(S)$ are the *Fourier coefficients* of f. We have the formula $\widehat{f}(S) = \mathbf{E}[f(x) \prod_{i \in S} x_i]$. (Here and throughout, expectation $\mathbf{E}[\cdot]$ is with respect to the uniform distribution of x on $\{-1,1\}^n$, unless otherwise specified.) In particular, $\widehat{f}(\emptyset) = \mathbf{E}[f]$. An important basic fact about Fourier coefficients is *Parseval's identity*: $\sum_{S \subseteq [n]} \widehat{f}(S)^2 = \mathbf{E}[f(x)^2]$. A consequence of Parseval is that $\sum_{S \subseteq [n]} \widehat{f}(S)^2 = 1$ for Boolean-valued f. Thus the numbers $\widehat{f}(S)^2$ can be thought of as a probability distribution on the subsets of $[n]$.

Given $f\colon \{-1,1\}^n \to \mathbb{R}$ and $i \in [n]$, we define the *discrete derivative* $D_i f : \{-1,1\}^n \to \mathbb{R}$ by $D_i f(x) = \frac{f(x^{(i=1)}) - f(x^{(i=-1)})}{2}$, where $x^{(i=b)}$ denotes $(x_1, \dots, x_{i-1}, b, x_{i+1}, \dots, x_n)$. It holds that

$$\widehat{D_i f}(S) = \begin{cases} 0 & \text{if } i \in S, \\ \widehat{f}(S \cup \{i\}) & \text{if } i \notin S; \end{cases}$$

i.e,. D_i acts on the Fourier expansion as formal differentiation. The *influence of i on f* is

$$\mathbf{Inf}_i[f] = \mathbf{E}[(D_i f)^2] = \sum_{S \ni i} \widehat{f}(S)^2.$$

In the particular case that f is Boolean-valued, the derivative $D_i f$ is $\{-1, 0, 1\}$-valued and we have the combinatorial interpretation $\mathbf{Inf}_i[f] = \mathbf{Pr}[f(x^{(i=1)}) \neq f(x^{(i=-1)})]$. The *total influence* of $f\colon \{-1,1\}^n \to \mathbb{R}$ is

$$\mathbf{I}[f] = \sum_{i=1}^{n} \mathbf{Inf}_i[f] = \sum_{S \subseteq [n]} \widehat{f}(S)^2 |S|.$$

For $0 \leq \rho \leq 1$, we say that $x, y \in \{-1,1\}^n$ are a pair of *ρ-correlated random strings* if x is distributed uniformly randomly on $\{-1,1\}^n$ and y is formed by setting $y_i = x_i$ with probability $\frac{1}{2} + \frac{1}{2}\rho$, $y_i = -x_i$ with probability $\frac{1}{2} - \frac{1}{2}\rho$,

independently for each $i \in [n]$. We may now define the *noise stability of f at ρ* and give its Fourier formula:

$$\mathbf{Stab}_\rho[f] = \mathop{\mathbf{E}}_{x,y \ \rho\text{-correlated}}[f(x)f(y)] = \sum_{S \subseteq [n]} \widehat{f}(S)^2 \rho^{|S|}.$$

We often stratify the Fourier coefficients into *levels*; the level of S is simply $|S|$. We define the *weight of f at level k* to be $\mathbf{W}^k[f] = \sum_{|S|=k} \widehat{f}(S)^2$. Thus

$$\mathbf{I}[f] = \sum_{k=0}^{n} \mathbf{W}^k[f] \cdot k, \quad \mathbf{Stab}_\rho[f] = \sum_{k=0}^{n} \mathbf{W}^k[f] \rho^k.$$

Finally, given a random variable or probability distribution we write $\boldsymbol{H}[\cdot]$ for its (binary) Shannon entropy. Hence for $f\colon \{-1,1\}^n \to \{-1,1\}$, $\boldsymbol{H}[\widehat{f}^2] = \sum_{S \subseteq [n]} \widehat{f}(S)^2 \log \frac{1}{\widehat{f}(S)^2}$, called the *Fourier entropy*, or *spectral entropy*, of f. Thus the Fourier Entropy–Influence Conjecture may be stated as follows: there is a universal constant C such that $\boldsymbol{H}[\widehat{f}^2] \le C \cdot \mathbf{I}[f]$ holds for all Boolean-valued functions f.

2.2 Some Boolean Function Classes

We will call a function $f\colon \{-1,1\}^n \to \mathbb{R}$ *symmetric* if it is invariant under any permutation of the coordinates $[n]$. Equivalently, f is symmetric if the value of $f(x)$ depends only the *Hamming weight* of x, defined to be $\#\{i \in [n] : x_i = -1\}$. In this case we may identify f with the function $f\colon \{0,1,\dots,n\} \to \mathbb{R}$ whose value at s equals $f(x)$ for any x of Hamming weight s.

We generalize the notion to that of *d-part-symmetric* functions, $d \in \mathbb{Z}_+$. We say the function $f\colon \{-1,1\}^n \to \mathbb{R}$ is d-part-symmetric if there is a partition $[n] = V_1 \cup V_2 \cup \cdots \cup V_d$ such that f is invariant under any permutation of the coordinates in any part V_i. Equivalently, f is d-part-symmetric if, after relabeling coordinates, it has the same output value under the action of $S_{n_1} \times \cdots \times S_{n_d}$ for some numbers $n_1 + \cdots + n_d = n$. Note that a symmetric function is 1-part-symmetric, and every function $f\colon \{-1,1\}^n \to \mathbb{R}$ is n-part-symmetric.

We also generalize the notion of Fourier "levels" for d-part-symmetric functions. Suppose f is d-part-symmetric with respect to the partition $[n] = V_1 \cup \cdots \cup V_d$, where $|V_i| = n_i$. Then $\widehat{f}(S)$ depends only on the numbers $|S \cap V_1|$, …, $|S \cap V_d|$. We consider all possible such sequences

$$\boldsymbol{k} \in \{0,1,\dots,n_1\} \times \{0,1,\dots,n_2\} \times \cdots \times \{0,1,\dots,n_d\},$$

and say that $S \subseteq [n]$ is at *multi-level* $\boldsymbol{k}$ if $|S \cap V_i| = \boldsymbol{k}_i$ for each $i \in [d]$. We also use the notation

$$|\boldsymbol{k}| = \boldsymbol{k}_1 + \boldsymbol{k}_2 + \cdots + \boldsymbol{k}_d, \qquad \mathbf{W}^{\boldsymbol{k}}[f] = \sum_{S \text{ at multi-level } \boldsymbol{k}} \widehat{f}(S)^2,$$

so $\mathbf{I}[f] = \sum_{\boldsymbol{k}} \mathbf{W}^{\boldsymbol{k}}[f] \cdot |\boldsymbol{k}|$.

3 Symmetric and d-Part-Symmetric Functions

In this section we prove Theorems 1 and 2, establishing the FEI Conjecture for symmetric and $O(1)$-part-symmetric functions. Although Theorem 2 strictly generalizes Theorem 1, we prefer to prove Theorem 1 separately and then generalize it afterward.

When $f\colon \{-1,1\}^n \to \{-1,1\}$ is symmetric we have $\widehat{f}(S)^2 = \mathbf{W}^k[f]/\binom{n}{k}$ whenever $|S| = k$. Hence

$$\boldsymbol{H}[\widehat{f}^2] = \sum_{k=0}^{n} \mathbf{W}^k[f] \log \tfrac{\binom{n}{k}}{\mathbf{W}^k[f]} = \sum_{k=0}^{n} \mathbf{W}^k[f] \log \binom{n}{k} + \sum_{k=0}^{n} \mathbf{W}^k[f] \log \tfrac{1}{\mathbf{W}^k[f]}. \quad (1)$$

Thus Theorem 1 is an immediate consequence of the following two theorems:

Theorem 4. *Let $f\colon \{-1,1\}^n \to \{-1,1\}$ be a symmetric function. Then $\sum_{k=0}^{n} \mathbf{W}^k[f] \log \binom{n}{k} \le C_1 \cdot \mathbf{I}[f]$, where $C_1 = \frac{1}{\ln 2}(1 + \frac{4\sqrt{2e}}{\sqrt{\pi}}) \le 9.04$.*

Theorem 5. *Let $f\colon \{-1,1\}^n \to \{-1,1\}$ be any function, not necessarily symmetric. Then $\sum_{k=0}^{n} \mathbf{W}^k[f] \log \frac{1}{\mathbf{W}^k[f]} \le 3 \cdot \mathbf{I}[f]$.*

We prove these theorems in the subsequent subsections of the paper, following which we give the extension to d-part-symmetric functions.

3.1 Theorem 4: Derivatives of Symmetric Functions are Noise-Sensitive

We begin with an easy lemma, the proof of which is omitted due to space reasons.

Lemma 1. *Let $p_1, \dots, p_m$ be a nonnegative unimodal sequence; i.e., there exists $k \in [m]$ such that $p_1, \dots, p_k$ is a nondecreasing sequence and $p_k, \dots, p_m$ is a nonincreasing sequence. Let $g : [m] \to \{-1,0,1\}$ have the property that the sets $g^{-1}(-1)$ and $g^{-1}(1)$ are interleaving. Then $|\sum_{i=1}^{m} p_i g(i)| \le \max\{p_i\}$.*

We now show the key theorem stated in Section 1.3 on the noise sensitivity of symmetric derivatives.

Theorem 6. *Let g be a discrete derivative of a symmetric function $f\colon \{-1,1\}^{n+1} \to \{-1,1\}$. Then for all real $1 \le c \le n$ it holds that $\mathbf{Stab}_{1-\frac{c}{n}}[g] \le \frac{2/\sqrt{\pi}}{\sqrt{c}} \mathbf{E}[g^2]$.*

Proof. Let (x, y) be a $(1 - \frac{c}{n})$-correlated pair of random strings in $\{-1,1\}^n$. We will show that

$$\Big|\mathbf{E}[g(y) \mid x]\Big| \le \frac{2/\sqrt{\pi}}{\sqrt{c}}, \quad \text{independent of } x. \quad (2)$$

Since g is $\{-1,0,1\}$-valued, it will follow from (2) that

$$\mathbf{Stab}_{1-\frac{c}{n}}[g] = \mathbf{E}[g(x)g(y)] \le \frac{2/\sqrt{\pi}}{\sqrt{c}} \mathbf{E}[|g(x)|] = \frac{2/\sqrt{\pi}}{\sqrt{c}} \mathbf{E}[g^2],$$

as required. To show (2) we first observe that given x of Hamming weight s, the Hamming weight t of y is distributed as the sum of two independent binomial random variables, $t_1 \sim \mathrm{Bin}(s, 1-\frac{c}{2n})$ and $t_2 \sim \mathrm{Bin}(n-s, \frac{c}{2n})$. Being the sum of independent Bernoulli random variables, it is well known [13] that t has a unimodal probability distribution. Since g is a derivative of a symmetric Boolean-valued function, it is itself symmetric; hence we may identify it with a function $g : \{0, 1, \ldots, n\} \to \{-1, 0, 1\}$. Further, because f is $\{-1, 1\}$-valued, g must have the property that $g^{-1}(-1)$ and $g^{-1}(1)$ are interleaving subsets of $\{0, 1, \ldots, n\}$. Thus (2) follows from Lemma 1 assuming that $\max_i\{\mathbf{Pr}[t = i]\} \le \frac{2/\sqrt{\pi}}{\sqrt{c}}$. To show this, we may assume without loss of generality that $s \ge n/2$. Then

$$\max_i\{\mathbf{Pr}[t = i]\} \le \max_i\{\mathbf{Pr}[t_1 = i]\} \le \frac{1}{\sqrt{2\pi s(1-\frac{c}{2n})\frac{c}{2n}}} \le \frac{1}{\sqrt{2\pi\frac{n}{2}\frac{1}{2}\frac{c}{2n}}} = \frac{2/\sqrt{\pi}}{\sqrt{c}},$$

where the second inequality is the basic estimate $\max_i\{\mathbf{Pr}[\mathrm{Bin}(m, p)] = i\} \le \frac{1}{\sqrt{2\pi mp(1-p)}}$ which uses Stirling's formula (see [5, Ch. VII.3]).

We can now give the proof of Theorem 4.

Proof. (Theorem 4.) Using $\binom{n}{k} \le (\frac{en}{k})^k$ for all $1 \le k \le n$ we have

$$\sum_{k=0}^{n} \mathbf{W}^k[f] \log \binom{n}{k} \le \sum_{k=1}^{n} \mathbf{W}^k[f] k \log(e) + \sum_{k=1}^{n} \mathbf{W}^k[f] k \log \tfrac{n}{k},$$

since $\sum_{k=1}^{n} \mathbf{W}^k[f] k \log(e) = \mathbf{I}[f]/(\ln 2)$, it suffices to show

$$\sum_{k=1}^{n} \mathbf{W}^k[f] k \ln \tfrac{n}{k} \le \frac{4\sqrt{2e}}{\sqrt{\pi}} \cdot \mathbf{I}[f]. \tag{3}$$

Let g be any derivative of f; say $g = D_n f$. By symmetry of f, the right side of (3) is $\frac{4\sqrt{2e}}{\sqrt{\pi}} \cdot n\,\mathbf{E}[g^2]$. As for the left side, for $k \in [n]$ we have

$$\mathbf{W}^{k-1}[g] = \sum_{\substack{S \subseteq [n-1] \\ |S|=k-1}} \widehat{g}(S)^2 = \sum_{\substack{S \subseteq [n] \\ |S|=k, S \ni n}} \widehat{f}(S)^2 = \frac{k}{n} \mathbf{W}^k[f].$$

Hence the left side of (3) is $\sum_{k=1}^{n} n\mathbf{W}^{k-1}[g] \ln \frac{n}{k}$. Thus after dividing by n we see that (3) is equivalent to

$$\sum_{k=0}^{n-1} \mathbf{W}^k[g] \ln \frac{n}{k+1} \le \frac{4\sqrt{2e}}{\sqrt{\pi}} \cdot \mathbf{E}[g^2]. \tag{4}$$

Using the approximation $\ln m + \gamma + \frac{1}{2m} - \frac{1}{12m^2} \le \sum_{j=1}^{m} \frac{1}{j} \le \ln m + \gamma + \frac{1}{2m}$ one may obtain

$$\sum_{k=0}^{n-1} \mathbf{W}^k[g] \ln \frac{n}{k+1} \le \sum_{k=0}^{n-1} \mathbf{W}^k[g] \sum_{j=k+1}^{n} \frac{1}{j} = \sum_{j=1}^{n} \frac{1}{j} \sum_{k=0}^{j-1} \mathbf{W}^k[g] \le \sqrt{e} \sum_{j=1}^{n} \frac{1}{j} \mathbf{Stab}_{1-\frac{1}{2j}}[g],$$

where in the last step we used that $\sqrt{e}(1-\frac{1}{2j})^k \geq 1$ for all $k \leq j-1$. We may now apply Theorem 6 with $c = \frac{n}{2j}$ to obtain

$$\sum_{k=0}^{n} \mathbf{W}^k[g] \ln \frac{n}{k+1} \leq \sqrt{e} \sum_{j=1}^{n} \frac{1}{j} \frac{2\sqrt{2}}{\sqrt{\pi}} \frac{\sqrt{j}}{\sqrt{n}} \cdot \mathbf{E}[g^2] \leq \frac{4\sqrt{2e}}{\sqrt{\pi}} \cdot \mathbf{E}[g^2],$$

using $\sum_{j=1}^{n} \frac{1}{\sqrt{j}} \leq 2\sqrt{n}$. Thus we have verified (4) and completed the proof.

3.2 Theorem 5: Spectral Level Entropy Versus Influence

In this section we establish Theorem 5. We begin with a well-known fact about "maximum entropy" which we prove for completeness:

Proposition 1. *Let K be a random variable supported on $\mathbb{Z}_+$. Then $\boldsymbol{H}[K] \leq \mathbf{E}[K]$.*

Proof. For $k \in \mathbb{Z}_+$ write $p_k = \mathbf{Pr}[K = k]$, and let G be a Geometric($\frac{1}{2}$) random variable. Then by the nonnegativity of binary relative entropy,

$$0 \leq \mathbf{D}(K||G) = \sum_{k=1}^{n} p_k \log \frac{p_k}{(1/2)^k} = -\boldsymbol{H}[P] + \sum_{k=1}^{n} p_k \log(2^k) = \mathbf{E}[K] - \boldsymbol{H}[K].$$

We will also need a simple corollary of the edge-isoperimetric inequality for $\{-1,1\}^n$.

Proposition 2. *Let $f\colon \{-1,1\}^n \to \{-1,1\}$ and write $\mathbf{W}^0[f] = 1-\epsilon$. Then $\mathbf{I}[f] \geq \frac{1}{2}\epsilon \log(1/\epsilon) + \epsilon$.*

Proof. By negating f if necessary we may assume $\mathbf{E}[f] \leq 0$. We think of f as the indicator of a subset $A \subseteq \{0,1\}^n$; the density of this set is $p = \frac{1}{2} + \frac{1}{2}\mathbf{E}[f] = \frac{1}{2} - \frac{1}{2}\sqrt{1-\epsilon} \leq 1/2$. It follows from the edge-isoperimetric inequality [10,19,2] that $\mathbf{I}[f]$, which is 2^{-n} times the size of A's edge-boundary, is at least $2p\log(1/p)$. It is then elementary to check that $(1-\sqrt{1-\epsilon})\log(\frac{2}{1-\sqrt{1-\epsilon}}) \geq \frac{1}{2}\epsilon\log(1/\epsilon) + \epsilon$, as required.

We can now establish Theorem 5.

Proof. (Theorem 5.) Write $\mathbf{W}^0[f] = 1-\epsilon$. We may assume $\epsilon > 0$ else the result is trivial. Then

$$\sum_{k=0}^{n} \mathbf{W}^k[f] \log \tfrac{1}{\mathbf{W}^k[f]} \leq \left(\sum_{k \neq 0} \mathbf{W}^k[f] \log \tfrac{1}{\mathbf{W}^k[f]} \right) + 2\epsilon \quad (\text{as } (1-\epsilon)\log\tfrac{1}{1-\epsilon} \leq \tfrac{1}{\ln 2}\epsilon \leq 2\epsilon)$$

$$= \epsilon \left(\sum_{k \neq 0} \frac{\mathbf{W}^k[f]}{\epsilon} \log \frac{\epsilon}{\mathbf{W}^k[f]} \right) + \epsilon \log \frac{1}{\epsilon} + 2\epsilon \leq \epsilon \left(\sum_{k \neq 0} \frac{\mathbf{W}^k[f]}{\epsilon} \cdot k \right) + 2 \cdot \mathbf{I}[f] = 3 \cdot \mathbf{I}[f],$$

where the last inequality used Propositions 1 and 2.

3.3 Theorem 2: Extension to d-Part-Symmetric Functions

We show now how to extend the proof of Theorem 1 to obtain Theorem 2, the FEI Conjecture for d-part-symmetric functions. Suppose then that f is d-part-symmetric with respect to the partition $[n] = V_1 \cup \cdots \cup V_d$, where $|V_i| = n_i$, and recall the multi-level index notation $\boldsymbol{k}$ from Section 2.2. Since $\widehat{f}(S)^2 = \mathbf{W}^{\boldsymbol{k}}[f] / \prod_{i=1}^{d} \binom{n_i}{\boldsymbol{k}_i}$ whenever S is at multi-level $\boldsymbol{k}$, we have

$$\boldsymbol{H}[\widehat{f^2}] = \sum_{i=1}^{d} \sum_{\boldsymbol{k}} \mathbf{W}^{\boldsymbol{k}}[f] \log \binom{n_i}{\boldsymbol{k}_i} + \sum_{\boldsymbol{k}} \mathbf{W}^{\boldsymbol{k}}[f] \log \tfrac{1}{\mathbf{W}^{\boldsymbol{k}}[f]},$$

similarly to (1). Since $\mathbf{I}[f] = \sum_{\boldsymbol{k}} \mathbf{W}^{\boldsymbol{k}}[f] \cdot |\boldsymbol{k}| = \sum_{i=1}^{d} \sum_{\boldsymbol{k}} \mathbf{W}^{\boldsymbol{k}}[f] \cdot \boldsymbol{k}_i$, we can prove Theorem 2 by establishing the following two generalizations of Theorems 4 and 5:

Theorem 7. *Let $f\colon \{-1,1\}^{V_1 \cup \cdots \cup V_d} \to \{-1,1\}$ be invariant under permutations of the coordinates in V_i. Then $\sum_{\boldsymbol{k}} \mathbf{W}^{\boldsymbol{k}}[f] \log \binom{n_i}{\boldsymbol{k}_i} \le C_1 \cdot \sum_{\boldsymbol{k}} \mathbf{W}^{\boldsymbol{k}}[f] \cdot \boldsymbol{k}_i$, where $C_1 = \frac{1}{\ln 2}(1 + \frac{4\sqrt{2e}}{\sqrt{\pi}}) \le 9.04$.*

Theorem 8. *Let $f\colon \{-1,1\}^{V_1 \cup \cdots \cup V_d} \to \{-1,1\}$ be any function, not necessarily with any symmetries. Then $\sum_{\boldsymbol{k}} \mathbf{W}^{\boldsymbol{k}}[f] \log \frac{1}{\mathbf{W}^{\boldsymbol{k}}[f]} \le (3 + \log d) \cdot \mathbf{I}[f]$.*

Proof. (Theorem 7.) We assume $i = d$ without loss of generality and write $\overline{V_d} = V_1 \cup \cdots \cup V_{d-1}$. For $y \in \{-1,1\}^{\overline{V_d}}$ we define $f_y\colon \{-1,1\}^{V_d} \to \{-1,1\}$ by $f_y(z) = f(y,z)$; the function f_y is symmetric for each y by assumption. Applying Theorem 4 to each f_y and then taking expectations we obtain

$$\sum_{k'=0}^{n_d} \mathop{\mathbf{E}}_{y}\big[\mathbf{W}^{k'}[f_y]\big] \log \binom{n_d}{k'} \le C_1 \cdot \sum_{k'=0}^{n_d} \mathop{\mathbf{E}}_{y}\big[\mathbf{W}^{k'}[f_y]\big] \cdot k'. \tag{5}$$

Now, $\mathop{\mathbf{E}}_{y}\big[\mathbf{W}^{k'}[f_y]\big] = \sum_{\substack{S \subseteq V_d \\ |S| = k'}} \mathop{\mathbf{E}}_{y}\big[\widehat{f_y}(S)^2\big] = \sum_{\substack{S \subseteq V_d \\ |S| = k'}} \sum_{T \subseteq \overline{V_d}} \widehat{f}(T \cup S)^2 = \sum_{\boldsymbol{k}\colon \boldsymbol{k}_d = k'} \mathbf{W}^{\boldsymbol{k}}[f]$,

where the middle equality is an easy exercise using Parseval. Substituting this into (5) yields the desired inequality.

As for Theorem 8, its proof is essentially identical to that of Theorem 5, using the following generalization of Proposition 3:

Proposition 3. *Let $\boldsymbol{K}$ be a random variable supported on $\mathbb{N}^d \setminus \{\mathbf{0}\}$ and write $L = |\boldsymbol{K}|$. Then $\boldsymbol{H}[\boldsymbol{K}] \le (1 + \log d)\, \mathbf{E}[L]$.*

Proof. Using the chain rule for entropy as well as Proposition 1, we have

$$\boldsymbol{H}[\boldsymbol{K}] = \boldsymbol{H}[L] + \boldsymbol{H}[\boldsymbol{K} \mid L] \le \mathbf{E}[L] + \sum_{\ell=1}^{\infty} \mathbf{Pr}[L = \ell] \cdot \boldsymbol{H}[\boldsymbol{K} \mid L = \ell]. \tag{6}$$

Given $L = \ell$ there are precisely $\binom{\ell+d-1}{d-1}$ possibilities for $\boldsymbol{K}$; hence

$$\boldsymbol{H}[\boldsymbol{K} \mid L = \ell] \le \log \binom{\ell + d - 1}{d - 1} = \ell \cdot \frac{\log \binom{\ell+d-1}{d-1}}{\ell} \le \ell \cdot \frac{\log \binom{1+d-1}{d-1}}{1} = \ell \cdot \log d. \tag{7}$$

The second inequality here follows from the technical Lemma 2 below. The proof is completed by substituting (7) into (6).

Here we give the technical lemma used in the previous proof.

Lemma 2. *For each $c \in \mathbb{N}$ the function $\frac{1}{\ell} \log \binom{\ell+c}{c}$ is a decreasing function of ℓ on $\mathbb{Z}_+$.*

Proof. We wish to show that for all $\ell \in \mathbb{Z}_+$,

$$\tfrac{1}{\ell} \ln \tbinom{\ell+c}{c} \ge \tfrac{1}{\ell+1} \ln \tbinom{\ell+1+c}{c} \Leftrightarrow (\ell+1) \ln \tbinom{\ell+c}{c} \ge \ell \ln \tbinom{\ell+1+c}{c} \Leftrightarrow \tbinom{\ell+c}{c}^{\ell+1} \ge \tbinom{\ell+1+c}{c}^{\ell}$$
$$\Leftrightarrow \tbinom{\ell+c}{c} \ge \left[\tbinom{\ell+1+c}{c} / \tbinom{\ell+c}{c} \right]^{\ell} \quad \Leftrightarrow \tbinom{\ell+c}{c} \ge (1 + \tfrac{c}{\ell+1})^{\ell}.$$

This last inequality is easily shown by induction on ℓ: if one multiplies both sides by $(1 + \frac{c}{\ell+1})$ one obtains $\binom{\ell+1+c}{c} \ge (1 + \frac{c}{\ell+1})^{\ell+1}$ which exceeds $(1 + \frac{c}{\ell+2})^{\ell+1}$ as required for the induction step.

4 Closing Remarks

As the general Fourier Entropy–Influence Conjecture seems difficult to resolve, one may try to prove it for additional interesting classes of Boolean functions. For example, Wan has suggested linear threshold functions as a possibly tractable case [22]. One may also try tackling the Fourier Min-Entropy–Influence Conjecture first (or the slightly stronger consequence that $\exists S \neq \emptyset$ s.t. $\widehat{f}(S)^2 \ge 2^{-C \cdot \mathbf{I}[f]/\mathbf{Var}[f]}$). However this will already likely require stronger tools than the ones used in this paper. The reason is that as mentioned in Section 1.1, this conjecture implies the KKL Theorem, and there is no known proof of KKL which avoids the Hypercontractive or Logarithmic Sobolev Inequalities.

Acknowledgments. The authors would like to thank Rocco Servedio, Li-Yang Tan, and Andrew Wan for sharing their insights on the Entropy–Influence Conjecture.

References

1. Ben-Or, M., Linial, N.: Collective coin flipping. In: Micali, S. (ed.) Randomness and Computation. Academic Press, New York (1990)
2. Bernstein, A.: Maximally connected arrays on the n-cube. SIAM Journal on Applied Mathematics, 1485–1489 (1967)
3. Boppana, R.: The average sensitivity of bounded-depth circuits. Information Processing Letters 63(5), 257–261 (1997)

4. De, A., Etesami, O., Trevisan, L., Tulsiani, M.: Improved pseudorandom generators for depth 2 circuits. In: Proceedings of the 14th Annual International Workshop on Randomized Techniques in Computation, pp. 504–517 (2010)
5. Feller, W.: An introduction to probability theory and its applications, 3rd edn., vol. 1. Wiley, Chichester (1968)
6. Friedgut, E., Kalai, G.: Every monotone graph property has a sharp threshold. Proceedings of the American Mathematical Society 124(10), 2993–3002 (1996)
7. Gopalan, P., Kalai, A., Klivans, A.: Agnostically learning decision trees. In: Proceedings of the 40th Annual ACM Symposium on Theory of Computing, pp. 527–536 (2008)
8. Gopalan, P., Kalai, A., Klivans, A.: A query algorithm for agnostically learning DNF? In: Proceedings of the 21st Annual Conference on Learning Theory, pp. 515–516 (2008)
9. Gross, L.: Logarithmic Sobolev inequalities. American Journal of Mathematics 97(4), 1061–1083 (1975)
10. Harper, L.: Optimal assignments of numbers to vertices. Journal of the Society for Industrial and Applied Mathematics 12(1), 131–135 (1964)
11. Kahn, J., Kalai, G., Linial, N.: The influence of variables on Boolean functions. In: Proceedings of the 29th Annual IEEE Symposium on Foundations of Computer Science, pp. 68–80 (1988)
12. Kalai, G.: The entropy/influence conjecture. Posted on Terence Tao's What's new blog (2007), http://terrytao.wordpress.com/2007/08/16/gil-kalai-the-entropyinfluence-conjecture/
13. Keilson, J., Gerber, H.: Some results for discrete unimodality. Journal of the American Statistical Association 66(334), 386–389 (1971)
14. Klivans, A., Lee, H., Wan, A.: Mansours Conjecture is true for random DNF formulas. Electronic Colloquium on Computational Complexity TR10-023 (2010)
15. Mansour, Y.: Learning boolean functions via the Fourier Transform. In: Roychowdhury, V., Siu, K.-Y., Orlitsky, A. (eds.) Theoretical Advances in Neural Computation and Learning, ch. 11, pp. 391–424. Kluwer Academic Publishers, Dordrecht (1994)
16. Mansour, Y.: An $O(n^{\log \log n})$ learning algorithm for DNF under the uniform distribution. Journal of Computer and System Sciences 50(3), 543–550 (1995)
17. O'Donnell, R.: Some topics in analysis of boolean functions. In: Proceedings of the 40th Annual ACM Symposium on Theory of Computing, pp. 569–578 (2008)
18. O'Donnell, R., Saks, M., Schramm, O., Servedio, R.: Every decision tree has an influential variable. In: Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science, pp. 31–39 (2005)
19. Steiglitz, K., Bernstein, A.: Optimal binary coding of ordered numbers. Journal of the Society for Industrial and Applied Mathematics 13(2), 441–443 (1965)
20. Talagrand, M.: On Russo's approximate zero-one law. Annals of Probability 22(3), 1576–1587 (1994)
21. Titsworth, R.: Correlation properties of cyclic sequences. PhD thesis, California Institute of Technology (1962)
22. Wan, A.: Talk at the Center for Computational Intractability (2010)

Nonmonotone Submodular Maximization via a Structural Continuous Greedy Algorithm (Extended Abstract)

Moran Feldman*, Joseph (Seffi) Naor**, and Roy Schwartz

Computer Science Dept., Technion, Haifa, Israel
{moranfe,naor,schwartz}@cs.technion.ac.il

Abstract. Consider a suboptimal solution S for a maximization problem. Suppose S's value is small compared to an optimal solution OPT to the problem, yet S is structurally similar to OPT. A natural question in this setting is whether there is a way of improving S based solely on this information. In this paper we introduce the *Structural Continuous Greedy Algorithm*, answering this question affirmatively in the setting of the Nonmonotone Submodular Maximization Problem. We improve on the best approximation factor known for this problem. In the Nonmonotone Submodular Maximization Problem we are given a non-negative submodular function f, and the objective is to find a subset maximizing f. Our method yields an 0.42-approximation for this problem, improving on the current best approximation factor of 0.41 given by Gharan and Vondrák [5]. On the other hand, Feige et al. [4] showed a lower bound of 0.5 for this problem.

1 Introduction

Consider a situation where one has a suboptimal solution S for a maximization problem. The most natural measure for the quality of S is its value with respect to the objective function. However, an additional measure is the structural similarity of S to an optimal solution OPT of the problem. We are interested in the relation between these two measures. Suppose S is structurally similar to OPT, yet its value is small. We call problems in which such sets exist *uncorrelated objective-structure problems*. For such problems it is possible for an algorithm to produce a solution which is structurally very similar to OPT, and yet is a poor approximation with respect to values. For example, in the Max Cut problem, suppose the input is a star. If we put all vertices on one side of the cut, we get a solution of value 0 which is only different from OPT in one vertex.

For every uncorrelated objective-structure problem $\mathcal{P}$, a natural question is whether there is an efficient algorithm that, given a solution S structurally similar to OPT, produces a solution S' of large value; i.e., we look for an algorithm

* Moran Feldman is a recipient of the Google Europe Fellowship in Market Algorithms, and this research is supported in part by this Google Fellowship.
** This work was partly supported by ISF grant 1366/07.

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 342–353, 2011.

trading structural similarity to OPT for value. If we find such an algorithm for problem $\mathcal{P}$, then we can improve approximation algorithms for $\mathcal{P}$ which produce (directly, or indirectly) solutions that are structurally similar to OPT.

In this paper we suggest the *structural continuous greedy* algorithm trading structural similarity to OPT for value in the context of the Nonmonotone Submodular Maximization Problem (NSM). Given a ground set $\mathcal{E}$, a function $f : 2^{\mathcal{E}} \rightarrow \mathbb{R}$ is called *submodular* if for every $A, B \subseteq \mathcal{E}$, $f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$. An instance of NSM consists of a non-negative submodular function f, and its objective is to find a subset of $\mathcal{E}$ maximizing f. An explicit representation of f might be exponential in the size of $\mathcal{E}$. Hence, we assume the *value oracle* model in which the algorithm has access to an oracle that given a set S returns $f(S)$. This model is probably the most standard one, and is common throughout the literature.

To see why NSM is indeed an uncorrelated objective-structure problem, observe the following example. Consider the groundset $\mathcal{E} = [n] \times \{a, b\}$, and the submodular function: $f(S) = \sum_{i=1}^{n} g(S \cap \{(i, a), (i, b)\})$, where g is an indicator function for the event that its argument is a set of size 1. The set $\mathcal{E}' = [n/2] \times \{a, b\}$ has $f(\mathcal{E}') = 0$ although it contains half of the elements of every optimal solution.

In the next section we describe the structural continuous greedy algorithm. We need a few well known terms, which we define here for completeness. Given a function f, if $f(\varnothing) = 0$ we say that f is *normalized*, and if for every $A \subseteq B \subseteq \mathcal{E}$, $f(A) \leq f(B)$ (respectively, $f(A) \geq f(B)$), we say that f is *monotone* (respectively, *down monotone*). Also, we use the notation of OPT throughout the article to denote an optimal solution to the problem in question. If there are multiple optimal solutions, OPT denotes an arbitrary fixed one.

1.1 The Structural Continuous Greedy Algorithm

Let us formally define the structural similarity of a set S to OPT. One natural definition is $f(S \cap OPT)$. This definition makes sense for a monotone f. However, when f may be nonmonotone, we need a definition that also captures structural similarity due to elements outside of OPT missing also from S. Thus, we define the structural similarity of a set S to OPT as $f(S \cap OPT) + f(S \cup OPT)$.[1]

Consider a set S. Removing all elements of $S - OPT$ changes the value of this set to $f(S \cap OPT)$; hence, if $f(S \cap OPT) > f(S)$, removing all the elements of $S - OPT$ from S increases the objective function. By submodularity, the last property also implies that there is at least one element in $S - OPT$ whose removal from S increases its value. Putting it all together, $f(S \cap OPT) > f(S)$ implies the existence of an element $e \in S - OPT$ such that $f(S - \{e\}) > f(S)$. Similarly, $f(S \cup OPT) > f(S)$ implies the existence of an element $e \in OPT - S$ such that $f(S \cup \{e\}) > f(S)$. We can now relate our observations to the definition of structural similarity: if $f(S \cap OPT) + f(S \cup OPT) > 2f(S)$, then the value of S can be improved by adding or removing a single element from it.

[1] The structural similarity is defined with an arbitrary optimal solution (in case there are multiple options). Hence, the continuous greedy algorithm is guaranteed to improve a set if it is structurally similar to any optimal solution.

The last conclusion seems to suggest that given a set S which is structurally similar to OPT, and yet has a low value, S can be improved using a local search algorithm seeking at every stage to improve S by adding or removing a single element. However, as this algorithm adds and removes elements, the structural similarity to OPT might decrease; and we do not know how to relate this decrease to the increase in $f(S)$. Therefore, although we can conclude that such a local search algorithm will improve S, it is not clear how to give any positive lower bound on this improvement.

To work around this difficulty, we propose the structural continuous greedy algorithm. This algorithm maintains a fractional set (i.e., a set in which every element appears independently with some probability). Initially, every element of the input set S appears in the fractional set S' with probability 1, and every other element appears in it with probability 0. In every iteration the probability of each element can be changed by a small δ. The probability of an element is increased if doing so increases the expected value of the fractional set, assuming all other probabilities are unchanged; and decreased otherwise.

Using arguments similar to those above, we can lower bound the improvement in every iteration of the structural continuous greedy algorithm in terms of $\mathbb{E}[f(S' \cap OPT) + f(S' \cup OPT) - 2f(S')]$. Hence, as long as S' is structurally similar to OPT, in expectation, the structural continuous greedy is guaranteed to improve it by changing the probabilities of elements. Moreover, after m iterations, the change in the structural similarity of S' to OPT can be upper bounded using the following observation: every element of the original set S appears in S' with probability at least $1 - m\delta$, and every element outside of S appears in S' with probability at most $m\delta$; which intuitively means that S' is quite similar (structurally) to S, and therefore, also to OPT.

The structural continuous greedy inherits much of its structure from an algorithm of Vondrák [16] called "continuous greedy"; however, the two algorithms are analyzed very differently. The roots of [16]'s continuous greedy algorithm can be traced back to Wolsey [18], who also gives an algorithm called "continuous greedy", although, the two "continuous greedy" algorithms share only vague general ideas. The continuous greedy algorithm of [16] starts with an empty set, and then only increases the probabilities of elements. This is fine since [16] deals with monotone submodular functions. On the other hand, the structural continuous greedy, introduced in this paper, works with nonmonotone functions, which requires both modifications of the algorithm and its analysis (with respect to the algorithm and analysis of [16]).

There are two differences between our structural continuous greedy algorithm, and the "continuous greedy" algorithm of [16]. First, the initial point of the structural continuous greedy algorithm is an arbitrary set, and second, it may both *increase* and *decrease* probabilities. Though these differences are quite minor, the analysis of the two algorithms is completely different. The continuous greedy of [16] constructs a solution starting with an empty set, and its analysis

strongly uses the monotonicity of f (which implies that increasing the probabilities of OPT's elements is always good). On the other hand, the structural continuous greedy improves existing sets, and its analysis is based on the structural similarity of the input to OPT.

1.2 Taking Advantage of the Structural Continuous Greedy

The structural continuous greedy algorithm can improve sets that are already structurally similar to OPT (at the cost of possibly decreasing their structural similarity to OPT). However, it is not clear, at first glance, how this can be used to produce an approximation algorithm for NSM. In order to answer this question, we have to understand the details of the known algorithms for NSM.

The first constant factor approximation algorithms for NSM were given by Feige et al. [4]. The simplest algorithm they provide simply selects every element of the ground set with probability 1/2. Feige et al. [4] showed that this simple algorithm already achieves a 1/4-approximation. They also suggested a local search algorithm which basically starts with an arbitrary set and adds or removes single elements as long as this improves the value of the set. The output set of this algorithm is called *locally optimal* because it cannot be improved by adding or removing a single element. In [4], it is shown that any locally optimal set gives a 1/3-approximation for NSM.

The problem with local search algorithms is that they tend to get "stuck" in local optima. One way to get around that is to add some noise to the system. For that purpose, [4] defines for every set S, a random set $R(S)$ containing every element of S with probability 2/3 and every other element with probability 1/3 (in other words, to get $R(S)$, we start with S and switch the state of every element, from being in S to being outside of it or vice versa, with probability 1/3). The local search algorithm now tries to find a set maximizing $\mathbb{E}[f(R(S))]$. Somewhat surprisingly, this randomized local search has an improved approximation ratio of 2/5. Looking more carefully at the analysis of [4], it actually shows that if S is a locally optimal set (with respect to the last algorithm), then:

$$\mathbb{E}[f(R(S))] + \frac{f(\bar{S})}{9} \geq \frac{4f(OPT)}{9} \ .$$

This inequality guarantees that the expected values of $\max\{f(\bar{S}), f(R(S))\}$ is at least $0.4f(OPT)$. Now, observe that $f(S \cap OPT) + f(\bar{S} \cap OPT) \geq f(OPT)$, and $f(S \cup OPT) + f(\bar{S} \cup OPT) \geq f(OPT)$. This suggests a negative correlation between the structural similarity to OPT of S and $\bar{S}$ in the following sense: if one of them is structurally far from OPT, then the other one must be structurally close to OPT. Since $R(S)$ is basically S with some noise, we also get a negative correlation between the structural similarity to OPT of $R(S)$ and $\bar{S}$. Hence, if $R(S)$ has a low expected value, and it is too far from OPT for the structural continuous greedy algorithm to improve it significantly, then $\bar{S}$ must have both:

significant value and structural similarity to OPT; hence, running the structural continuous greedy algorithm on it is guaranteed to produce a good set.

Using the above observations, we can get a 0.413-approximation, which already slightly improves on the state of the art algorithm of Gharan and Vondrack [5] that gives a 0.41-approximation (we defer details to a full version of this paper). However, we can do better if we use the structural continuous greedy algorithm together with [5]'s algorithm, though the combination of these two techniques requires some work. The algorithm of [5] is a *simulated annealing* algorithm. It starts as a local search algorithm with a lot of noise (i.e., $R(S)$ is obtained from S by adding or removing every element with high probability), and gradually decreases the noise level. Intuitively, starting with a lot of noise should help the algorithm avoid inferior local maxima.

Simulated annealing algorithms are common in practice, but they often turn out to be very hard to analyze. The analysis of [5] works as following. For some, relatively high, level of noise p_0, the algorithm is analyzed as a noisy local search algorithm, producing a lower bound on the value of the algorithm's solution at noise level p_0. Now, observe what happens as the algorithm reduces the noise level. In a locally optimal solution, infinitesimally increasing the probability of every element inside the solution, and decreasing the probability of every element outside of the solution, only improve the solution. This change in the probabilities is exactly equivalent to a reduction in the noise level. Hence, whenever the algorithm reduces the noise level, since the current solution is locally optimal (the algorithm does not reduce the noise level unless this is the case), the reduction in the noise level can only improve the solution.

The main observation of [5] is a way to show a *positive* lower bound on the improvement achieved when the noise is reduced. This lower bound is negatively correlated with $f(\bar{S})$, where S is the current solution of the algorithm. This immediately implies that either the set $\bar{S}$ is good at some point, or the value of S significantly increases as the noise level decreases. Together with the lower bound on the value of S at noise level p_0, an approximation guarantee follows.

The above description does not give an obvious way to combine [5]'s algorithm with the structural continuous greedy. In order for the structural continuous greedy algorithm to be useful, we must find sets in the proof of [5]'s algorithm with the following properties.

- Structurally similar to OPT, at least when [5]'s algorithm behaves poorly.
- Has large value, again, at least when [5]'s algorithm behaves poorly.

If we find such a set, it can be improved using the structural continuous greedy algorithm, and then be used as an alternative solution. In other words, whenever the original algorithm behaves poorly, the structural continuous greedy produces a good alternative solution, which results in an improved approximation ratio.

To this end, we make the following observation. The bound, given by [5], on the improvement achieved when the noise is reduced is actually positively correlated with $f(S \cap OPT) + f(S \cup OPT)$. The negative correlation of the bound with $f(\bar{S})$ follows since, by submodularity, $f(\bar{S}) \geq f(OPT) - f(S \cap OPT) - f(S \cup OPT)$. Another inequality that follows from f's submodularity

is $f(\bar{S} \cap OPT) + f(\bar{S} \cup OPT) \geq 2f(OPT) - f(S \cap OPT) - f(S \cup OPT)$. Hence, the lower bound is also negatively correlated with the structural similarity of $\bar{S}$ to OPT. In conclusion, if at some point the value of the algorithm's solution does not improve fast enough, then at that point $\bar{S}$ has two properties: it has a relatively high value and it is structurally similar to OPT. Therefore, applying the structural continuous greedy algorithm to $\bar{S}$ is guaranteed to produce a good set. This is the intuition behind our 0.42 approximation for NSM.

We note [4] showed that no algorithm making a polynomial number of oracle queries gives better than 1/2-approximation for NSM. This hardness result holds even when f is symmetric, i.e., $f(S) = f(\mathcal{E} - S)$, in which case it is tight [4].

1.3 Related Work

It is well known that submodular functions can be minimized in polynomial time [14]. However, maximizing a submodular function turns out to be much more difficult. Maximization of nonnegative submodular functions generalizes NP-hard problems such as Max-Cut [6], and [4] proved it impossible to give better than 0.5-approximation for it using a polynomial number of oracle queries. Related problems ask to maximize a nonnegative submodular function subject to various combinatorial constraints on the sets we are allowed to choose [7,11,12,17] or minimize a submodular function subject to such constraints [15,8,9].

Another line of work deals with maximizing normalized monotone submodular functions, again, subject to various combinatorial constraints. A continuous greedy algorithm was given by Calinescu et al. [1] for maximizing a normalized monotone submodular function subject to a matroid constraint. Later, Lee et al. [12] gave a local search algorithm achieving $1/p - \varepsilon$ approximation for maximizing such functions subject to intersection of p matroids. Kulik et al. [10] showed a $1 - 1/e - \varepsilon$ approximation algorithm for maximizing a normalized monotone submodular function subject to multiple knapsack constraints. Recently, Chekuri et al. [3], gave a nonmonotone counterpart of the continuous greedy algorithm of [16]. This result improves several nonmonotone submodular optimization problems. Some of the above results were generalized by Chekuri et al. [2], which gives a dependent rounding for various polytopes, including matroid and matroid-intersection polytops. The advantage of this rounding technique is that it guarantees strong concentration bounds for submodular functions.

2 Preliminaries

Formally, in NSM, we are given a nonnegative submodular function $f : 2^{\mathcal{E}} \rightarrow \mathbb{R}^+$. The objective is to find a subset $S \subseteq \mathcal{E}$ maximizing $f(S)$. We denote $|\mathcal{E}|$ by n. In all algorithms presented, we assume n is larger than any given constant (if this is not the case, one can solve NSM optimally using an exhaustive search).

There are two well known extensions of a submodular function $f : 2^{\mathcal{E}} \rightarrow \mathbb{R}^+$ to the hypercube $[0,1]^{\mathcal{E}}$. The first one is $F(x) : [0,1]^{\mathcal{E}} \rightarrow \mathbb{R}^+$, the *multilinear extension* introduced by [1]. For a given vector $z \in [0,1]^{\mathcal{E}}$, let $R(z)$ be a set containing every element $e \in E$ with probability z_e, independently. The multilinear

extension of f is defined as $F(z) = \mathbb{E}[R(z)]$. This extensions is called multilinear because it can also be written as $F(z) = \sum_{S \subseteq \mathcal{E}} \left(\prod_{e \in S} z_e \cdot \prod_{e \notin S} (1 - z_e) \cdot f(S) \right)$.

The other known extension of submodular functions is the Lovász extension introduced in [13]. Define $T_\lambda(z)$ to be the set of elements whose coordinate in z is at least λ. The Lovász extension of a submodular function $f : 2^{\mathcal{E}} \to \mathbb{R}^+$ is defined as $\hat{f}(z) = \int_0^1 f(T_\lambda(z)) d\lambda$. This definition can also be interpreted in probabilistic terms as the expected value of f over the set $T_\lambda(z)$, where λ is uniformly selected from the range $[0, 1]$. In this paper, the Lovász extension is used only to lower bound the multilinear extension. This is done using the following theorem.

Theorem 1 (Lemma A.4 in [17]). *Let $F(z)$ and $\hat{f}(z)$ be the multilinear and Lovász extensions of a submodular function f, respectively. Then, for every $z \in [0, 1]^{\mathcal{E}}$, $F(z) \geq \hat{f}(z)$.*

We abuse notation, and write $F(z \cup S)$ to denote $\mathbb{E}[f(R(z) \cup S)]$, $F(z - S)$ to denote $\mathbb{E}[f(R(z) - S)]$ and so on. We also use the shorthand $\partial_e F(z)$ for $F(z \cup e) - F(z - e)$. The multilinear nature of F yields the following useful observation which relates these notations.

Observation 1. *Let $F(z)$ be the multilinear extension of a submodular function $f : 2^{\mathcal{E}} \to R^+$. Then, for every $e \in \mathcal{E}$, $\partial_e F(z) = \frac{F(z \cup e) - F(z)}{1 - z_e} = \frac{F(z) - F(z - e)}{z_e}$, assuming the denominators are not 0.*

The following lemma shows that the change in $F(z)$ induced by small modifications to the coordinates of z is almost equal to the sum of changes that would have resulted from modifying each coordinate independently. This kind of lemmata are standard, hence, we omit its proof from this extended abstract.

Lemma 1. *Consider two vectors $z, z' \in [0, 1]^{\mathcal{E}}$ such that for every $e \in \mathcal{E}$, $|z_e - z'_e| \leq \delta$, for $\delta \leq n^{-3}$. Then, $F(z') - F(z) \geq \sum_{e \in \mathcal{E}} (z'_e - z_e) \cdot \partial_e F(z) - O(n^{-1}\delta)$.*

Given a set $S \subseteq E$ and a number $p \in [0.5, 1]$, let $z^p(S)$ be a vector of $[0, 1]^{\mathcal{E}}$, containing p in the coordinate of every element of S, and $1 - p$ in all other coordinates. Since the structural similarity of S to OPT is often used in our proofs, we denote it by $V(S)$, i.e., $V(S) = f(S \cap OPT) + f(S \cup OPT)$. For simplicity of the exposition we assume $f(OPT) = 1$. This assumption removes some clattering from the mathematical calculations, and can be easily removed.

3 The Structural Continuous Greedy Algorithm

Following is the structural continuous greedy algorithm. This algorithm improves a set S which has some structural resemblance to OPT (at the cost of possibly making the set less structurally similar to OPT).

Structural Continuous Greedy $(f, S, \mathcal{E})$**:**
1. Let $\delta = n^{-3}$. Initialize $t = 0$ and $z_0 = 1_S$.
2. While $t < 1$ do:
3. For every element $e \in \mathcal{E}$ do:
4. If $\partial_e F(z_t) > 0$,[a] $(z_{t+1})_e = \min\{1, (z_t)_e + \delta\}$.
5. If $\partial_e F(z_t) < 0$, $(z_{t+1})_e = \max\{0, (z_t)_e - \delta\}$.
6. $t \leftarrow t + \delta$
7. Return $R(z_1)$

[a] The algorithm does not have access to $\partial_e F(z_t)$, however, it can approximate it up to any required accuracy by averaging independent random samples. This is a standard practice, and we omit details (e.g., see [1]). Taking the error induced by the random sampling into account effects our results, with high probability, only by a lower order term.

We prove in this section the following theorem.

Theorem 2. *Assuming $V(S) \geq 2f(S)$, the expected value of the solution produced by the Structural Continuous Greedy Algorithm outputs is at least $\frac{V(S)}{4} \cdot \left[2 - \ln\left[3 - \frac{4f(S)}{V(S)}\right]\right] - O(n^{-1})$.*

We define a few sets of elements that we will refer to. Let $\mathcal{E}_t^0$ (resp. $\mathcal{E}_t^1$) be the set of elements whose coordinates in z_t are 0 (resp. 1), $\mathcal{E}_t^+$ be the set $\{e \in \mathcal{E} | \partial_e F(z_t) \geq 0\}$, and $\mathcal{E}_t^-$ be $\mathcal{E} - \mathcal{E}_t^+$.

The following lemma shows that there is a set of elements for which the gain achieved by changing the coordinate of each one of them independently (i.e., when changing one coordinate, the others are kept unchanged) is significant.

Lemma 2. *At every time t, $\sum_{e \in OPT \cap (\mathcal{E}_t^+ - \mathcal{E}_t^1)} \partial_e F(z_t) - \sum_{e \in \overline{OPT} \cap (\mathcal{E}_t^- - \mathcal{E}_t^0)} \partial_e F(z_t) \geq F(OPT \cap z_t) + F(OPT \cup z_t) - 2F(z_t)$.*

Proof. Observe that $\sum_{e \in OPT \cap (\mathcal{E}_t^- - \mathcal{E}_t^1)} F(z_t \cup e) - F(z_t) = \sum_{e \in OPT \cap (\mathcal{E}_t^- - \mathcal{E}_t^1)} (1 - (z_t)_e) \cdot \partial_e F(z_t) \leq 0$. Using this observation, we get:

$$\begin{aligned} \sum_{e \in OPT \cap (\mathcal{E}_t^+ - \mathcal{E}_t^1)} \partial_e F(z_t) &\geq \sum_{e \in OPT \cap (\mathcal{E}_t^+ - \mathcal{E}_t^1)} F(z_t \cup e) - F(z_t) \qquad (1) \\ &\geq \sum_{e \in OPT - \mathcal{E}_t^1} F(z_t \cup e) - F(z_t) \\ &\geq F(z_t \cup (OPT - \mathcal{E}_t^1)) - F(z_t) = F(z_t \cup OPT) - F(z_t) \ . \end{aligned}$$

Analogously, we can also get:

$$\sum_{e \in \overline{OPT} \cap (\mathcal{E}_t^- - \mathcal{E}_t^0)} \partial_e F(z_t) \leq F(z_t) - F(z_t - \overline{OPT}) \ . \qquad (2)$$

The lemma now follows by combining (1) and (2).

Together with Lemma 1, Lemma 2 shows that there is a set of elements whose coordinates can be changed together to produce a significant improvement. The following lemma shows that the algorithm indeed finds such a set.

Lemma 3. $F(z_{t+\delta}) - F(z_t) \geq \delta[F(OPT \cap z_t) + F(OPT \cup z_t) - 2F(z_t)] - O(\delta n^{-1})$.

Proof. The algorithm increases the coordinates of all elements in $\mathcal{E}_t^+ - \mathcal{E}_t^1$ by δ, and decreases the coordinates of all elements in $\mathcal{E}_t^- - \mathcal{E}_t^0$ by δ. Hence, by Lemma 1, $F(z_{t+\delta}) - F(z_t) \geq \delta \sum_{e \in \mathcal{E}_t^+ - \mathcal{E}_t^1} \partial_e F(z) - \delta \sum_{e \in \mathcal{E}_t^- - \mathcal{E}_t^0} \partial_e F(z) - O(n^{-1}\delta)$. The lemma now follows by combining this inequality with Lemma 2.

The following lemma lower bounds the similarity of $R(z_t)$ to OPT in terms of the similarity of the original set S to OPT and the time that passed so far.

Lemma 4. *For every time t, the following two inequalities hold:*

- $F(z_t \cap OPT) \geq (1-t) \cdot f(S \cap OPT)$.
- $F(z_t \cup OPT) \geq (1-t) \cdot f(S \cup OPT)$.

Proof. Clearly, for every $e \in OPT$, $f(OPT) \geq f(OPT - e)$. By submodularity, this implies that f is monotone on subsets of OPT. Observe that $f(X \cap OPT)$ is also a submodular function. Thus, by applying Theorem 1 to it, we get

$$F(z_t \cap OPT) \geq \int_0^1 f(T_\lambda(z_t) \cap OPT) d\lambda \geq \int_0^{1-t} f(T_\lambda(z_t) \cap OPT) d\lambda \ .$$

At time t, every element $e \in S$ must have $(z_t)_e \geq 1-t$, hence, the set $T_\lambda(z_t)$ in the integrand must contain S. Plugging this observation into the previous inequality, and using the monotonicity of f over subsets of OPT, we get $F(z_t \cap OPT) \geq \int_0^{1-t} f(S \cap OPT) d\lambda = (1-t) \cdot f(S \cap OPT)$. The other inequality guaranteed by the lemma is proved analogously.

Plugging Lemma 4 into Lemma 3, we get the following lower bound on the improvement made by the algorithm at each step.

Corollary 1. $F(z_{t+\delta}) - F(z_t) \geq \delta[(1-t)[f(S \cap OPT) + f(S \cup OPT)] - 2F(z_t)] - O(n^{-1}\delta) = \delta[(1-t)V(S) - 2F(z_t)] - O(n^{-1}\delta)$.

Let g be the solution of the following recursive formula. $g(t+\delta) - g(t) = \delta[(1-t)V(S) - 2g(t)]$, with the boundary condition $g(0) = f(S)$. Lemmata 5 and 6 prove that it is sufficient to show that at some point $g(t) \geq X$ in order to prove $F(z_1) \geq X - O(n^{-1})$.

Lemma 5. *For evert time t, $F(z_t) \geq g(t) - O(n^{-1}t)$.*

Proof. Let c be the constant hiding behind the big O in Corollary 1. We prove $F(z_t) \geq g(t) - cn^{-1}t$ by induction. For $t = 0$, the claim holds since $F(z_0) = f(S) = g(0)$. Assume the claim holds for t, let us prove it for $t+\delta$ via Corollary 1.

$$\begin{aligned} F(z_{t+\delta}) &\geq (1-t)\delta V(S) + (1-2\delta)F(z_t) - cn^{-1}\delta \\ &\geq (1-t)\delta V(S) + (1-2\delta)g(t) - cn^{-1}(\delta + t) = g(t+\delta) - cn^{-1}(\delta + t) \ . \end{aligned}$$

Lemma 6. *For every time t, $F(z_1) \geq F(z_t) - O(n^{-1})$.*

Proof. The algorithm increases only coordinates of elements with a positive $\partial_e F(z_t)$, and decreases only coordinates of elements with a negative $\partial_e F(z_t)$. Hence, by Lemma 1, $F(z_t)$ decreases by at most $O(n^{-1}\delta)$ in every time step. Since there are δ^{-1} time steps, the lemma follows.

Let h be the function $h(t) = \frac{3}{4}V(S) \cdot \left[1 - \frac{2t}{3} - e^{-2t}\right] + e^{-2t} \cdot f(S)$. Observe that $dh/dt = (1-t)V(S) - 2h(t)$. The next lemma shows that h lower bounds g (given some condition), and therefore, also lower bounds the value of the algorithm's solution.

Lemma 7. *If for every $t' \leq t$, $g(t') \leq 0.5(1-t)V(S)$, then $g(t) \geq h(t)$.*

Proof. We prove the lemma by induction on t. For $t = 0$ the lemma follows from the definition of $h(0)$. Assume the lemma holds for some time t, let us prove it for time $t + \delta$. Let t_1 be the last time in the range $[t, t+\delta]$ in which $h(t_1) \leq g(t)$.

$$\begin{aligned} h(t+\delta) &= h(t_1) + \int_{t_1}^{t+\delta} h'(\tau)d\tau = h(t_1) + \int_{t_1}^{t+\delta} \left((1-\tau)V(S) - 2h(\tau)\right) d\tau \\ &\leq g(t) + \int_{t_1}^{t+\delta} \left((1-t)V(S) - 2g(t)\right) d\tau \\ &= g(t) + (t+\delta-t_1)\left((1-t)V(S) - 2g(t)\right) \\ &\leq g(t) + \delta\left((1-t)V(S) - 2g(t)\right) = g(t+\delta) \ . \end{aligned}$$

Let $t^* = 0.5 \ln\left[3 - \frac{4f(S)}{V(S)}\right]$. Notice that if $f(S \cup OPT) + f(S \cap OPT) \geq 2f(S)$, then $\ell^* \in [0, 0.5 \ln 3] \subseteq [0, 1]$.

Lemma 8. *Assuming $V(S) \geq 2f(S)$, $h(t^*) = 0.25V(S) \cdot \left[2 - \ln\left[3 - \frac{4f(S)}{V(S)}\right]\right]$.*

Proof.

$$\begin{aligned} h(t^*) &= \frac{3}{4}V(S)\left[1 - \frac{\ln\left[3 - \frac{4f(S)}{V(S)}\right]}{3} - \frac{V(S)}{3V(S) - 4f(S)}\right] + f(S)\frac{V(S)}{3V(S) - 4f(S)} \\ &= \frac{3}{4}V(S)\left[1 - \frac{\ln\left[3 - \frac{4f(S)}{V(S)}\right]}{3}\right] - \frac{V(S)}{4} = \frac{1}{4}V(S)\left[2 - \ln\left[3 - \frac{4f(S)}{V(S)}\right]\right] \ . \end{aligned}$$

We are now ready to prove Theorem 2.

Proof (of Theorem 2). First let us claim that somewhere g gets the value:

$$0.25V(S) \cdot \left[2 - \ln\left[3 - \frac{4f(S)}{V(S)}\right]\right] - O(n^{-3}) \ .$$

If the assumption of Lemma 7 does not hold for t^*, then g gets, somewhere, the value:

$$0.5(1-t^*)V(S) = 0.5\left(1 - 0.5\ln\left[3 - \frac{4f(S)}{V(S)}\right]\right)V(S) \ .$$

Which proves the claim. Hence, we can assume that the assumption of Lemma 7 holds, which implies g gets value of at least $h(\delta \cdot \lfloor t^*/\delta \rfloor)$. In order to lower bound this value, we can use the observation that for every $t \in [0, t^*]$, $h'(t) \leq 2$. Thus,

$$h(\delta \cdot \lfloor t^*/\delta \rfloor) \geq h(t^*) - 2\delta \geq h(t^*) - 2n^{-3} \ .$$

And the theorem now follows from Lemmata 5 and 6.

4 Simulated Annealing Algorithm

Given a set S, let us denote by $C(S)$ the random set resulting from running the structural continuous greedy algorithm on S. Consider the following algorithm.

Simulated Annealing Algorithm $(f, \mathcal{E})$**:**
1. Let $d = 0.752 - \sqrt{2}/(1+\sqrt{2})$ and $\delta = d\lceil n^3 d \rceil^{-1}$.[a]
2. Initialize $p = \sqrt{2}/(1+\sqrt{2})$ and $A_p = \varnothing$.
3. Repeat:
4. Set $B_p \leftarrow A_p$
5. While there exists a set S such that:
(i) $|S \oplus B_p| = 1$
(ii) $F(z^p(S)) > F(z^p(B_t))$
6. Replace B_p with S
7. Set $A_{p+\delta} \leftarrow B_p$
8. Update $p \leftarrow p + \delta$
9. Until $p = 0.752$.
10. Return the best set in $\{R(z^{0.752}(B_{0.752}))\} \cup \{\bar{B}_p, C(\bar{B}_p)\}_{p=\frac{\sqrt{2}}{1+\sqrt{2}}}^{0.752}$

[a] Informally δ is the inverse of a number which is both at least n^3 and dividable by d.

Remark: As written the Simulated Annealing algorithm does not run in polynomial time for two reasons: the number of iterations that the algorithm makes might be exponential, and checking condition (ii) exactly cannot be done in polynomial time using only value oracle access to f. However, both problems can be solved using standard means (see, e.g., [12,1]), at the cost of losing a lower order term in the approximation ratio. We omit the details.

Theorem 3. *The Simulated Annealing Algorithm returns a solution whose expected value is at least* 0.42.

The proof of the theorem combines ideas from the proof of the simulated annealing algorithm of [5] with the new observations described in Section 1.2. Due to lack of space, this proof is deferred to a full version of this paper.

References

1. Calinescu, G., Chekuri, C., Pál, M., Vondrák, J.: Maximizing a monotone submodular function subject to a matroid constraint. To appear in SIAM J. Comput.

2. Chekuri, C., Vondrák, J., Zenklusen, R.: Dependent randomized rounding via exchange properties of combinatorial structures. In: 51st Annual Symposium on Foundations of Computer Science, pp. 575–584. IEEE Computer Society, Washington DC (2010)
3. Chekuri, C., Vondrák, J., Zenklusen, R.: Submodular function maximization via the multilinear relaxation and contention resolution schemes. To appear in the 42nd ACM Symposium on Theory of Computer Science (2011)
4. Feige, U., Mirrokni, V.S., Vondrák, J.: Maximizing non-monotone submodular functions. In: 48th Annual IEEE Symposium on Foundations of Computer Science, pp. 461–471. IEEE Computer Society, Washington DC (2007)
5. Gharan, S.O., Vondrák, J.: Submodular maximization by simulated annealing. In: 22nd ACM-SIAM Symposium on Discrete Algorithms, pp. 1096–1116 (2011)
6. Goemans, M.X., Williamson, D.P.: Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. J. ACM 42(6), 1115–1145 (1995)
7. Gupta, A., Roth, A., Schoenebeck, G., Talwar, K.: Constrained non-monotone submodular maximization: Offline and secretary algorithms. In: Saberi, A. (ed.) WINE 2010. LNCS, vol. 6484, pp. 246–257. Springer, Heidelberg (2010)
8. Iwata, S., Nagano, K.: Submodular function minimization under covering constraints. In: 50th Annual IEEE Symposium on Foundations of Computer Science, pp. 671–680. IEEE Computer Society, Washington DC (2009)
9. Jegelka, S., Bilmes, J.: Cooperative cuts: Graph cuts with submodular edge weights. Technical report, Max Planck Institute for Biological Cybernetics (2010)
10. Kulik, A., Shachnai, H., Tamir, T.: Maximizing submodular set functions subject to multiple linear constraints. In: 20th ACM-SIAM Symposium on Discrete Algorithms, pp. 545–554. Society for Industrial and Applied Mathematics, Philadelphia (2009)
11. Lee, J., Mirrokni, V.S., Nagarajan, V., Sviridenko, M.: Maximizing non-monotone submodular functions under matroid or knapsack constraints. SIAM J. Discrete Mathematics 23(4), 2053–2078 (2010)
12. Lee, J., Sviridenko, M., Vondrák, J.: Submodular maximization over multiple matroids via generalized exchange properties. In: Dinur, I., Jansen, K., Naor, J., Rolim, J. (eds.) APPROX 2009. LNCS, vol. 5687, pp. 244–257. Springer, Heidelberg (2009)
13. Lovász, L.: Submodular functions and convexity. In: Bachem, A., Grötschel, M., Korte, B. (eds.) Mathematical Programming: the State of the Art, pp. 235–257. Springer, Berlin (1983)
14. Lovász, L., Grötschel, M., Schrijver, A.: The ellipsoid method and its consequences in combinatorial optimization. Combinatoria 1(2), 169–197 (1981)
15. Svitkina, Z., Fleischer, L.: Submodular approximation: Sampling-based algorithms and lower bounds. In: 49th Annual IEEE Symposium on Foundations of Computer Science, pp. 697–706. IEEE Computer Society, Washington DC (2008)
16. Vondrák, J.: Optimal approximation for the submodular welfare problem in the value oracle model. In: 40th ACM Symposium on Theory of Computer Science, pp. 67–74. ACM, New York (2008)
17. Vondrák, J.: Symmetry and approximability of submodular maximization problems. In: 50th Annual IEEE Symposium on Foundations of Computer Science, pp. 651–670. IEEE Computer Society, Washington DC (2009)
18. Wolsey, L.A.: An analysis of the greedy algorithm for the submodular set covering problem. Combinatorica 2(4), 385–393 (1982)

Submodular Cost Allocation Problem and Applications*

Chandra Chekuri and Alina Ene

Dept. of Computer Science, University of Illinois, Urbana, IL 61801, USA
{chekuri,ene1}@cs.illinois.edu

Abstract. We study the Minimum Submodular-Cost Allocation problem (MSCA). In this problem we are given a finite ground set V and k non-negative submodular set functions $f_1, \ldots, f_k$ on V. The objective is to partition V into k (possibly empty) sets $A_1, \cdots, A_k$ such that the sum $\sum_{i=1}^{k} f_i(A_i)$ is minimized. Several well-studied problems such as the non-metric facility location problem, multiway-cut in graphs and hypergraphs, and uniform metric labeling and its generalizations can be shown to be special cases of MSCA. In this paper we consider a convex-programming relaxation obtained via the Lovász-extension for submodular functions. This allows us to understand several previous relaxations and rounding procedures in a unified fashion and also develop new formulations and approximation algorithms for related problems. In particular, we give a $(1.5 - 1/k)$-approximation for the hypergraph multiway partition problem. We also give a $\min\{2(1-1/k), H_\Delta\}$-approximation for the hypergraph multiway cut problem when Δ is the maximum hyperedge size. Both problems generalize the multiway cut problem in graphs and the hypergraph cut problem is approximation equivalent to the node-weighted multiway cut problem in graphs.

1 Introduction

We consider the following allocation problem with submodular costs.

Minimum Submodular-Cost Allocation (MSCA)**.** Let V be a finite ground set and let $f_1, \cdots, f_k$ be k non-negative submodular set functions on V. That is, for $1 \leq i \leq k$, $f_i : 2^V \to \mathbb{R}_+$ and $f_i(A) + f_i(B) \geq f_i(A \cup B) + f_i(A \cap B)$ for all $A, B \subseteq V$. In the MSCA problem the goal is to partition the ground set V into k (possibly empty) sets $A_1, \cdots, A_k$ such that the sum $\sum_{i=1}^{k} f_i(A_i)$ is minimized.

We observe that the problem is interesting only if the f_i's are different for otherwise allocating all of V to f_1 is trivially an optimal solution. We assume that the functions f_i are given as a value oracle although in specific applications they may be available as explicit poly-time computable functions of some auxiliary

* This is an extended abstract without proofs. A longer version of the paper will be made available on the arXiv. The authors are supported in part by United States NSF grants CCF-0728782 and CCF-1016684.

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 354–366, 2011.

input. The special case of this problem in which all of the functions are monotone ($f(A) \leq f(B)$ if $A \subseteq B$) has been previously considered by Svitkina and Tardos [21]. In this paper, we consider the problem with both monotone and non-monotone functions. We show that several well-studied problems such as non-metric facility location, multiway cut problems in graphs and hypergraphs, uniform metric labeling and its generalization to hub location among others can be cast as special cases of MSCA. In particular, we investigate the integrality gap of a simple and natural convex-programming relaxation for MSCA that is obtained via the use of the Lovász extension of a submodular function.

Lovász extension and a convex program for MSCA: Let V be a finite ground set of cardinality n. Each real-valued set function on V corresponds to a function $f : \{0,1\}^n \to \mathbb{R}$ on the vertices of the n-dimensional hypercube. The Lovász extension of f to the continuous domain $[0,1]^n$ denoted by $\hat{f}$ is defined as[1]

$$\hat{f}(\mathbf{x}) = \mathbb{E}_{\theta \in [0,1]} \left[f(\mathbf{x}^\theta) \right] = \int_0^1 f(\mathbf{x}^\theta) d\theta$$

where $\mathbf{x}^\theta \in \{0,1\}^n$ for a given vector $\mathbf{x} \in [0,1]^n$ is defined as: $x_i^\theta = 1$ if $x_i \geq \theta$ and 0 otherwise.

Lovász showed that $\hat{f}$ is convex if and only if f is a submodular set function [16]. Moreover, it is easy to see that, given $\mathbf{x}$, the value $\hat{f}(\mathbf{x})$ can be computed in polynomial-time by using a value oracle for f. Via this extension, we obtain a straightforward relaxation for MSCA with a convex objective function and linear constraints. Let $v_1, \cdots, v_n$ denote the elements of V. The relaxation has variables $x(v,i)$ for $v \in V$ and $1 \leq i \leq k$ with the interpretation that $x(v,i)$ is 1 if v is assigned to A_i and 0 otherwise. Let $\mathbf{x}_i = (x(v_1,i), \cdots, x(v_n,i))$. The relaxation is given below.

LE-Rel

$$\begin{aligned} \min \quad & \sum_{i=1}^{k} \hat{f}_i(\mathbf{x}_i) & \\ & \sum_{i=1}^{k} x(v,i) = 1 & \forall v \\ & x(v,i) \geq 0 & \forall v, i \end{aligned}$$

Throughout, we use OPT and OPT_{FRAC} to denote the value of an optimal integral and an optimal fractional solution to LE-Rel (respectively).

We remark that LE-Rel can be solved in time that is polynomial in n and $\log(\max_{i, S \subseteq V} f_i(S))$ via the ellipsoid method. Moreover, for some problems of interest the above convex program can be rewritten into an equivalent linear program. We now describe several problems that can be cast as special cases of MSCA, and also how some previously considered linear-programming relaxations can be seen as being equivalent to the convex program above.

[1] The definition is not the standard one but is equivalent to it, see [23]. This definition is convenient to us in describing and understanding rounding procedures.

1.1 Problems Related to MSCA

Monotone MSCA (Monotone-MSCA) and Facility Location: In facility location, we have a set of facilities $\mathcal{F}$ and a set of clients or demands $\mathcal{D}$. There is a non-negative cost c_{ij} to connect facility i to client j (we do not necessarily assume that these costs form a metric). Opening facility $i \in \mathcal{F}$ costs f_i. The goal is to open a subset of the facilities and assign each client to an open facility so as to minimize the sum of the facility opening cost and the connection costs. Svitkina and Tardos [21] considered the setting where the cost of opening a facility i is a monotone submodular function g_i of the clients assigned to it, and gave an $(1+\ln|\mathcal{D}|)$-approximation, and matching hardness via a reduction from set cover. We note that this problem is equivalent to MSCA when all the f_i are monotone submodular functions, which we refer to as Monotone-MSCA. In [21] a greedy algorithm via submodular function minimization is used to derive the approximation. Here we prove that the integrality gap of LE-Rel is $(1+\ln|\mathcal{D}|)$, and describe how certain rounding algorithms achieve this bound. These algorithms are useful when considering functions that are not necessarily monotone.

Submodular Multiway Partition (Sub-MP): We define an abstract problem and then specialize to known problems. Let $f : 2^V \to \mathbb{R}_+$ be a submodular set function over V and let $S = \{s_1, s_2, \ldots, s_k\}$ be k terminals in V. The submodular multiway partition problem is to find a partition of V into $A_1, \ldots, A_k$ such that $s_i \in A_i$ and $\sum_{i=1}^{k} f(A_i)$ is minimized. This has been previously considered by Zhao, Nagamochi and Ibaraki [26]. This can be seen as a special case of MSCA as follows. Define the ground set to be $V' = V \setminus S$ and, for $1 \le i \le k$, $f_i : 2^{V'} \to \mathbb{R}_+$ is the function defined as $f_i(S) = f(S \cup \{s_i\})$. If in addition f is symmetric ($f(A) = f(V - A)$ for all A) we call this the symmetric Sub-MP problem (Sym-Sub-MP). Note that although the problem is based on a single function f, k different submodular functions (induced by the terminals) are needed to reduce it to MSCA. We now discuss some important special cases of this problem.

Multiway Cut in Graphs (Graph-MC): The input is an edge-weighted undirected graph $G = (V, E)$ and k terminal vertices $S = \{s_1, \ldots, s_k\}$; the goal is to remove a minimum-weight set of edges to disconnect the terminals. This can be seen as a special case of the symmetric submodular multiway partition problem by simply choosing f to be the cut-capacity function of G scaled down by a factor of 2. That is, $f(A) = \frac{1}{2}\sum_{e \in \delta(A)} w(e)$ where $w(e)$ is the weight of edge e. We observe that LE-Rel for this problem is equivalent to the well-known geometric LP relaxation of Calinescu, Karloff and Rabani [2], which led to significant improvements ($1.5 - 1/k$ in [2] and 1.3438 in [13]) over the $2(1-1/k)$-approximation obtained via the isolating-cut heuristic [4].

Multiway Cut and Partition in Hyper-Graphs: Given an edge-weighted hypergraph $\mathcal{G} = (V, \mathcal{E})$ and terminal set $S \subset V$, the Hypergraph Multiway Cut problem (Hypergraph-MC) (see [17,25,7]) asks for the minimum weight subset

of hyperedges whose removal disconnects the terminals. This can be seen as a special case of Sub-MP [17]; this reduction requires some care and the underlying submodular function is *asymmetric*. A related problem is the Hypergraph Multiway Partition problem (Hypergraph-MP) introduced by Lawler [15] where the cost for hyperedge e is proportional to the number of non-trivial pieces it is partitioned into. This can be seen as a special case of the Sym-Sub-MP with f being the hypergraph cut capacity function. We note that Graph-MC is a special case of both Hypergraph-MC and Hypergraph-MP.

Node-weighted Multiway Cut in Graphs (Node-wt-MC): In this problem [8] the graph has weights on nodes instead of edges and the goal is to find a minimum weight subset of nodes whose removal disconnects a given set of terminals. It is not difficult to show that Hypergraph-MC and Node-wt-MC are approximation equivalent [17].

Zhao *et al.* [26] consider generalizations of the above problems where some set of terminals $S \subseteq V$ and k are specified and the goal is to partition V into k sets such that each set contains at least one terminal and the total cost of the partition is minimized. We do not discuss these further since they are not directly related to MSCA, although one can reduce them to MSCA if k is a fixed constant.

Uniform Metric Labeling and Submodular Cost Labeling (Sub-Label): The metric labeling problem was introduced by Kleinberg and Tardos [14] as a general classification problem. We are given an undirected edge-weighted graph $G = (V, E)$ and k labels and the goal is to assign a label to each vertex to minimize the sum of the labeling cost and the edge-cut cost. The labeling cost is given by functions $c_i : V \to \mathbb{R}_+$ and the edge-cut cost is given by a metric d on the label space; $d(ij)$ is the distance between labels i and j. Assigning label i to v incurs a cost $c_i(v)$ and if an edge uv of weight $w(uv)$ has u labeled with i and v labeled with j then the edge-cut cost incurred is $w(uv) \cdot d(ij)$. The uniform metric labeling problem is obtained when $d(ij) = 1$ for all $i \neq j$. We consider the following generalization that we call the Submodular Cost Labeling (Sub-Label) problem which is a special case of MSCA. The k labels correspond to the k functions $f_1, \ldots, f_k$. We define f_i as the sum of two functions, a monotone function g_i that models the label assignment cost, and a non-monotone function h that models the cut-cost. The goal then is to partition V into $A_1, \ldots, A_k$ to minimize $\sum_{i=1}^{k} (g_i(A_i) + h(A_i))$. Note that uniform metric labeling is the special case when g_i are modular and h is the graph cut function, which is symmetric. We are motivated to consider this generalization by problems that have been considered previously, such as metric labeling on hypergraphs, hub location problem [9], and the extension of metric labeling to handle label opening costs [5].

1.2 Overview of Results and Techniques

In this paper we examine the complexity of MSCA primarily through the "integrality gap" of the convex relaxation LE-Rel which can be optimized in

polynomial time. All the problems we consider are NP-hard and our focus is on polynomial time approximation algorithms.

A significant portion of our contribution is to highlight the naturalness of MSCA and the Lovász-extension based relaxation LE-Rel by showing connections to previously studied problems, linear programming relaxations, and rounding strategies. Viewing these problems in the more abstract setting of submodularity gives insights into prior algorithms. In the process, we obtain new and interesting results. Although one would like to obtain a single unifying algorithm that achieves a good approximation for MSCA, it turns out that LE-Rel has a large integrality gap and we believe that MSCA is hard to approximate to a polynomial factor. However, it is fruitful to examine special cases of MSCA that admit good approximations via LE-Rel. We describe several applications below by summarizing our results; all of them are based on LE-Rel.

- The integrality gap of LE-Rel for Monotone-MSCA is $\Theta(\log n)$.
- There is a $(1.5 - 1/k)$-approximation for Hypergraph-MP.
- There is a $\min\{2(1 - 1/k), H_\Delta\}$-approximation for HMC, where Δ is the maximum hyperedge size and H_i is the i-th harmonic number. For $\Delta = 2$ this gives a 1.5-approximation and for $\Delta = 3$ this gives a 1.833-approximation.
- LE-Rel for HMC gives a new mathematical programming relaxation for Node-wt-MC and a new $2(1 - 1/k)$-approximation. Moreover, if all non-terminal nodes have degree at most 3 we obtain a 1.833-approximation improving upon the $2(1 - 1/k)$ known via the distance-based relaxation [8].
- The integrality gap of LE-Rel for Sym-Sub-MP is at most $2-2/k$; this gives an alternative approximation to previous combinatorial algorithms [18,26]. We raise the question as to whether the integrality gap is at most 1.5.
- There is an $O(\log n)$ for Sub-Label when the cut function is symmetric. We derive results for other special cases of Sub-Label.

Rounding the convex relaxation: Recall that the objective function in LE-Rel is $\sum_{i=1}^{k} \hat{f}_i(\mathbf{x}_i)$, where $\hat{f}_i(\mathbf{x}_i) = \mathbb{E}_{\theta \in [0,1]}[f(\mathbf{x}_i^\theta)]$. How do we round while preserving the objective function? If we focus on a specific i, the objective function suggests that we pick θ randomly from $[0, 1]$ and assign the elements in $\mathbf{x}_i^\theta$ to i; we call this θ-rounding. However, there are two issues to contend with. First, if we independently round for each i then the same element may be assigned multiple times. Second, we need to ensure that all elements are assigned, which is not guaranteed by the θ-rounding. We remark that there is an integrality gap example for hypergraph metric labeling that shows that there is no effective rounding strategy that works in general.

Our approach is to understand the rounding process by considering various special cases of interest. In particular, we consider monotone functions, symmetric functions, the hypergraph cut function (which is asymmetric), and combinations of such functions. Monotonocity helps in that if elements are assigned

to a label i, they can be removed without increasing the fractional cost. Although one can use different strategies to obtain an $O(\log n)$-approximation and integrality gap, a useful strategy here is the rounding of Kleinberg and Tardos [14] that they introduced for metric labeling. This has the additional property of ensuring that an element u is assigned to i with probability exactly $x(u,i)$. We then consider the rounding process for Sub-MP, in particular the symmetric case Sym-Sub-MP. Here, we crucially take advantage of the fact that there is a single underlying function f, and moreover the fact that it is symmetric. We consider the CKR-Rounding strategy from [2] and show its effectiveness for hypergraphs by abstracting away some of the properties specific to graphs that were previously exploited in the analysis. In the process, we also observe that a variant is equally effective for graphs but is more insightful for Sym-Sub-MP.

Finally, Sub-Label combines a monotone function and a non-monotone function. Here, we resort to KT-Rounding since it is a reasonable strategy to approximately preserve the cost of the monotone component. For the uniform metric labeling problem, [14] showed that KT-Rounding approximately (to within a factor of 2) preserves the fractional connection cost in the case of graphs. We show bounds for hypergraph cut functions in an analogous fashion. Our insights enable us to develop a variant of the rounding that gives an $O(\log n)$-approximation for Sub-Label when the cut function is an arbitrary symmetric submodular function. Due to space constraints, we omit the results for metric labeling.

Other Related Work: Due to space constraints, in this extended abstract we restrict our attention to closely related work. There has been much recent interest in optimizing with submodular set functions. In particular, maximization problems have been examined via combinatorial techniques as well as the multilinear relaxation [1]. The submodular welfare problem [22] is similar in spirit to MSCA except that one is interested in maximizing the value of an allocation rather than minimizing the cost. Minimization problems with submodular costs have also received substantial attention [19,11,12,10] with several negative results for basic problems as well as positive approximation results for problems such as the submodular cost vertex cover problem [12,10]. Lovász-extension based convex programs have been effectively used for these problems. Various submodular cut and partition problems and their special cases such as the hypergraph cut and partition have been studied recently [26,25,17,7]; however, these papers have typically focussed on greedy and divide-and-conquer based approaches while we use LE-Rel.

Recent Results for Sym-Sub-MP and Sub-MP: Very recently, building on the work in this paper and a non-trivial new technical theorem, we showed [3] that the integrality gap of SubMP-Rel is at most $1.5 - 1/k$ for Sym-Sub-MP and at most 2 for Sub-MP.

2 Monotone MSCA

In this section we consider MONOTONE-MSCA where $f_1, \ldots, f_k$ are monotone submodular functions. We will assume for simplicity that $f_i(\emptyset) = 0$ for all i.

KT-Rounding

let x be a solution to LE-REL
$S \leftarrow \emptyset$ $\quad$ $\langle\langle$*set of all assigned vertices*$\rangle\rangle$
$\langle\langle$*set of vertices that are eventually assigned to* $i\rangle\rangle$
$A_i \leftarrow \emptyset$ for all i $(1 \leq i \leq k)$
while $S \neq V$
  pick $i \in \{1, 2, \cdots, k\}$ uniformly at random
  pick $\theta \in [0, 1]$ uniformly at random
  $A_i \leftarrow A_i \cup (\{v \mid x(v, i) \geq \theta\} - S)$
  $S \leftarrow S \cup A_i$
return $(A_1, \cdots, A_k)$

Svitkina and Tardos [21] considered this problem in the context of facility location and gave a $(1 + \ln n)$-approximation and matching hardness via an approximation preserving reduction from set cover. Let $\alpha = \min_{S \subseteq V, 1 \leq i \leq k} f_i(S)/|S|$. The main observation in [21] is that $\alpha \leq \mathrm{OPT}/n$, and moreover a pair (S, i) such that $f_i(S)/|S| = \alpha$ can be computed in polynomial-time via submodular function minimization. One can then iterate using a greedy scheme, by using the monotonicity of the functions, to obtain a $(1 + \ln n)$-approximation. Using a similar argument, we can prove the following theorem.

Theorem 1. *The integrality gap of* LE-REL *for* MONOTONE-MSCA *is at most* $(1 + \ln n)$. *In particular,* $\alpha \leq \mathrm{OPT}_{\mathrm{FRAC}}/n$.

We consider other rounding algorithms that also achieve an $O(\log n)$-approximation. We focus on KT-ROUNDING derived from the work of Kleinberg and Tardos on metric labeling [14].

Theorem 2. KT-ROUNDING *achieves a randomized* $O(\ln n)$*-approximation for* MONOTONE MSCA.

3 Submodular Multiway Partition

We consider MSCA when the f_i can be non-monotone. We can show that the integrality gap of LE-REL even for a special case of labeling on hypergraphs can be $\Omega(n)$, and we suspect that the problem is hard to approximate to a polynomial factor in n. We therefore focus on SUBMODULAR MULTIWAY PARTITION (SUB-MP) and SUBMODULAR COST LABELING (SUB-LABEL); these are broad special cases which capture several problems that have been considered previously.

The reduction of SUB-MP to MSCA requires one to work with the non-terminals V' as the ground set. It is however more convenient to work with the terminals and non-terminals. In particular, we work with the relaxation below. Recall that $\mathbf{x}_i = (x(v_1, i), \cdots, x(v_n, i))$.

SubMP-Rel

$$\begin{aligned} \min \quad & \sum_{i=1}^{k} \hat{f}(\mathbf{x}_i) & & \\ & \sum_{i=1}^{k} x(v,i) & = 1 & \forall v \\ & x(s_i,i) & = 1 & \forall i \\ & x(v,i) & \geq 0 & \forall v,i \end{aligned}$$

As before, a starting point for rounding the relaxation is the basic θ-rounding that preserves the objective function. Suppose we do θ-rounding for each i to obtain sets $A(1,\theta),\ldots,A(k,\theta)$ where each $A(i,\theta) \subseteq V$. Here we could use independent random θ values for each i or the same θ. Note that the constraints ensure that $s_i \in A(j,\theta)$ iff $i = j$. However, the sets $A(1,\theta),\ldots,A(k,\theta)$ may intersect and also may not cover the entire set V, in which case we have to allocate the remaining elements in some fashion. First we show how to take advantage of the case when f is symmetric and then discuss how to obtain results for hypergraph problems that are special cases of Sub-MP.

A $2(1-1/k)$-approximation for Sym-Sub-MP: A $2(1-1/k)$-approximation for Sym-Sub-MP is known via greedy combinatorial algorithms [18,26]. However, no mathematical programming formulation for the problem has been previously considered. Here we show that, on instances of Sym-Sub-MP, the integrality gap of LE-Rel is $2(1-1/k)$ by using an uncrossing property of symmetric functions.

The following lemma is standard and it has been used in previous work [20].

Lemma 1. *Let f be a symmetric submodular set function over V and let $A_1,\ldots,A_k$ be subsets of V. Then there exist sets $A'_1,\ldots,A'_k$ such that (i) $A'_i \subseteq A_i$ for $1 \leq i \leq k$, (ii) $A'_1,\ldots,A'_k$ are mutually disjoint (iii) $\cup_i A'_i = \cup_i A_i$ and (iv) $\sum_i f(A'_i) \leq \sum_i f(A_i)$. Moreover, given the A_i's a collection of sets A'_i satisfying the above properties can be found in polynomial time via a value oracle for f.*

Theorem 3. *The integrality gap of* LE-Rel *for* Sym-Sub-MP *is* $\leq 2(1-1/k)$.

In an earlier version of this paper, we raised the following question.

Question. Is the integrality gap of LE-Rel for Sym-Sub-MP at most 1.5? As we already noted, we have shown in subsequent work [3].

Rounding for Hypergraph-MC and Hypergraph-MP: Calinescu *et al.* [2] gave a new geometric relaxation for Graph-MC, and a rounding procedure that gave a $(1.5-1/k)$-approximation; the integrality gap was subsequently improved to a bound of $1.3438-\varepsilon_k$ in [13], while the best known lower bound is $8/(7+1/k-1)$ [6]. Calinescu *et al.* [2] derived their relaxation as a way to improve the integrality gap of $2(1-1/k)$ for a natural distance based linear programming relaxation; in fact, it often goes unnoticed that [2] shows the equivalence of their geometric relaxation to that of another relaxation obtained by adding valid strengthening constraints to the distance based relaxation. Interestingly, when we specialize MSCA to Graph-MC, LE-Rel becomes the geometric relaxation

of [2]! The rounding procedure in [2] can be naturally extended to rounding LE-REL for SUB-MP and we describe it below.

CKR-Rounding

let x be a solution to SUBMP-REL
pick a random permutation π of $\{1, 2, \cdots, k\}$
pick $\theta \in [0, 1)$ uniformly at random
$S \leftarrow \emptyset$ $\quad$ $\langle\langle$*set of all assigned vertices*$\rangle\rangle$
for $i = 1$ to $k - 1$
$\quad A_{\pi(i)} \leftarrow (\{v \mid x(v, \pi(i)) \geq \theta\} - S)$
$\quad S \leftarrow S \cup A_{\pi(i)}$
$A_{\pi(k)} \leftarrow V - S$
return $(A_1, \cdots, A_k)$

CKR-ROUNDING uses the same θ for all i and a random permutation, both of which are crucially used in the 1.5-approximation analysis for GRAPH-MC. In this paper we investigate CKR-ROUNDING and other roundings for HYPERGRAPH-MC and HYPERGRAPH-MP.

Although HYPERGRAPH-MC and HYPERGRAPH-MP appear similar, their objective functions are different. The objective of HYPERGRAPH-MC is to remove a minimum weight subset of hyperedges such that the terminals are separated, whereas the objective of HYPERGRAPH-MP is to minimize $\sum_e w(e)p(e)$, where $p(e)$ is the number of non-trivial parts that e is partitioned into (a part is non-trivial if some vertex of e is in that part but not all of e). For graphs we have that either $p(e) = 0$ or $p(e) = 2$, and therefore the two problems HYPERGRAPH-MC and HYPERGRAPH-MP are equivalent; this is the reason why one can view GRAPH-MC as a partition problem as well. However, when the hyperedges can have size larger than 2, the objective function values are not related to each other (it is easy to see that the HYPERGRAPH-MP objective is always larger).

HYPERGRAPH-MP and HYPERGRAPH-MC have been studied for their theoretical interest and their applications. It is easy to see from its definition that HYPERGRAPH-MP is a special case of SYM-SUB-MP. It has been observed by a simple yet nice reduction [17] that HYPERGRAPH-MC is a special case of SUB-MP. In addition, it has been observed that HYPERGRAPH-MC is approximation-equivalent to the *node*-weighted multiway cut problem in graphs (NODE-WT-MC) [8].

We show that CKR-ROUNDING gives a $(1.5 - 1/k)$-approximation to HYPERGRAPH-MP and a tight H_Δ-approximation for HYPERGRAPH-MC with maximum hyperedge size Δ. Note that when $\Delta = 2$, $H_\Delta = 1.5$ and when $\Delta = 3$, $H_\Delta \simeq 1.833$. For $\Delta > 3$, CKR-ROUNDING gives a worse than 2 bound while we give an alternate rounding which gives a $2(1 - 1/k)$-approximation. Our analysis of CKR-ROUNDING differs from that in [2] since we cannot use the "edge alignment" properties of the fractional solution that hold for graphs and were exploited in [2]; our analysis is inspired by the proof given by Williamson and Shmoys [24].

It is natural to wonder whether CKR-ROUNDING is crucial to obtaining a bound that is better than 2 for these problems, and in particular whether it gives a 1.5-approximation for SYM-SUB-MP. We show that a $1.5 - 1/k$-approximation for HYPERGRAPH-MP (and hence GRAPH-MC also) can be obtained via a

different algorithm as well; in particular, the crucial ingredient in CKR-Rounding for Graph-MC when viewed as a special case of Hypergraph-MP is the correlation provided by the use of the same θ for all i; one can replace the random permutation by the uncrossing scheme in Lemma 1. We describe this algorithm in the next section. However, for Hypergraph-MC, the random permutation is important in proving the H_Δ-bound.

3.1 A 1.5-Approximation for Hypergraph Multiway Partition

We start by understanding the objective function of SubMP-Rel in the context of Hypergraph-MP. Let $\mathbf{x}$ be a feasible fractional solution, and let $\mathbf{x}_i = (x(v_1, i), \cdots, x(v_n, i))$ be the allocation to i. Recall that f here is the hypergraph cut function. What is $\sum_{i=1}^{n} \hat{f}(\mathbf{x}_i)$? For each terminal i and each hyperedge e, let $I(e, i) = [\min_{v \in e} x(v, i), \max_{v \in e} x(v, i)]$. Let $d(e, i)$ denote the length of $I(e, i)$, and let $d(e) = \sum_{i=1}^{k} d(e, i)$. Note that $d(e) \in [0, |e|]$.

Lemma 2. $\sum_{i=1}^{k} \hat{f}(\mathbf{x}_i) = \sum_e w(e) d(e)$.

A crucial technical lemma that we need is the following which states that the contribution of any i to $d(e)$ is at most $d(e)/2$.

Lemma 3. *For any i, $d(e, i) \leq d(e)/2$.*

SymSubMP-Rounding

let x be a feasible solution to SubMP-Rel
pick $\theta \in [0, 1]$ uniformly at random
$A(i, \theta) \leftarrow \{v \mid x(v, i) \geq \theta\}$ for each i $(1 \leq i \leq k)$
$\langle\langle$*uncross* $A(1, \theta), \cdots, A(k, \theta)\rangle\rangle$
$A'_i \leftarrow A(i, \theta)$ for each i $(1 \leq i \leq k)$
while there exist $i \neq j$ such that $A'_i \cap A'_j \neq \emptyset$
  if $\left(f(A'_i) + f(A'_j - A'_i) \leq f(A'_i) + f(A'_j)\right)$
    $A'_j \leftarrow A'_j - A'_i$
  else
    $A'_i \leftarrow A'_i - A'_j$
return $(A'_1, \cdots, A'_{k-1}, V - (A'_1 \cup \cdots A'_{k-1}))$

The algorithm SymSubMP-Rounding that we analyze is described in the adjacent box. We can prove that CKR-Rounding gives the same bound; however, SymSubMP-Rounding and its analysis are perhaps more intuitive in the context of symmetric functions. The algorithm does θ-rounding to obtain sets $A(1, \theta), \ldots, A(k, \theta)$ and then uncrosses these sets to make them disjoint without increasing the expected cost (see Lemma 1).

Theorem 4. SymSubMP-Rounding *achieves an $(1.5 - 1/k)$-approximation for* Hypergraph-MP.

Lemma 4. *Let i^* be the index such that the interval $I(e, i^*)$ has the rightmost ending point among the intervals $I(e, i)$. More precisely, $I(e, i^*)$ is an interval such that $\max_{v \in e} x(v, i^*) = \max_i \max_{v \in e} x(v, i)$; if there are several such intervals, we choose one arbitrarily. Let Z_e be an indicator random variable equal to 1 iff $e \in \delta(V - (A(1, \theta) \cup \cdots \cup A(k, \theta)))$. Then $\mathbb{E}[Z_e] \leq d(e, i^*)$.*

Theorem 4 follows from Lemma 1 and Lemma 4.

3.2 Algorithms for Hypergraph Multiway Cut

Now we consider HYPERGRAPH-MC. For each hyperedge e, pick an arbitrary representative node $r(e) \in e$. Define the function $f : 2^V \to \mathbb{R}_+$ as follows: for $A \subseteq V$, let $f(A) = \sum_{e:r(e)\in A, e\not\subseteq A} w(e)$ be the weight of hyperedges whose representatives are in A and they cross A. It is easy to verify that f is asymmetric and submodular. SUB-MP with this function f captures HYPERGRAPH-MC [17].

Let $\mathbf{x}$ be a feasible fractional allocation and $\mathbf{x}_i$ be the allocation for i. For each hyperedge e and each terminal i, let $I(e,i) = [\min_{v\in e} x(v,i), \max_{v\in e} x(v,i)]$. Let $d(e,i) = x(r(e),i) - \min_{v\in e} x(v,i)$ and $d(e) = \sum_{i=1}^{k} d(e,i)$.

Lemma 5. $\sum_{i=1}^{k} \hat{f}(\mathbf{x}_i) = \sum_e w(e)d(e)$.

Half-Rounding

let x be a solution to SUBMP-REL
pick $\theta \in (1/2, 1]$ uniformly at random
for $i = 1$ to $k-1$
  $A(i,\theta) \leftarrow \{v \mid x(v,i) \geq \theta\}$
$U(\theta) \leftarrow V - (A(1,\theta) \cup \cdots \cup A(k-1,\theta))$
return $(A(1,\theta), \cdots, A(k-1,\theta), U(\theta))$

For HYPERGRAPH-MC we show that HALF-ROUNDING achieves a $2(1-1/k)$-approximation and that CKR-ROUNDING achieves an H_Δ-approximation where Δ is the maximum hyperedge size.

Theorem 5. *Let F be the set of all hyperedges crossing the partition returned by* HALF-ROUNDING. *For each hyperedge e,* $\Pr[e \in F] \leq 2d(e)$.

Theorem 6. *Let F be the set of all hyperedges crossing the partition returned by* CKR-ROUNDING. *For each hyperedge e,* $\Pr[e \in F] \leq H_{|e|} \cdot d(e)$. *Moreover, this analysis is tight for the rounding.*

Acknowledgments. We thank Lisa Fleischer for suggesting that we contact Zoya Svitkina about MSCA and thank Zoya for pointing out her work in [21] on monotone MSCA. CC thanks Jan Vondrak for pointing out the interpretation of the Lovász extension from his paper [23] which was very helpful in thinking about rounding procedures. AE thanks Sungjin Im and Ben Moseley for discussions.

References

1. Calinescu, G., Chekuri, C., Pál, M., Vondrák, J.: Maximizing a submodular set function subject to a matroid constraint (Extended abstract). In: Fischetti, M., Williamson, D.P. (eds.) IPCO 2007. LNCS, vol. 4513, pp. 182–196. Springer, Heidelberg (2007)
2. Calinescu, G., Karloff, H.J., Rabani, Y.: An improved approximation algorithm for multiway cut. Journal of Computer and System Sciences 60(3), 564–574 (2000), Preliminary version in STOC 1998
3. Chekuri, C., Ene, A.: Approximation algorithms for submodular multiway partition (April 2011) (manuscript)
4. Dahlhaus, E., Johnson, D.S., Papadimitriou, C.H., Seymour, P.D., Yannakakis, M.: The complexity of multiterminal cuts. SIAM Journal on Computing 23(4), 864–894 (1994), Preliminary version in STOC 1992

5. Delong, A., Osokin, A., Isack, H.N., Boykov, Y.: Fast approximate energy minimization with label costs. In: IEEE Computer Vision and Pattern Recognition (CVPR), pp. 2173–2180 (2010)
6. Freund, A., Karloff, H.J.: A lower bound of 8/(7+(1/k)-1) on the integrality ratio of the calinescu-karloff-rabani relaxation for multiway cut. Information Processing Letters 75(1-2), 43–50 (2000)
7. Fukunaga, T.: Computing Minimum Multiway Cuts in Hypergraphs from Hypertree Packings. In: Eisenbrand, F., Shepherd, F.B. (eds.) IPCO 2010. LNCS, vol. 6080, pp. 15–28. Springer, Heidelberg (2010)
8. Garg, N., Vazirani, V.V., Yannakakis, M.: Multiway cuts in node weighted graphs. Journal of Algorithms 50(1), 49–61 (2004), Preliminary version in ICALP 1994
9. Ge, D., Ye, Y., Zhang, J.: The Fixed-Hub Single Allocation Problem: A Geometric Rounding Approach (2007), preprint
http://www.stanford.edu/~yyye/revisedHub.pdf
10. Goel, G., Karande, C., Tripathi, P., Wang, L.: Approximability of combinatorial problems with multi-agent submodular cost functions. In: IEEE Symposium on Foundations of Computer Science (FOCS), pp. 755–764 (2009)
11. Goemans, M.X., Harvey, N.J.A., Iwata, S., Mirrokni, V.S.: Approximating submodular functions everywhere. In: ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 535–544 (2009)
12. Iwata, S., Nagano, K.: Submodular function minimization under covering constraints. In: IEEE Symposium on Foundations of Computer Science (FOCS), pp. 671–680 (2009)
13. Karger, D.R., Klein, P.N., Stein, C., Thorup, M., Young, N.E.: Rounding algorithms for a geometric embedding of minimum multiway cut. Mathematics of Operations Research 29(3), 436–461 (2004), Preliminary version in STOC 1999
14. Kleinberg, J.M., Tardos, É.: Approximation algorithms for classification problems with pairwise relationships: Metric labeling and Markov random fields. Journal of the ACM (JACM) 49(5), 616–639 (2002), Preliminary version in FOCS 1999
15. Lawler, E.L.: Cutsets and partitions of hypergraphs. Networks 3(3), 275–285 (1973)
16. Lovász, L.: Submodular functions and convexity. In: Mathematical Programming: The State of the Art, pp. 235–257 (1983)
17. Okumoto, K., Fukunaga, T., Nagamochi, H.: Divide-and-conquer algorithms for partitioning hypergraphs and submodular systems. Algorithmica, 1–20 (2010), Preliminary version in ISAAC 2009
18. Queyranne, M.: Minimizing symmetric submodular functions. Mathematical Programming 82(1), 3–12 (1998), Preliminary version in SODA 1995
19. Svitkina, Z., Fleischer, L.: Submodular approximation: Sampling-based algorithms and lower bounds. In: IEEE Symposium on Foundations of Computer Science (FOCS), pp. 697–706 (2008)
20. Svitkina, Z., Tardos, É.: Min-max multiway cut. In: Jansen, K., Khanna, S., Rolim, J.D.P., Ron, D. (eds.) RANDOM 2004 and APPROX 2004. LNCS, vol. 3122, pp. 207–218. Springer, Heidelberg (2004)
21. Svitkina, Z., Tardos, É.: Facility location with hierarchical facility costs. ACM Transactions on Algorithms (TALG) 6(2), 1–22 (2010), Preliminary version in SODA 2006
22. Vondrák, J.: Optimal approximation for the submodular welfare problem in the value oracle model. In: ACM Symposium on Theory of Computing (STOC), pp. 67–74 (2008)

23. Vondrák, J.: Symmetry and Approximability of Submodular Maximization Problems. In: IEEE Symposium on Foundations of Computer Science (FOCS), pp. 651–670 (2010)
24. Williamson, D.P., Shmoys, D.B.: The design of approximation algorithms (2010), preprint http://www.designofapproxalgs.com
25. Xiao, M.: Finding minimum 3-way cuts in hypergraphs. Information Processing Letters 110(14-15), 554–558 (2010), Preliminary version in TAMC 2008
26. Zhao, L., Nagamochi, H., Ibaraki, T.: Greedy splitting algorithms for approximating multiway partition problems. Mathematical Programming 102(1), 167–183 (2005)

Robust Independence Systems

Naonori Kakimura* and Kazuhisa Makino*

Department of Mathematical Informatics,
University of Tokyo,
Tokyo 113-8656, Japan
{kakimura,makino}@mist.i.u-tokyo.ac.jp

Abstract. An independence system $\mathcal{F}$ is one of the most fundamental combinatorial concepts, which includes a variety of objects in graphs and hypergraphs such as matchings, stable sets, and matroids. We discuss the robustness for independence systems, which is a natural generalization of the greedy property of matroids. For a real number $\alpha > 0$, a set $X \in \mathcal{F}$ is said to be α-*robust* if for any k, it includes an α-approximation of the maximum k-independent set, where a set Y in $\mathcal{F}$ is called k-independent if the size $|Y|$ is at most k. In this paper, we show that every independence system has a $1/\sqrt{\mu(\mathcal{F})}$-robust independent set, where $\mu(\mathcal{F})$ denotes the *exchangeability* of $\mathcal{F}$. Our result contains a classical result for matroids and the ones of Hassin and Rubinstein [12] for matchings and Fujita, Kobayashi, and Makino [7] for matroid 2-intersections, and provides better bounds for the robustness for many independence systems such as b-matchings, hypergraph matchings, matroid p-intersections, and unions of vertex disjoint paths. Furthermore, we provide bounds of the robustness for nonlinear weight functions such as submodular and convex quadratic functions. We also extend our results to independence systems in the integral lattice with separable concave weight functions.

Keywords: independence systems, matroids, exchangeability, robustness.

1 Introduction

Let E be a finite set. A family $\mathcal{F}$ of subsets in E is an *independence system* if $\emptyset \in \mathcal{F}$, and $I \subseteq J \in \mathcal{F}$ implies $I \in \mathcal{F}$. A set F in $\mathcal{F}$ is called *independent*, and k-*independent* if $|F| \leq k$ holds in addition. For an independence system $\mathcal{F}$ with a nonnegative weight $w \in \mathbb{R}_+^E$ and a positive integer k, we consider the problem of finding a maximum weighted k-independent set, called *the maximum k-independent set problem*.

$$\begin{aligned} \text{Problem } \mathrm{P}_k(\mathcal{F}) : \text{maximize } & w(X) \\ \text{subject to } & |X| \leq k, \\ & X \in \mathcal{F}, \end{aligned}$$

* This work was partially supported by Grant-in-Aid for Scientific Research and by Global COE Program "The research and training center for new development in mathematics" from the Ministry of Education, Culture, Sports, Science and Technology of Japan.

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 367–378, 2011.

where for $X \subseteq E$, we define $w(X) = \sum_{i \in X} w(i)$. If k is sufficiently large, e.g., $k \geq |E|$, then it is called *the maximum independent set problem*, and denoted by $\mathrm{P}(\mathcal{F})$. The maximum (k-)independent set problem is one of the most fundamental and important combinatorial optimization problems, which includes a variety of graph and hypergraph problems such as matching, stable set, and matroid problems. See e.g., [17,20,26] for the details. The problem is NP-hard in general, while it is polynomially solvable if $\mathcal{F}$ belongs to some special classes. For example, it is well-known that, for a matroid $\mathcal{F}$, a greedy algorithm computes a maximum independent set [4,25]. Note that this greedy solution X has a good property, called *robustness*, in the sense that it contains a maximum k-independent set for each k. More precisely, for each k, the heaviest k elements in X is an optimal solution for the maximum k-independent set problem. Thus the greedy solution X is adaptable to all the sizes k. This paper investigates this kind of structural properties for the independence systems.

For an independence system, a robust independent set does not always exist. Let X_k denote an optimal solution for Problem $\mathrm{P}_k(\mathcal{F})$, and for $X = \{x_1, \ldots, x_p\} \subseteq E$ with $w(x_i) \geq w(x_j)$ if $i < j$, we define

$$w_{\leq k}(X) = \sum_{i \leq k} w(x_i), \quad k = 1, 2, \ldots, |E|.$$

For a real number $\alpha > 0$, an independent set X is called *α-robust* (with respect to w) if $w_{\leq k}(X) \geq \alpha \cdot w(X_k)$ for all k's. By definition, a robust independent set is exactly 1-robust. We also say that an independence system $\mathcal{F}$ is *α-robust* if it has an α-robust solution for any nonnegative weight w.

1.1 Previous and Our Main Results for the Robustness

The α-robustness for independence systems was first introduced by Hassin and Rubinstein [12]. They proved that the greedy solution is ν-robust, where ν is the *rank quotient* defined in Section 2.2. Moreover, they showed that the maximum matching problem admits a $1/\sqrt{2}$-robust solution, and that $1/\sqrt{2}$-robustness is the best possible for the maximum matching problem. Fujita, Kobayashi, and Makino [7] extend their matching result to the matroid intersection problem, and show that computing an α-robust matching is NP-hard for any $\alpha\,(> 1/\sqrt{2})$. The robustness was also studied for several combinatorial optimization problems such as trees and paths [8,13]. Another concept similar to the robustness, called the *incremental problems*, has been investigated for covering-type problems in connection with the online algorithms [21,24].

In this paper, we analyze the robustness for independence systems $\mathcal{F}$ by using parameter $\mu(\mathcal{F})$, called the *exchangeability* of $\mathcal{F}$. For a nonnegative integer μ, we say that $\mathcal{F}$ is *μ-exchangeable* if

$$\forall X, Y \in \mathcal{F}, \forall i \in Y \setminus X, \exists Z \subseteq X \setminus Y \text{ s.t. } |Z| \leq \mu,\ X \cup \{i\} \setminus Z \in \mathcal{F}. \quad (1)$$

We denote by $\mu(\mathcal{F})$ the minimum μ satisfying the condition above. The exchangeability was introduced by Mestre [23],[1] to measure the performance of the

[1] In fact, he introduced "μ-extendibility," which is equivalent to μ-exchangeability.

greedy algorithm. It is known [23] that $\mathcal{F}$ is a matroid if and only if $\mu(\mathcal{F}) \leq 1$, and that a number of independence systems arising from natural combinatorial optimization such as (hypergraph) matchings, stable sets, acyclic subgraphs, union of vertex disjoint paths, and matroid intersections, have bounded μ (see Section 2.1).

In this paper we obtain the following theorem.

Theorem 1. *Let $\mathcal{F}$ be an independence system on a finite set E. Then it is $\min\{1, 1/\sqrt{\mu(\mathcal{F})}\}$-robust. In particular, for any weight $w \in \mathbb{R}_+^E$, a w^2-optimal independent set is $\min\{1, 1/\sqrt{\mu(\mathcal{F})}\}$-robust with respect to w.*

Here, for a weight $u \in \mathbb{R}_+^E$, an independent set X of maximum weight $u(X)$ is called *u-optimal*. We denote by w^b the vector defined by $(w^b)(i) = w(i)^b$ for $i \in E$.

To obtain our robustness, we show a maximum weight independent set with respect to the b-th power weight w^b is $\min\{1, \mu(\mathcal{F})^{-1/b}, 1/\mu(\mathcal{F})^{-1+(1/b)}\}$-robust, where our main result is obtained by fixing $b = 2$. The statement is similar to Hassin and Rubinstein [12] and Fujita et al. [7], but different from their proofs, ours, described in Section 3, exploits a polyhedral description of the b-th power weight w^b such that a given set X is maximum with respect to w^b. Note that such constraints can be represented as (an exponential number of) inequalities which are linear in terms of w^b. We show that X is $\min\{1, \mu(\mathcal{F})^{-1/b}, 1/\mu(\mathcal{F})^{-1+(1/b)}\}$-robust with respect to any weight w satisfying the constraints, by considering the minimization of the total weight of the heaviest k elements in X subject to these inequalities.

We also show that the ratio $\min\{1, 1/\sqrt{\mu(\mathcal{F})}\}$ in Theorem 1 is tight.

Theorem 2. *For any positive integer μ, there exists an independence system $\mathcal{F}$ with $\mu(\mathcal{F}) = \mu$ and a weight $w \in \mathbb{R}_+^E$ such that for any $\alpha > 1/\sqrt{\mu}$, $\mathcal{F}$ has no α-robust independent set with respect to w.*

Since $\mathcal{F}$ is a matroid if and only if $\mu(\mathcal{F}) \leq 1$, Theorem 1 includes a classical result for the greediness of matroids [4,25] (see Corollary 1 (i)), and it can be regarded as a generalization of Hassin and Rubinstein [12] and Fujita et al. [7] (see Corollary 1 (ii) with $b \equiv 1$ and (v) with $p = 2$, respectively), since independence systems from matchings and matroid 2-intersections are both 2-exchangeable. Moreover, our result implies the existence of highly robust independent sets for a variety of combinatorial optimization problems.

Corollary 1. **(i)** *Let $\mathcal{F}$ be a matroid. Then it is 1-robust.*

(ii) *For a graph $G = (V, E)$ with $b \in \mathbb{Z}_+^V$, let $\mathcal{F} \subseteq 2^E$ be the family of b-matchings in G. Then it is $1/\sqrt{2}$-robust.*

(iii) *For a complete directed graph $G = (V, E)$, we define $\mathcal{F} \subseteq 2^E$ to be the family of unions of vertex disjoint paths, i.e., $\mathcal{F} = \{P (= \bigcup_i P_i) \subseteq E \mid P_i$ are pairwise vertex disjoint paths$\}$. Then it is $1/\sqrt{3}$-robust.*

(iv) *For a hypergraph $\mathcal{E} \subseteq 2^V$ on a finite set V, let $\mathcal{F}$ be the family of matchings in $\mathcal{E}$, i.e., the family of disjoint hyperedges in $\mathcal{E}$. Let r denote the maximum*

number of disjoint neighbors of hyperedges. Then it is $1/\sqrt{r}$-robust. In particular, for any k-hypergraph $\mathcal{E}$ (i.e., $|E| \leq k$ for $E \in \mathcal{E}$), it is $1/\sqrt{k}$-robust.

(v) *Let $\mathcal{F}$ be the intersection of $p\,(\geq 2)$ matroids. Then it is $1/\sqrt{p}$-robust.*

(vi) *For a d-dimensional knapsack problem, i.e., maximizing $w^{\mathrm{T}}x$ subject to $Ax \leq b$ and $x \in \{0,1\}^n$, where $A \in \mathbb{R}_+^{d\times n}$, $b \in \mathbb{R}_+^d$ and $w \in \mathbb{R}_+^n$, let $\mathcal{F}$ denote the independence system corresponding to the set of the feasible vectors. Define*

$$\mu(A) = \sum_{i=1}^{d} \left\lceil \frac{\max\{a_{ij} \mid 1 \leq j \leq n\}}{\min\{a_{ij} \mid a_{ij} \neq 0, 1 \leq j \leq n\}} \right\rceil.$$

Then it is $1/\sqrt{\mu(A)}$-robust.

(vii) *For a graph $G = (V, E)$, let $\mathcal{F} \subseteq 2^V$ denote the family of stable sets in G. Then it is $1/\sqrt{d_{\max}}$-robust, where $d_{\max}$ denotes the maximum degree of G.*

(viii) *For a directed graph $G = (V, E)$, let $\mathcal{F} \subseteq 2^E$ denote the family of acyclic subgraphs in G. Then it is $1/\sqrt{\lambda}$-robust, where λ denotes the maximum edge connectivity between two vertices.*

We remark that Theorem 1 improves upon the existing bounds for the robustness. For example, we have a stronger bound for the robustness of $\mathcal{F}$ if it is obtained from hypergraph matchings or unions of vertex disjoint paths in a complete directed graph (Corollary 1 (ii)(iii)(iv)). See Lemma 3 in Section 2. We also note that the bounds in the corollary are all tight as Theorem 2.

From a viewpoint of complexity, a robust solution stated above can be computed in polynomial time if the maximum independent set problem $\mathrm{P}(\mathcal{F})$ is polynomially solvable. For example, a $1/\sqrt{2}$-robust b-matching can be found in polynomial time, since the maximum b-matching problem can be computed in polynomial time [9]. If a given graph is perfect, we can find a $1/\sqrt{d_{\max}}$-robust stable set [11], and if a directed graph is planar, we can find a $1/\sqrt{\lambda}$-robust acyclic subgraph in polynomial time [6,22].

However, it is often NP-hard to solve $\mathrm{P}(\mathcal{F})$, for example, if $\mathcal{F}$ is the family of matchings in a hypergraph [2], the intersection of p matroids [19], or the family of stable sets in a graph. You might expect that a γ-approximation solution with respect to w^2 provides a $\sqrt{\frac{\gamma}{\mu}}$-robust set with respect to w. However, this is not the case.

Theorem 3. *Let E be a finite set with $|E| \geq 3$. There exist an independence system $\mathcal{F} \subseteq 2^E$ and a weight $w \in \mathbb{R}_+^E$ such that a γ-approximation solution with respect to w^2 is not $1/\sqrt{(\gamma^{-1}-1)(|E|-1)}$-robust.*

We further generalize Theorem 1 in the following two ways. First, we consider a non-linear weight $w : 2^E \to \mathbb{R}_+$. Let $h : \mathbb{R}_+ \to \mathbb{R}_+$ be a one-dimensional function, and $q \in \mathbb{R}_+^E$ be a vector. We say that a weight set function $w : 2^E \to \mathbb{R}_+$ is *ρ-approximated by h and q* if

$$h(\sum_{i\in X} q(i)) \leq w(X) \leq \rho \cdot h(\sum_{i\in X} q(i)) \quad \text{for all } X \in 2^E.$$

If h is *monotone* (i.e., $h(x) \leq h(y)$ for any $x, y \in \mathbb{R}_+$ with $x \leq y$) and *submultiplicative* (i.e., $h(xy) \leq h(x)h(y)$ for any $x, y \in \mathbb{R}_+$), then we have the following result.

Theorem 4. *If a weight $w : 2^E \to \mathbb{R}_+$ is ρ-approximated by a monotone submultiplicative function h and $q \in \mathbb{R}_+^E$, then any independence system $\mathcal{F} \subseteq 2^E$ has a $\frac{1}{\rho \cdot h(\sqrt{\mu(\mathcal{F})})}$-robust independent set with respect to w.*

As corollaries, we have the robustness results for submodular and convex quadratic weight functions as below. To obtain the result for a monotone submodular function w, we use an algorithm due to Goemans et al. [10], which finds in polynomial time and number of queries a vector $q \in \mathbb{R}_+^E$ so that a function u of the form $u(X) = \sqrt{q(X)}$ is a good approximation of w. Note that a greedy algorithm for submodular functions returns a $1/(\mu(\mathcal{F}) + 1)$-robust solution [1,5].

Corollary 2. *Let $\mathcal{F}$ be an independence system on E with $n = |E|$.*

(i) *If a weight function $w : 2^E \to \mathbb{R}_+$ is monotone submodular, then $\mathcal{F}$ has a $\frac{1}{\mu(\mathcal{F})^{1/4} O(\sqrt{n} \log n)}$-robust independent set with respect to w. In particular, if w is a matroid rank function, it has a $\frac{1}{\mu(\mathcal{F})^{1/4}\sqrt{n+1}}$-robust independent set. In either case, such robust sets can be computed in polynomial time and number of queries if $\mathrm{P}(\mathcal{F})$ is polynomially solvable.*

(ii) *Let $w : 2^E \to \mathbb{R}_+$ be a convex quadratic function, i.e., a function defined by $w(X) = \sum_{i,j \in X} a_{ij}$ for $X \subseteq E$ with a positive definite matrix $A = (a_{ij}) \in \mathbb{R}^{E \times E}$. Then $\mathcal{F}$ has a $\frac{\lambda_{\min}}{\sqrt{\mu(\mathcal{F})}\lambda_{\max}}$-robust independent set with respect to w, where $\lambda_{\max}$ and $\lambda_{\min}$ are the maximum and minimum eigenvalues of A, respectively.*

The second generalization is to independence systems in integral lattice $\mathbb{Z}_+^E$. We say that a function $w : \mathbb{Z}^E \to \mathbb{R}_+$ is *separable concave* if w can be written as $w(x) = \sum_{i \in E} w_i(x_i)$ for some one-dimensional concave functions $w_i : \mathbb{R} \to \mathbb{R}_+$ for $i \in E$. In this paper, we assume that $w_i(0) = 0$ and monotonicity.

Theorem 5. *Let $\mathcal{F}$ be a bounded independence system in $\mathbb{Z}_+^E$ and w be a separable concave function. Then $\mathcal{F}$ has a $\min\{1, 1/\sqrt{\mu(\mathcal{F})}\}$-robust independent vector with respect to w.*

By this theorem, we have a corollary for polymatroids, polymatroid intersections, and packing systems.

The rest of the paper is organized as follows. In Section 2, we define the exchangeability for independence systems and discuss its basic properties. In Section 3, we consider the robustness for independence systems in the Boolean lattice. In particular, we show Theorems 1 and 2. Due to the space limitation, most of the proofs are omitted, where they can be found in the full version of this paper [18].

2 Exchangeable Independence Systems

For a nonnegative integer μ, we say that an independence system $\mathcal{F}$ is μ-*exchangeable* if $\mathcal{F}$ satisfies (1). We denote by $\mu(\mathcal{F})$ the minimum μ satisfying (1). Note that $\mathcal{F}$ is 0-exchangeable if and only if $\mathcal{F} = 2^J$ for some $J \subseteq E$. Thus a 0-exchangeable independence system consists of the unique maximal independent set.

After providing a variety of examples of μ-exchangeable independence systems, in this section, we show further properties on μ-exchangeable independence systems. In particular, we prove that the exchangeability is at least the inverse of the rank quotient, and it is NP-hard to approximate $\mu(\mathcal{F})$ for a given $\mathcal{F}$.

2.1 Examples of Exchangeable Independence Systems

This subsection describes some basic independence systems with small μ.

Matroids: An independence system $\mathcal{M}$ is called a *matroid* if $M_1, M_2 \in \mathcal{M}$ and $|M_1| < |M_2|$ implies $M_1 \cup \{e\} \in \mathcal{M}$ for some $e \in M_2 \setminus M_1$. It is known in [23] that $\mathcal{F}$ is 1-exchangeable if and only if it is a matroid.

Matchings and b-Matchings of a Graph: For a graph $G = (V, E)$, let $\mathcal{F} \subseteq 2^E$ denote the family of matchings F in G, i.e., $e \cap e' = \emptyset$ holds for any distinct $e, e' \in F$. Note that for any edge $e \in E$ and any matching F, at most two edges in F intersect e. Thus $\mathcal{F}$ turns out to be 2-exchangeable. More generally, for a vector $b \in \mathbb{Z}_+^V$, we say that a subset $F \subseteq E$ is a *b-matching* if for all $v \in V$ the number of edges in F incident to v is at most b_v. Then the b-matchings also form a 2-exchangeable independence system.

Unions of Vertex Disjoint Paths and Asymmetric Traveling Salesman Systems: For a complete directed graph $G = (V, E)$, let $\mathcal{F}$ be the family of unions of vertex disjoint paths $P \subseteq E$, i.e., $\mathcal{F} = \{P (= \bigcup_i P_i) \subseteq E \mid P_i$ are pairwise vertex disjoint paths$\}$, and let $\mathcal{H}$ be the family of sets $H \subseteq E$ such that H is either unions of vertex disjoint paths or a Hamilton cycle. The family $\mathcal{H}$ is well studied to solve the maximum asymmetric traveling salesman problem, and it is known [16] that $\mathcal{H}$ is 3-exchangeable. Similarly to $\mathcal{H}$, we have $\mu(\mathcal{F}) \leq 3$.

Matchings in Hypergraphs: For a hypergraph $\mathcal{E} \subseteq 2^V$ on a finite set V, let $\mathcal{F}$ be the family of matchings in $\mathcal{E}$, i.e., the family of disjoint hyperedges in $\mathcal{E}$. Let r denote the maximum number of disjoint neighbors of hyperedges, i.e., $r = \max_{J \in \mathcal{E}} \{|\mathcal{M}| \mid \mathcal{M} \in \mathcal{F}, I \cap J \neq \emptyset \text{ for all } I \in \mathcal{M}\}$. Then it is not difficult to see that $\mathcal{F}$ is r-exchangeable. In particular, for any k-hypergraph $\mathcal{E}$ (i.e., $|E| \leq k$ for $E \in \mathcal{E}$), $\mathcal{F}$ is k-exchangeable. This problem is also known as *k-set packing* [3].

Mestre [23] provides a maximum profit scheduling problem as an example of the maximum independent set problem for $\mathcal{F}$ with $r = 2$. We also remark that the family of the triangles, i.e., complete subgraphs with size three, in a graph is an example with $r = 3$.

Intersections of p Matroids: For matroids $\mathcal{M}_i \subseteq 2^E$ $(i = 1, \ldots, p)$, define $\mathcal{F} = \bigcap_{i=1}^{p} \mathcal{M}_i$. Then we can see that $\mathcal{F}$ is p-exchangeable [23].

Multidimensional Knapsack Systems: For a positive integer d, we consider the following *d-dimensional knapsack problem* with n items:

$$\begin{aligned} &\text{maximize } w^{\mathrm{T}}x \\ &\text{subject to } Ax \leq b, \\ &\qquad\qquad x \in \{0,1\}^n, \end{aligned}$$

where $A \in \mathbb{R}_+^{d\times n}$, $b \in \mathbb{R}_+^d$ and $w \in \mathbb{R}_+^n$. Let $\mathcal{F}$ denote the independence system corresponding to the set of the feasible vectors. Recall that $\mu(A)$ is defined in Corollary 1 (vi). Then we have

$$\mu(\mathcal{F}) \leq \mu(A) \leq d\left\lceil \frac{a_{\max}}{a_{\min}} \right\rceil,$$

where $a_{\min}$ and $a_{\max}$ denote the minimum and maximum values of nonzero entries in A, respectively.

Stable Sets of a Graph: For a graph $G = (V, E)$, let $\mathcal{F} \subseteq 2^V$ denote the family of stable sets (also called independent sets) in G, i.e., the set of vertices not directly connected by edges. Then it is $d_{\max}$-exchangeable, where $d_{\max}$ denotes the maximum degree of G.

Acyclic Subgraphs: For a directed graph $G = (V, E)$, let $\mathcal{F} \subseteq 2^E$ denote the family of acyclic subgraphs in G. Let λ denote the maximum edge connectivity between two vertices. Then $\mathcal{F}$ is λ-exchangeable.

2.2 Exchangeability and Rank Quotient

Let $\mathcal{F}$ be an independence system. For $X, Y \in \mathcal{F}$, a pair (Z_X, Z_Y), where $Z_X \subseteq X \setminus Y$ and $Z_Y \subseteq Y \setminus X$, is said to be (X, Y)-*admissible* if $X \setminus Z_X \cup Z_Y \in \mathcal{F}$. For $X, Y \in \mathcal{F}$ with $Y \setminus X \neq \emptyset$, we denote

$$\mu_{\mathcal{F}}(X, Y) = \max_{i \in Y\setminus X} \min\{|Z| \mid (Z, \{i\}) \text{ is } (X, Y)\text{-admissible}\}.$$

If $Y \setminus X = \emptyset$, we define $\mu_{\mathcal{F}}(X, Y) = 0$. Then we have

$$\mu(\mathcal{F}) = \max_{X,Y\in\mathcal{F}} \mu_{\mathcal{F}}(X, Y).$$

If there is no ambiguity, we simply use $\mu(X, Y)$ and μ instead of $\mu_{\mathcal{F}}(X, Y)$ and $\mu(\mathcal{F})$, respectively. We observe that $\mu(\mathcal{F}) = \max_{X,Y\in\mathcal{B}_{\mathcal{F}}} \mu_{\mathcal{F}}(X, Y)$, where $\mathcal{B}_{\mathcal{F}}$ is the family of maximal independent sets in $\mathcal{F}$.

We next consider the exchangeability of independence systems obtained by contraction and deletion. For $Z \subseteq E$, we define the *contraction* $\mathcal{F}/Z$ and *deletion* $\mathcal{F}\backslash Z$ of $\mathcal{F}$ by $\mathcal{F}/Z = \{X \subseteq E\backslash Z \mid X \cup Z \in \mathcal{F}\}$ and $\mathcal{F}\backslash Z = \{X \in \mathcal{F} \mid X \subseteq E\backslash Z\}$.

Lemma 1. *Let $\mathcal{F} \subseteq 2^E$ be an independence system on E. Then for any $Z \subseteq E$, it holds that $\mu(\mathcal{F} \setminus Z), \mu(\mathcal{F}/Z) \leq \mu(\mathcal{F})$.*

For an independence system $\mathcal{F} \subseteq 2^E$, we define two parameters $\nu(\mathcal{F})$ and $\kappa(\mathcal{F})$.

It is known [19] that $\mathcal{F}$ can be represented as $\mathcal{F} = \bigcap_{i=1}^{k} \mathcal{M}_i$ for some k matroids $\mathcal{M}_i$. We denote by $\kappa(\mathcal{F})$ the minimum number of matroids to describe $\mathcal{F}$ as the matroid intersection. For $J \subseteq E$, let $\rho(J)$ and $\gamma(J)$ be the minimum and maximum sizes of maximal independent sets in $\mathcal{F}$ contained in J, respectively. Define the *rank quotient* $\nu(\mathcal{F})$ to be

$$\nu(\mathcal{F}) = \min_{J \subseteq E} \frac{\rho(J)}{\gamma(J)}.$$

Jenkyns [15] and Korte and Hausmann [19] showed the greedy algorithm finds a $\nu(\mathcal{F})$-approximation solution for $\mathrm{P}(\mathcal{F})$. Hassin and Rubinstein [12] proved that the greedy solution is in fact $\nu(\mathcal{F})$-robust.

The three parameters $\mu(\mathcal{F})$, $\nu(\mathcal{F})$, and $\kappa(\mathcal{F})$ have the following relations, which was also mentioned in [1].

Lemma 2. *For an independence system $\mathcal{F} \subseteq 2^E$ with $\mu(\mathcal{F}) \geq 1$, we have*

$$\frac{1}{\nu(\mathcal{F})} \leq \mu(\mathcal{F}) \leq \kappa(\mathcal{F}).$$

We remark that Lemma 2, together with Korte and Hausmann [19], implies that the greedy algorithm provides a $1/\mu(\mathcal{F})$-approximation solution for $\mathrm{P}(\mathcal{F})$, which was also shown in [23].

We note that in many cases the first inequality in Lemma 2 attains the equality.

Lemma 3. *If an independence system $\mathcal{F}$ with $\mu(\mathcal{F}) \geq 1$ satisfies one of the following conditions, then we have $\frac{1}{\nu(\mathcal{F})} = \mu(\mathcal{F})$.*

(i) *it is a matroid,*
(ii) *for a graph $G = (V, E)$, it is the family of matchings in G,*
(iii) *for a complete directed graph $G = (V, E)$ with $|V| \geq 4$, it is the family of unions of vertex disjoint paths $P \subseteq E$,*
(iv) *for a hypergraph $\mathcal{E} \subseteq 2^V$, it is the family of matchings in $\mathcal{E}$.*

By this lemma, we can see that the bound for the robustness by Theorem 1 is *stronger* than the one obtained by using rank quotient $\nu(\mathcal{F})$ in [12]. We finally remark that the gaps in the inequalities in Lemma 2 might be large in general.

Before concluding this section, we show that computing $\mu(\mathcal{F})$ which is useful for the robustness by Theorem 1 is intractable. More precisely, we prove that it is NP-hard to approximate $\mu(\mathcal{F})$ by reducing the maximum stable set problem [14].

Theorem 6. *For an independence system $\mathcal{F}$ on E with $n = |E|$, $\mu(\mathcal{F})$ is not approximable within $n^{1/2-\varepsilon}$ for any $\varepsilon > 0$, unless $P = NP$.*

3 Robust Independence Systems in Boolean Lattice

In this section, we investigate the robustness for independence systems in Boolean lattice. Especially, we present the proof sketches of Theorems 1, 2, and Corollary 1. Let us first consider Theorem 1.

Theorem 7. *Let $\mathcal{F}$ be an independence system on a finite set E and $w \in \mathbb{R}_+^E$ be a weight vector on E. Then, for $b \geq 1$, a w^b-optimal independent set is $\min\{1, 1/\mu(\mathcal{F})^{1/b}, 1/\mu(\mathcal{F})^{1-1/b}\}$-robust with respect to w.*

When b is sufficiently large, a w^b-optimal independent set can be obtained by a greedy algorithm for the original weight w. Thus the theorem implies that a greedy solution is $1/\mu$-robust. Theorem 1 is obtained by maximizing the formula in the theorem, i.e., when $b = 2$. It should be noted that the ratio $1/\sqrt{\mu(\mathcal{F})}$ cannot be improved to $\sqrt{\nu(\mathcal{F})}$ in Theorem 1.

We remark that it is natural to ask whether or not a given μ-exchangeable independence system has an α-robust independent set for a given $\alpha > 1/\sqrt{\mu}$. It is, however, NP-hard even when an independence system is the family of matchings in a bipartite graph [7].

In order to prove Theorem 7, we show Lemma 4 below. For two subsets $X, Y \in \mathcal{F}$, we denote $\mathcal{F}_{X,Y} = \{Z \in \mathcal{F} \mid X \cap Y \subseteq Z \subseteq X \cup Y\}$. We say that a weight vector $w \in \mathbb{R}_+^E$ is (X, Y)-*optimal* if w satisfies

$$w(X) \geq w(Z) \quad \text{for any } Z \in \mathcal{F}_{X,Y}. \tag{2}$$

Lemma 4. *Let $\mathcal{F}$ be an independence system on E, and X, Y be two sets in $\mathcal{F}$ with $X \cap Y = \emptyset$ and $|Y| = k$. If a weight vector $w \in \mathbb{R}_+^E$ is (X, Y)-optimal, then for any β with $0 \leq \beta \leq 1$,*

$$(w^\beta)_{\leq k}(X) \geq \min\left\{1, \frac{1}{\mu^\beta}, \frac{1}{\mu^{1-\beta}}\right\} w^\beta(Y).$$

We will present the proof outline of Lemma 4 in the next subsection. Here we remark that Lemma 4 immediately implies Theorem 7. We henceforth denote $\alpha(\mathcal{F}) = \min\{1, \frac{1}{\mu^\beta}, \frac{1}{\mu^{1-\beta}}\}$.

3.1 The Proof Outline of Lemma 4

Before proving Lemma 4, we first observe the following lemma about the sizes of two independent sets X and Y. For a weight vector w, we denote $\Gamma(w) = \{i \in E \mid w(i) \neq 0\}$.

Lemma 5. *Let X, Y be two independent sets in $\mathcal{F}$ with $X \cap Y = \emptyset$. If a weight vector $w \in \mathbb{R}_+^E$ is (X, Y)-optimal, then we have $\mu(\mathcal{F})\,|X| \geq |\Gamma(w) \cap Y|$.*

Proof. By the definition of $\mu(\mathcal{F})$, there exists a (Y, X)-admissible pair (Z, X) such that $Y \setminus Z \cup X \in \mathcal{F}$ and $|Z| \leq \mu(\mathcal{F})\,|X|$. Since $|X| \leq |Y \setminus Z \cup X|$ and $w(X) \geq w(Y \setminus Z \cup X)$ by (2), we have $w(Y \setminus Z) = 0$. This means $(Y \setminus Z) \cap \Gamma(w) = \emptyset$, and hence we have $\mu(\mathcal{F})\,|X| \geq |\Gamma(w) \cap Y|$. □

We will show Lemma 4 by induction on $|X|$. The following lemma shows that it is true when $|X| = 1$.

Lemma 6. *Let $X \in \mathcal{F}$ with $|X| = 1$, and $Y \in \mathcal{F}$ with $X \cap Y = \emptyset$. Then $w^\beta(X) \geq \alpha(\mathcal{F}) w^\beta(Y)$ for any (X, Y)-optimal w.*

Proof. Let i be the index with $X = \{i\}$. The (X, Y)-optimality implies that $w(i) \geq w(Y)$. If $w(Y) = 0$, i.e., $q = |\Gamma(w) \cap Y| = 0$, then the lemma is clearly true. Otherwise, by maximizing $w^\beta(Y)$ subject to $w(Y) \leq w(i)$ and $q = |\Gamma(w) \cap Y|$, we have $w^\beta(Y) \leq w(i)^\beta q^{1-\beta}$. Since $q \leq \mu(\mathcal{F})$ by Lemma 5, we obtain

$$\frac{w^\beta(i)}{w^\beta(Y)} \geq \frac{w(i)^\beta}{w(i)^\beta q^{1-\beta}} \geq \frac{1}{\mu^{1-\beta}(\mathcal{F})} \geq \alpha(\mathcal{F}). \qquad \square$$

We assume that Lemma 4 is true when $|X| \leq p - 1$ and consider the case in which $|X| = p\,(\geq 2)$. By induction hypothesis, the following two lemmas hold for any (X, Y)-optimal w.

Lemma 7. *If $w(i) = 0$ for some $i \in X$, then we have $(w^\beta)_{\leq k}(X) \geq \alpha(\mathcal{F}) w^\beta(Y)$.*

Lemma 8. *If there exists a set $Z \in \mathcal{F}_{X,Y}$ with $w(X) = w(Z)$, $X \cap Z \neq \emptyset$, and $X \cap Z \subsetneq X$. Then we have $w^\beta_{\leq k}(X) \geq \alpha(\mathcal{F}) w^\beta(Y)$.*

The proof outline of Lemma 8 is as follows. We denote $X_1 = X \cap Z$ and $Y_1 = Y \setminus (Y \cap Z)$. Let $X_2 = X \setminus X_1$ and $Y_2 = Y \setminus Y_1$. Define the two independence systems $\mathcal{F}_1 = \mathcal{F}/Y_2$ and $\mathcal{F}_2 = \mathcal{F}/X_1$. We then know that w is (X_1, Y_1)-optimal with respect to $\mathcal{F}_1$, and that w is (X_2, Y_2)-optimal with respect to $\mathcal{F}_2$. Therefore, by applying the induction hypothesis to $\mathcal{F}_1$ and $\mathcal{F}_2$, together with Lemma 1, we obtain Lemma 8.

For a vector $u \in \mathbb{R}^E_+$, we define a function $f_X : \mathbb{R}^E_+ \to \mathbb{R}_+$ to be $f_X(u) = (u^\beta)_{\leq k}(X)$. Given an (X, Y)-optimal vector $w \in \mathbb{R}^E_+$, let w^* be an (X, Y)-optimal vector such that $f_X(w^*)$ is minimum over $w^*(i) = w(i)$ for $i \in E \setminus X$. Note that such a w^* exists, because the feasible region represented by linear inequalities is nonempty, w^* is nonnegative, and $f_X(u)$ is continuous and nonnegative. We call such w^* a *minimizer* of f_X.

By Lemmas 7 and 8, we may assume that w^* satisfies

$$\begin{aligned} w^*(X) &> w^*(Z) \quad \text{for any } Z \in \mathcal{F}_{X,Y} \text{ with } X \cap Z \neq \emptyset \text{ and } X \cap Z \subsetneq X, \\ w^*(X) &\geq w^*(Z) \quad \text{for any } Z \in \mathcal{F}_{X,Y} \text{ with } X \subseteq Z \text{ or } X \cap Z = \emptyset, \\ w^*(i) &> 0, \qquad \text{for any } i \in X. \end{aligned} \tag{3}$$

Note that the second inequality is equivalent to $w^*(X) \geq w^*(Y)$ and $w^*(X) \geq w^*(Z)$ for any $Z \in \mathcal{F}_{X,Y}$ with $X \subseteq Z$. Furthermore, it follows that $w^*(X) = w^*(Y)$ by the minimality of w^*. We denote $W = w^*(X) = w^*(Y)$.

For a minimizer w^* of f_X satisfying (3), we have the following lemma. This lemma follows from the concavity of the β-th power of numbers in f_X.

Lemma 9. *Assume that a minimizer w^* of f_X satisfies (3). Then $w^*(j) = W/p$ holds for $j \in X$.*

We then prove the following lemma, which completes the proof of Lemma 4. This lemma is shown in a similar way to Lemma 6.

Lemma 10. *If* $w^*(j) = W/p$ *for* $j \in X$, *then it holds that* $(w^\beta)_{\leq k}(X) \geq \alpha(\mathcal{F})w^\beta(Y)$.

3.2 The Proof of Theorem 2

This section concludes with the proof of Theorem 2. Let p denote an integer with $p \geq 2$. For $i = 1, \ldots, p$, let $V_i = \{v_1^i, \ldots, v_p^i\}$, and $V = \bigcup_{i=1}^p V_i$. By definition, $|V| = p^2$. Let $\mathcal{E} \subseteq 2^V$ be the hypergraph defined as $\mathcal{E} = \{e_0, e_1, \ldots, e_p\}$, where $e_0 = (v_1^1, v_2^2, \ldots, v_p^p)$ and $e_j = (v_j^1, \ldots, v_j^p)$ for $j = 1, \ldots, p$. Let $\mathcal{F}$ be the family of matchings in $\mathcal{E}$. It follows that $\mu(\mathcal{F}) = p$. We define a weight $w \in \mathbb{R}_+^E$ as $w(e_0) = \sqrt{p}$ and $w(e_j) = 1$ for $j = 1, \ldots, p$.

We can see that $\mathcal{F}$ has exactly two maximal independent sets $I = \{e_1, \ldots, e_p\}$ and $J = \{e_0\}$. For a positive integer k, J is a w-optimal k-independent set if $k \leq \lfloor \sqrt{p} \rfloor$, and so is $\{e_j \mid j = 1, \ldots, k\}$ if $k > \sqrt{p}$. Hence, for any $\alpha > 1/\sqrt{p}$, J is not α-robust, since $w_{\leq p}(J)/w(I) = 1/\sqrt{p}$. Similarly, I is not α-robust, since $w_{\leq 1}(I)/w(J) = 1/\sqrt{p}$. Thus no independent set is α-robust. This proves Theorem 2.

Note that the example above also shows the tightness for Corollary 1 (ii), (iv) and (v), since $\mathcal{F}$ can be regarded as the family of matchings (if $p = 2$), hypergraph matchings, and p-matroid intersection. Corollary 1 (i) is clearly tight. We can also construct tight examples for (iii), (vi), (vii) and (viii). See [18] in details.

References

1. Calinescu, G., Chekuri, C., Pal, M., Vondrak, J.: Maximizing a submodular set function subject to a matroid constraint. SIAM Journal on Computing (to appear)
2. Chan, Y.H., Lau, L.C.: On linear and semidefinite programming relaxations for hypergraph matching. In: Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2010), pp. 1500–1511 (2010)
3. Chandra, B., Halldórsson, M.: Greedy local improvement and weighted set packing approximation. Journal of Algorithms 39, 223–240 (2001)
4. Edmonds, J.: Matroids and the greedy algorithm. Mathematical Programming 1, 127–136 (1971)
5. Fisher, M.L., Nemhauser, G.L., Wolsey, L.A.: An analysis of approximations for maximizing submodular set functions ii. Mathematical Programming Study 8, 73–87 (1978)
6. Frank, A.: How to make a digraph strongly connected. Combinatorica 1, 145–153 (1981)
7. Fujita, R., Kobayashi, Y., Makino, K.: Robust matchings and matroid intersections. In: de Berg, M., Meyer, U. (eds.) ESA 2010. LNCS, vol. 6347, pp. 123–134. Springer, Heidelberg (2010)
8. Fukunaga, T., Halldórsson, M., Nagamochi, H.: Robust cost colorings. In: Proceedings of the 19th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2008), pp. 1204–1212 (2008)

9. Gabow, H.N.: An efficient reduction technique for degree-constrained subgraph and bidirected network flow problems. In: Proceedings of the 15th ACM Symposium on Theory of Computing (STOC 1983), pp. 448–456 (1983)
10. Goemans, M.X., Harvey, N.J.A., Iwata, S., Mirrokni, V.: Approximating submodular functions everywhere. In: Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2009), pp. 535–544 (2009)
11. Grötschel, M., Lovász, L., Schrijver, A.: Geometric Algorithms and Combinatorial Optimization, 2nd edn. Springer, Heidelberg (1993)
12. Hassin, R., Rubinstein, S.: Robust matchings. SIAM Journal on Discrete Mathematics 15, 530–537 (2002)
13. Hassin, R., Segev, D.: Robust subgraphs for trees and paths. ACM Transaction on Algorithms 2, 263–281 (2006)
14. Hásted, J.: Clique is hard to approximate within $n^{1-\varepsilon}$. Acta Mathematica 182, 105–142 (1999)
15. Jenkyns, T.A.: The efficacy of the "greedy" algorithm. In: Proceedings of the 7th Southeastern Conference on Combinatorics, Graph Theory and Computing, pp. 341–350 (1976)
16. Jenkyns, T.A.: The greedy traveling salesman's problem. Networks 9, 363–373 (1979)
17. Jungnickel, D.: Graphs, Networks, and Algorithms, 2nd edn. Algorithms and Computation in Mathematics, vol. 5. Springer, Heidelberg (2002)
18. Kakimura, N., Makino, K.: Robust independence systems, Mathematical Engineering Technical Reports METR 2011-14, University of Tokyo (2011)
19. Korte, B., Hausmann, D.: An analysis of the greedy heuristic for independence systems. Annals of Discrete Mathematics 2, 65–74 (1978)
20. Korte, B., Vygen, J.: Combinatorial Optimization: Theory and Algorithms. Springer, Heidelberg (2006)
21. Lin, G., Nagarajan, C., Rajarama, R., Williamson, D.: A general approach for incremental approximation and hierarchical clustering. SIAM Journal on Computing 39, 3633–3669 (2010)
22. Lucchesi, C.L.: A Minimax Equality for Directed Graphs, PhD thesis, University of Waterloo (1976)
23. Mestre, J.: Greedy in approximation algorithms. In: Azar, Y., Erlebach, T. (eds.) ESA 2006. LNCS, vol. 4168, pp. 528–539. Springer, Heidelberg (2006)
24. Mettu, R.R., Plaxton, C.G.: The online median problem. SIAM Journal on Computing 32, 816–832 (2003)
25. Rado, R.: Note on independence relations. Proceedings of the London Mathematical Society 7, 300–320 (1957)
26. Schrijver, A.: Combinatorial Optimization — Polyhedra and Efficiency. Springer, Heidelberg (2003)

Buyback Problem - Approximate Matroid Intersection with Cancellation Costs

Ashwinkumar Badanidiyuru Varadaraja

Cornell University, Ithaca, NY
ashwin85@cs.cornell.edu

Abstract. In the buyback problem, an algorithm observes a sequence of bids and must decide whether to accept each bid at the moment it arrives, subject to some constraints on the set of accepted bids. Decisions to reject bids are irrevocable, whereas decisions to accept bids may be canceled at a cost that is a fixed fraction of the bid value. Previous to our work, deterministic and randomized algorithms were known when the constraint is a matroid constraint. We extend this and give a deterministic algorithm for the case when the constraint is an intersection of k matroid constraints. We further prove a matching lower bound on the competitive ratio for this problem. This problem has applications to banner advertisement, semi-streaming, routing, load balancing and other problems where preemption or cancellation of previous allocations is allowed.

1 Introduction

Consider the online problem of resource allocation in which preemption is allowed. This kind of problem has been heavily studied in a wide variety of settings which range from advertisement allocations to routing to load balancing. In online weighted resource allocation without preemption we cannot get any nontrivial worst case guarantee on the sum of weights allocated. Consider the simplest problem of choosing the maximum number in a sequence. Any deterministic or randomized algorithm cannot have a constant competitive ratio up to any factor. Usually this impossibility is circumvented in the literature by placing some restrictions. This could be by allowing the input to be either a random permutation [6,5,13,26,24] or drawn iid from some probability distribution [21,8]. Other approaches [4,9,18,1,7,3,19,20] which do not relax any conditions on the input assume that either preemption is allowed or preemption with a penalty is allowed and give guarantees for every input. In this paper we study this kind of relaxation.

Consider the following generic problem. There is a set system $\mathcal{I}$ (downward closed) for the ground set $\mathcal{E}$. Elements from $\mathcal{E}$ are presented to the algorithm in a sequential manner. Each element e_i is also associated with a value w_{e_i}. When element e_i is presented to the algorithm it must be accepted or rejected immediately. When e_i is accepted the algorithm could cancel (preempt) some of the previously accepted elements. If S denotes the set currently accepted, then

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 379–390, 2011.

the constraint for the algorithm is to have $S \in \mathcal{I}$. The utility of the algorithm is the value of the accepted elements minus the penalty paid to the canceled elements. All the canceled elements are paid a penalty proportional to their corresponding value. We present some applications of this generic problem below.

Banner advertisement. The buyback problem was first defined and studied in [4,9]. Specifically they give deterministic algorithms for the case when $\mathcal{I}$ is a matroid. This was later extended by [2] which gave a randomized algorithm with better competitive ratio. Consider an advertisement system for a single advertisement slot. In certain systems bidding for this slot starts well in advance. In such a system bidders come and bid in an online manner. The system accepts the bids or rejects them immediately. The system could later accept much higher bids and cancel previously accepted ones. But this causes a loss for the previously accepted bidders. Hence the system pays them back with a penalty. The work in [4,9,2] also generalizes to much more general systems where the accepted bids can form a matroid. They leave open the question of finding algorithms for more general constraints. One of the key constraints not modeled by this is when each bidder desires a single item among a set of items, i.e when $\mathcal{I}$ is a valid matching in a bipartite graph. Our result in section 2 solves this as well as a generalization to k arbitrary matroid constraints. We also prove matching lower bounds in section 4. It is important to note that we ignore the incentives throughout our work and assume that bids/values are truthfully reported.

Free Disposal. Consider the problem of online ad allocation with free disposal studied in [18]. Here we have a set of advertisers A known in advance together with an integer impression contract $n(a)$ for each advertiser $a \in A$. $n(a)$ denotes the maximum number of impressions for which advertiser can be charged. When an impression $i \in I$ arrives the utility of this impression w_{i_a} for every advertiser a is also revealed. The final utility of the algorithm is the total amount charged to each user. We note that there is a very straightforward reduction from this problem to the problem we define, specifically from the case when $k = 2$ and zero penalty for preemption[1]. Our algorithm gives a 5.828 competitive ratio. [18] gives an unconditional 2 competitive and conditional $e/(e-1)$ competitive algorithm. Note that our algorithm is tight for the generic problem defined due to the lower bound shown in section 4. [18] is able to give better competitive ratio as this problem is more restrictive than the generic problem we define.

Semi Streaming. Recently a few papers [28,16,12,10,11,17] have studied graph problems in the semi-streaming model. Consider the problem of finding a weighted matching in a stream which uses $\tilde{O}(n)$ memory, $O(1)$ update time and 1 pass. [17] introduced this problem and gave a factor 6 approximation. [28] improved this to 5.828 approximate semi-streaming algorithm. Both of these algorithms are characterized by the fact that they always maintain a valid matching. Hence our lower bound in section 4 proves that among the class of algorithms which maintain only edges of a valid matching no algorithm can achieve better than approximation ratio 5.828. This improves the 4.967 lower bound proved in [14].

[1] The system could allocate more than $n(a)$ impressions and charge for the top $n(a)$.

Routing with Preemption and Load Balancing. There has been a huge literature [1,7,3,19,20] on routing in networks and load balancing where preemption of previously allocated resources in allowed. Many of these results studied can be very succinctly generalized by the following problem.

Consider an arbitrary downward closed set system $\mathcal{I}$. Elements $e_i \in \mathcal{I}$ are presented to the algorithm along with their weight and the sets to which they belong. The algorithm has to either accept or reject the element immediately. An accepted element can be preempted or canceled later but canceled/rejected elements cannot be taken later. It is a simple excercise to show that such a generic problem does not admit a competitive ratio better than $n-1$ even when the penalty is 0. One should note that the papers which study routing with preemption and load balancing are able to achieve better competitive ratio by exploiting some additional structure in the problem.

The offline problem of intersection of k matroids was introduced and studied in [23,25]. The problem has also been studied under more general submodular utility functions in the CS theory/OR literature. Some of the recent papers in this direction are [15,29,27].

2 Preliminaries

First we define the problem formally.

2.1 Model

Consider a ground set of elements $\mathcal{E} = \{e_1, e_2, \ldots, e_n\}$. Let $\mathcal{M}_1, \ldots, \mathcal{M}_k$[2] be k arbitrary matroid constraints on $\mathcal{E}$. Let the corresponding subset system be $\mathcal{I}_1, \ldots, \mathcal{I}_k$ and let $\mathcal{I} = \cap_{j=1}^{k} \mathcal{I}_j$. We define the online problem with the following constraints.

1. The elements of $\mathcal{E}$ are presented to the algorithm in some arbitrary order. The value w_{e_i} of element e_i and the matroid constraints it is involved with are revealed to the algorithm when the element is presented to it.
2. When element e_i is presented it must be accepted or rejected immediately. Additionally it could be canceled at a later point in time. When an element is canceled the algorithm must pay a penalty $f \cdot w_{e_i}$ where f is a constant. (Note the difference between reject and cancel.)
3. Let $\mathcal{A}$ be the set of elements accepted and $\mathcal{R}$ be the set of elements accepted and later canceled. Then the utility of the algorithm is defined as $\sum_{e \in \mathcal{A}} w_e - (1+f) \sum_{e \in \mathcal{R}} w_e$. Note that all elements in $\mathcal{R}$ are also counted in $\mathcal{A}$. Moreover the currently maintained set $S = \mathcal{A} - \mathcal{R}$ must be an independent set, i.e. $S \in \mathcal{I}$.

Here we desire to find a competitive online algorithm with the above constraints.

[2] A matroid $\mathcal{M}_i = (\mathcal{E}, \mathcal{I}_i)$ is constructed from a ground set $\mathcal{E} \neq \phi$ and a nonempty family of subsets of $\mathcal{E}$, called the independent subsets of $\mathcal{E}$, such that if $B \in \mathcal{I}_i$ and $A \subseteq B$ then $A \in \mathcal{I}_i$ ($\mathcal{I}_i$ is hereditary). Additionally, if $A, B \in \mathcal{I}_i$ and $|A| < |B|$, then there is some element $x \in B - A$ such that $A \cup \{x\} \in \mathcal{I}_i$ (exchange property).

Initialize $S = \emptyset$.
for all elements e_i, in order of arrival, **do**
  if $S \cup \{e_i\} \in \mathcal{I}$ **then**
    $S = S \cup \{e_i\}$
  else
    for all $1 \leq j \leq k$ **do**
      if $S \cup \{e_i\} \notin \mathcal{I}_j$ **then**
        e_{i_j} be the element of smallest value such that $S \cup \{e_i\} \setminus \{e_{i_j}\} \in \mathcal{I}_j$
      else
        $e_{i_j} = NULL$.
      end if
    end for
    Let $C_{e_i} = \cup_{j=1}^{k} \{e_{i_j}\}$
    if $w_{e_i} \geq r \cdot (\sum_{j=1}^{k} w_{e_{i_j}})$ **then**
      $S = S \cup \{e_i\} \setminus \cup_{j=1}^{k} \{e_{i_j}\}$
    end if
  end if
end for

Fig. 1. Algorithms for k matroids intersection

2.2 Algorithm

The algorithm is shown in Figure 1 as Algorithm 1. At each step the algorithm maintains an independent set S. Assume it sees the element e_i at some step. If $S \cup \{e_i\}$ is also an independent set, then it includes $\{e_i\}$ into the current set S. Otherwise $S \cup \{e_i\}$ has a circuit in some of the matroids $\mathcal{I}_j$. It first finds the minimum value element (e_{i_j}) it must remove in set S to make $S \cup \{e_i\}$ an independent set in each of $\mathcal{I}_j$. Now suppose $w_{e_i} \geq r \cdot (\sum_{j=1}^{k} w_{e_{i_j}})$, then it includes the element e_i and discards the elements $\cup_{j=1}^{k} \{e_{i_j}\}$. We will prove that the above algorithm is $\frac{(k \cdot r-1)r}{r-1-f}$ competitive. Here r is a constant defined later to optimize the competitive ratio.

3 Analysis of the Algorithm

Let $S(i)$ be the set S at the end of step i and let $OPT \subseteq \mathcal{E}$ be optimal solution to the weighted intersection of k matroids. The main part of competitive analysis is based on the following lemma.

Lemma 1. *$w(S(n))\frac{(k \cdot r-1)r}{r-1} \geq w(OPT)$ where $w(S(n)) = \sum_{e \in S(n)} w_e$.*

We will prove Lemma 1 in section 3.1. The proof is based on a charging scheme. For now we just assume it to analyze the competitive ratio of the algorithm.

Theorem 1. *The online algorithm with cancellations for k matroid constraints has a competitive ratio $c = \frac{(k \cdot r-1)r}{r-1-f}$. This ratio is minimized when $\frac{r}{1+f} = 1 + \sqrt{1 - \frac{1}{k(1+f)}}$ and has a value $c = k(1+f)(1 + \sqrt{1 - \frac{1}{k(1+f)}})^2$*

The competitive ratio of our algorithm matches the case $k = 1$ given in [4,9]. Later in section 4 we will show that this is tight for every k.

Proof. The utility of the algorithm consists of two terms. One is due to the utility of $S(n)$ and the other is the penalty due to the canceled set R.

- For each element e_i we define a value $P(e_i)$ recursively. If e_i was accepted in step 3 or was never accepted, then $P(e_i) = 0$. Else if elements $C_{e_i} = \{e_{i_1}, e_{i_2}, \ldots, e_{i_k}\}$ were canceled, then $P(e_i) = f \cdot \sum_{j=1}^{k} w_{e_{i_j}} + \sum_{j=1}^{k} P(e_{i_j})$. Now each canceled item e_{i_j}'s penalty is accounted to the item e_i which canceled it. We prove that for any element e_i the total penalty accounted is less than or equal to $\frac{f}{r-1} \cdot w_{e_i}$. The proof is by induction. The base case when $P(e_i) = 0$ is simple. The inductive case is as follows.

$$\begin{aligned} P(e_i) &= f \cdot \textstyle\sum_{j=1}^{k} w_{e_{i_j}} + \sum_{j=1}^{k} P(e_{i_j}) \\ &\leq f \cdot \textstyle\sum_{j=1}^{k} w_{e_{i_j}} + \sum_{j=1}^{k} f \cdot \frac{w_{e_{i_j}}}{r-1} \\ &\leq f \cdot \tfrac{w_{e_i}}{r} + f \cdot \tfrac{w_{e_i}}{r(r-1)} \\ &= f \cdot \tfrac{w_{e_i}}{r-1} \end{aligned} \tag{1}$$

  Hence the total penalty is at most $\sum_{e_i \in S(n)} f \cdot \frac{w_{e_i}}{r-1} = \frac{f \cdot w(S(n))}{r-1}$
- The final weight of the set $S(n)$ is bounded by Lemma 1. Combining the two parts we get the total utility of the algorithm.

$$\begin{aligned} \text{Utility} &\geq w(S(n)) - f \cdot \tfrac{w(S(n))}{r-1} \\ &= \tfrac{r-1-f}{r-1} \cdot w(S(n)) \\ &\geq \tfrac{r-1-f}{(k \cdot r-1)r} \cdot w(OPT) \quad \text{(Using Lemma 1)} \end{aligned} \tag{2}$$

Hence we get competitive ratio of $c = \frac{(k \cdot r-1)r}{r-1-f}$. Optimizing over r we get $\frac{r}{1+f} = 1 + \sqrt{1 - \frac{1}{k(1+f)}}$ and $c = k(1+f)(1 + \sqrt{1 - \frac{1}{k(1+f)}})^2$. □

3.1 Charging Scheme

Here we will prove Lemma 1. This portion is technically the hardest part of the paper and requires developing a new charging scheme. This is aided by a graph construction. We first give some notation. Each element carries two kinds of charges (ch_1 and ch_2). Let at any step j, $\mathsf{ch}(e_i, j) = \mathsf{ch}_1(e_i, j) + \mathsf{ch}_2(e_i, j)$. Additionally let $C_{e_i} = \cup_{j=1}^{k} e_{i_j}$ (the set of elements discarded when e_i is included) be as defined in step 13 of Algorithm 1. Let $S(j)$ denote the set S after step j. Let $\mathsf{ch}(S', j) = \sum_{e \in S'} \mathsf{ch}(e, j)$ (analogously for ch_1 and ch_2).

Sketch. We start with a total charge of OPT on the elements. $\mathsf{ch}_1(e_i, j)$ denotes the charge which the element carries from the beginning. $\mathsf{ch}_2(e_i, j)$ denotes the

charge which the element gets from some other elements. At any step either e_i or C_{e_i} is discarded. When any element is discarded all its charge is added to $S(i)$ (or $S(j)$ for $j \geq i$). There by all the charge is stored in $S(n)$. Next we bound the amount of charge any element can carry.

There are two ways charge is transferred. One way is for ch_1 and another for ch_2. At step i if C_{e_i} is discarded, then we always transfer $\mathsf{ch}_2(C_{e_i}, i-1)$ to $\mathsf{ch}_2(e_i, i)$. Another way of transfer is for ch_1. This is done by a fairly sophisticated graph construction. Essentially we construct k bipartite graphs and then prove that each one has a matching that matches all vertices on the left side of the bipartition using Hall's Theorem. Suppose e_i is matched to e_j, we transfer $\mathsf{ch}_1(e_i, a)$ to $\mathsf{ch}_2(e_j, b)$. Here a denotes the step at which e_i was removed and b denotes the step before which e_j was removed (n otherwise). Note that we will need causal consistency ($a < b$) for this.

The charges at the beginning of the algorithm are defined as follows.

- if $e_i \in OPT$ then $\mathsf{ch}(e_i, 0) = \mathsf{ch}_1(e_i, 0) = w_{e_i}$ and $\mathsf{ch}_2(e_i, 0) = 0$
- if $e_i \notin OPT$ then $\mathsf{ch}(e_i, 0) = \mathsf{ch}_1(e_i, 0) = \mathsf{ch}_2(e_i, 0) = 0$.

Before defining the charging scheme we define a graph construction which will aid us in the charging scheme.

Graph Construction. Construct k bipartite graphs as the algorithm proceeds. Here the p^{th} graph corresponds to the p^{th} matroid. Let $P_1(p)$ denote partite set 1 of p^{th} graph and $P_2(p)$ denote partite set 2 of p^{th} graph. Additionally let $N_p(S)$ denote the set of neighbors of $S \subseteq P_1(p)$ in p^{th} graph. Let $rank_p(S)$ be the rank of set S in the p^{th} matroid.

1. The bipartite graph starts empty and edges are added. Each endpoint of an edge corresponds to an element e_i. The node corresponding to an element e_i exists only when the corresponding edge is added and removed when all its adjacent edges are deleted. An edge in the graph corresponds to a potential ch_1 transfer.
2. Consider step 14 in the algorithm. If $w_{e_i} < r \cdot \sum_{j=1}^{k} w_{e_{i_j}}$, then e_i is not included in S. Now if $e_i \in OPT$, then add a node e_i to $P_1(p)$ (for each p). Let $Ckt(e_i, p)$ be the unique circuit in p^{th} matroid in $S \cup \{e_i\}$. Then add edge e_i, e_j for each $e_j \in Ckt(e_i, p) - \{e_i\}$ with e_j belonging to $P_2(p)$.
3. Consider step 14 in the algorithm. If $w_{e_i} \geq r \cdot \sum_{j=1}^{k} w_{e_{i_j}}$, then $C_i = \{e_{i_1}, \ldots, e_{i_k}\}$ is deleted from S and e_i is included into it. Delete each e_{i_p} from the corresponding $P_2(p)$[3]. For each existing edge e_q, e_{i_p} add edges e_q, e_j for each $e_j \in Ckt(e_i, p) - \{e_{i_p}\}$ with e_j belonging to $P_2(p)$. Additionally if $e_{i_p} \in OPT$ (i.e $\mathsf{ch}_1(e_{i_p}, 0) > 0$), then readd it to $P_1(p)$. Add edges e_{i_p}, e_j for each $e_j \in Ckt(e_i, p) - \{e_{i_p}\}$.

Lemma 2. *The graph construction has the following properties.*

1. $P_1(p) \subseteq OPT - S(n)$ *for each* p.

[3] Note that e_{i_p} is deleted only from $P_2(p)$ and not from $P_2(p')$ for $p' \neq p$.

2. $\forall \hat{S} \subseteq P_1(p)$,$N_p(\hat{S})$ *spans* $\hat{S} \subseteq P_1(p)$ *in* p^{th} *matroid.*
3. $\forall \hat{S} \subseteq P_1(p)$,$|N_p(\hat{S}) - OPT| \geq |\hat{S}|$.
4. *There exists a matching in graph* p *such that every* $e \in P_1(p)$ *is matched to a node in* $P_2(p) - OPT$.
5. *Any element* $e \in \mathcal{E} - S(n) - OPT$ *is matched in at most* $k-1$ *of the graphs from the side of* P_2.

Proof. 1. This is easily seen by construction. In steps 2 and 3 of Graph construction an element is added to $P_1(p)$ precisely when it is removed from S and when it belongs to OPT.

2. When a node e_j is added to $P_1(p)$ then edges to each element in $Ckt - \{e_j\}$ are added. Hence any node e_j is spanned by $N_p(\{e_j\})$. By matroid property this implies that for any set $\hat{S} \subseteq P_1(p)$ we have that $N_p(\hat{S})$ spans $\hat{S}$. [4]
3. Let $W = N_p(\hat{S}) \cap OPT$. We now assert some statements from which the inequality easily follows.
   - $rank_p((N_p(\hat{S}) - OPT) \cup W) \geq rank_p(\hat{S} \cup W)$[5]. This follows from property 2.
   - $rank_p(\hat{S} \cup W) = rank_p(\hat{S}) + rank_p(W) = |\hat{S}| + |W|$. This follows from the fact that $\hat{S}$ and W are disjoint and $\hat{S} \cup W \subseteq OPT$.
   - $rank_p((N_p(\hat{S}) - OPT) \cup W) \leq rank_p(N_p(\hat{S}) - OPT) + rank_p(W) \leq |N_p(\hat{S}) - OPT| + |W|$. These set of inequalities follows from the matroid property.

   Combining the above equations we get $|\hat{S}| \leq |N_p(\hat{S}) - OPT|$.
4. Follows from Hall's Theorem and property 3. If $e_i \in P_1(p)$ is matched to e_j, then let $e_j = M_p(e_i)$.
5. This follows from step 3 of graph construction. Here any element removed from $S(i)$ in any step is deleted from $P_2(p)$ (for one of the p's). Hence such an element could belong to P_2 of at most $k-1$ graphs and be matched at most $k-1$ times (from P_2's side). Note that any element $e \in OPT - S(n)$ could additionally be matched from the P_1's side. □

Charge Transfer. We finally explain the exact way the charge is transferred in each step.

1. Consider step 14 in the algorithm. If $w_{e_i} \geq r \cdot (\sum_{j=1}^{k} w_{e_{i_j}})$, then transfer all of $\mathsf{ch}_2(e_{i_j}, i-1)$ for each e_{i_j} to $\mathsf{ch}_2(e_i, i)$.
2. Consider step 14 in the algorithm. If $w_{e_i} \geq r \cdot (\sum_{j=1}^{k} w_{e_{i_j}})$, then let $C_{e_i} = \{e_{i_1}, \ldots, e_{i_k}\}$ be deleted from S and e_i is included into it. If e_{i_l} was added to $P_1(l)$, then transfer all of $\mathsf{ch}_1(e_{i_l}, 0)$ to $\mathsf{ch}_2(M_l(e_{i_l}), t)$ (where t is either the step $M_l(e_{i_l})$ is deleted or n). Note that in each transfer $M_l(e_{i_l})$ gets a $ch1$ transfer of atmost its value (Since e_{i_l} is the min weight element in the ckt).

[4] Note that even though the edges could later be deleted, the span property still holds due to additional edges being added.

[5] $rank_p(S)$ is defined as the largest subset $A \subseteq S$ such that $A \in \mathcal{I}_p$.

3. Consider step 14 in the algorithm. Let $w_{e_i} < r \cdot (\sum_{j=1}^{k} w_{e_{i_j}})$. Additionally if $e_i \in OPT$, then it would have been added to $P_1(l)$ of each graph l. Now e_i is matched to different nodes in different graphs. Transfer a portion of $\mathsf{ch}_1(e_i, 0)$ to $\mathsf{ch}_2(M_l(e_i), t)$ which is proportional to $w_{e_{i_l}}$ for each graph l. (where t is either the step $M_l(e_i)$ is deleted or n). Note that in each transfer $M_l(e_i)$ gets a $ch1$ transfer of atmost r times its value.
4. Note that the above transfer of charges does not violate causal consistency as the transfer of charge happens from e to some element in $S(i)$ of the future.

We finish the proof of Lemma 1 by analyzing the charge transfer. First note that any element in $\mathcal{E} - S(n) - OPT$ receives ch_1 transfer in step 2 or 3 of charge transfer at most $k-1$ times. This is by property 5 of Lemma 2. Additionally we can also see that each ch_1 transfer to element e_i is at most $r \cdot w_{e_i}$. Using these properties by induction that we prove that $\mathsf{ch}_2(e_i, j) \leq \frac{(k-1)r^2}{r-1} w_{e_i}$ for $e_i \in \mathcal{E} - S(n)$ and $\mathsf{ch}(e_i, j) \leq \frac{(k \cdot r-1)r}{r-1} w_{e_i}$ for $e_i \in S(n)$.

– For any element in $e_i \in \mathcal{E} - S(n)$ we have $\mathsf{ch}_2(e_i, j) \leq \frac{(k-1)r^2}{r-1} w_{e_i}$ if it was deleted at step j. Note that ch_1 transfer happens at most $k-1$ times for $e_i \in \mathcal{E} - S(n) - OPT$ and 0 times for $e_i \in \mathcal{E} - S(n) \cap OPT$

$$\begin{aligned} \mathsf{ch}_2(e_i, j) &\leq (k-1) \cdot (\mathsf{ch}_1 \text{ transfer}) + \textstyle\sum_{j=1}^{k} \mathsf{ch}_2(e_{i_j}, i-1) \\ &\leq (k-1) \cdot r \cdot w_{e_i} + \textstyle\sum_{j=1}^{k} \frac{(k-1)r^2}{r-1} w_{e_{i_j}} \\ &\leq (k-1) \cdot r \cdot w_{e_i} + \frac{(k-1)r^2}{r-1} \frac{w_{e_i}}{r} \\ &= \frac{(k-1)r^2}{r-1} w_{e_i} \end{aligned} \tag{3}$$

– For any element in $S(n) - OPT$ we have $\mathsf{ch}(e_i, n) \leq \frac{(k \cdot r-1)r}{r-1} w_{e_i}$. Note that this is also true for any element in $S(n) \cap OPT$, as they do not get any ch_1 transfer but have a non-zero ch_1.

$$\begin{aligned} \mathsf{ch}(e_i, n) &\leq k \cdot (\mathsf{ch}_1 \text{ transfer}) + \textstyle\sum_{j=1}^{k} \mathsf{ch}_2(e_{i_j}, i-1) \\ &\leq k \cdot r \cdot w_{e_i} + \textstyle\sum_{j=1}^{k} \frac{(k-1)r^2}{r-1} w_{e_{i_j}} \\ &\leq k \cdot r \cdot w_{e_i} + \frac{(k-1)r^2}{r-1} \frac{w_{e_i}}{r} \\ &= \frac{(k \cdot r-1)r}{r-1} w_{e_i} \end{aligned} \tag{4}$$

The above argument proves that

– $\forall e_i \in S(n)$, $\mathsf{ch}(e_i, n) \leq \frac{(k \cdot r-1)r}{r-1} w(e_i)$
– $\sum_{e_i \in S(n)} \mathsf{ch}(e_i, n) = w(OPT)$. This follows naturally from charge conservation in the system.

The proof of Lemma 1 can be easily seen from the above two properties.

4 Lower Bound

In this section we prove a matching lower bound of $c = k(1+f)(1+\sqrt{1-\frac{1}{k(1+f)}})^2$.

Theorem 2. *Any deterministic online algorithm $\mathcal{A}$ cannot achive a competitive ratio of $\beta < c = k(1+f)(1+\sqrt{1-\frac{1}{k(1+f)}})^2$ for the problem of online buyback problem with k matroid constraints.*

Sketch. Assume $\mathcal{A}$ is an online deterministic algorithm which achieves a competitive ratio $\beta < c$. Then we arrive at a contradiction. The proof will be in following steps.

1. We will construct a k-dimensional matching. Using this we will argue the existence of an infinite sequence $X = \{x_1, x_2, \ldots\}$ of the following form. $x_1 = 1$ and $x_i \geq 0, \forall i$. Additionally they will satisfy the following inequality.

$$\beta(x_i - f \cdot \sum_{j=1}^{i-1} x_j) \geq x_{i+1} + (k-1)\sum_{j=1}^{i+1} x_j, \forall i \geq 1 \tag{5}$$

2. Consider any sequence $X = \{x_1, x_2, \ldots\}$ $(x_1 = 1)$ which satisfies $x_i \geq 0, \forall i$ and $\beta(x_i - f \cdot \sum_{j=1}^{i-1} x_j) \geq x_{i+1} + (k-1)\sum_{j=1}^{i+1} x_j, \forall i$. Now if $\beta < c$, we arrive at a contradiction.

The proof of the Theorem 2 follows from the above two steps. We will prove the second part first.

4.1 Contradiction

Consider all sequences S of the form $x_1 = 1$,$x_i \geq 0, \forall i$ and satisfying equation 5. For any given sequence X, let $n(X)$ be the minimum i for which inequality 5 is strict. We first claim that if a sequence of the above form exists, then as X ranges over S, $n(X)$ should take unboundedly large values. Assume by contradiction that this claim is not true. Among the sequences consider sequence X for which $n(X) = N$ is as large as possible. We construct a sequence for which $n(X)$ is even larger thus arriving at a contradiction. Let λ be defined as follows.

$$\lambda = \frac{\beta(x_N - f \cdot \sum_{j=1}^{N-1} x_j) - (k-1)\sum_{j=1}^{N} x_j}{k \cdot x_{N+1}} \tag{6}$$

Let $X' = \{x_1, x_2, \ldots, x_N, \lambda x_{N+1}, \lambda_{N+2} x_{N+2}, \ldots\}$. Then it is easy to see that inequality 5 and other constraints are met. Additionally $n(X') > n(X)$. Hence we arrive at a contradiction. Let $Y = \{y_1, y_2, \ldots\}$ be the sequence defined as follows.

$$y_1 = 1, \qquad \beta(y_i - f \cdot \sum_{j=1}^{i-1} y_j) = y_{i+1} + (k-1)\sum_{j=1}^{i+1} y_j, \forall i \geq 1 \tag{7}$$

Then $y_i \geq 0, \forall i$. This is because by our previous claim $\exists$ a sequence $X \in S$ where $y_i = x_i \geq 0$. Let $z_i = \sum_{j=1}^{i} y_i$. Then we get the recurrence $k \cdot z_{i+1} = (1+\beta)z_i - \beta(1+f)z_{i-1}$ and $z_1 = 1$. As each $y_i \geq 0$ we also have that $z_i \geq 0$. Now we use a Lemma from [22] to get a contradiction.

Lemma 3. *[22] Let $u_n = au_{n-1} + bu_{n-2}$ be a linear recurrence of second order. If $p(x) = x^2 - ax - b$ has imaginary roots, then the sequence must have negative elements.*

Consider the case when $\beta < k(1+f)(1+\sqrt{1-\frac{1}{k(1+f)}})^2$. Then it is simple to see that $D = (1+\beta)^2 - 4 \cdot k \cdot \beta(1+f)) < 0$ which implies that $k \cdot x^2 = (1+\beta)x - \beta(1+f)$ has imaginary roots. Hence by Lemma 3 this implies that the sequence $\{z_i\}$ has negative elements which is a contradiction to our assumption.

4.2 Construction of Sequence

Given the online algorithm with competitive ratio β we construct k dimensional matching using which we will construct the sequence $\{x_i\}$. For ease of description we will restrict to the case $k = 2$. The general k case is similar. In other words we construct bipartite graphs. Here each partite corresponds to a partition matroid. Hence the subset of edges is in the intersection of the two matroids if and only if it forms a valid matching. The graph will have two types of edges ex_i and ey_i. We will use x_i and y_i to denote the corresponding weights of the edges.

- Start with edge ex_1 with weight $x_1 = 1$.
- For simplicity we just state the inductive step in the construction. At step i the edge held by the algorithm is ex_i (and no other edge).
- At step $i+1$ we add edges to both ends of ex_i such that they differ in weight by at most ϵ and the algorithm accepts exactly 1 of them. Due to the matching condition it has to reject the currently accepted edge ex_i. Here adding a new edge to one end of ex_i means a edge is revealed. This new edge shares one vertex with ex_i and the other vertex is a brand new vertex which hasn't yet appeared in the algorithm. We describe how this is done below.
  At step $i+1$ add edges to both ends of ex_i of weight ϵ. If the algorithm does not accept either of the new edges rewind the algorithm and instead add edges of weight 2ϵ. Do this rewind, add edges of higher weights (higher by ϵ) till the algorithm accepts exactly 1 new edge [6]. Due to the matching constraint this means the current edge ex_i sharing an end point must be canceled and a penalty paid to it. Let the accepted edge be named ex_{i+1} and the other newly added edge be named ey_i. By construction we have $x_{i+1} \leq y_i + \epsilon$.
- Use the above construction to construct an infinite sequence x_i and y_i.

We will note some properties of the sequence x_i and y_i.

[6] Since the two edges are added sequentially we can stop when one of them is accepted.

– At step $i+1$ the algorithm accepts ex_{i+1} and cancels ex_i while not accepting ey_i. Consider the rewinding procedure in which the algorithm is presented edge ex'_{i+1} of weight $x_{i+1} - \epsilon$ and ey_i before ex_{i+1} is presented. By the construction of our rewinding procedure both ex'_{i+1} and ey_i would not be accepted and ex_i would still be the currently maintained edge in the solution. In such a case we assert some statements based on which we derive an inequality.
  - The utility of the algorithm is $x_i - f\sum_{j=1}^{i-1} x_{i-1}$ as ex_i is the currently maintained edge and $\{ex_1, ..., ex_{i-1}\}$ are currently canceled edges.
  - It is clear that $\{ey_1, ey_2, \ldots, ey_i, ex'_{i+1}\}$ is a valid matching in the current set of revealed edges. This has total weight $x_{i+1} - \epsilon + \sum_{j=1}^{i} y_i$
  - By definition of β we have $\beta(x_i - f\sum_{j=1}^{i-1} x_{i-1}) \geq x_{i+1} - \epsilon + \sum_{j=1}^{i} y_i$.
  - By construction of the sequence we also know that $y_i + \epsilon \geq x_{i+1}$.

  Substituting we get $\beta(x_i - f\sum_{j=1}^{i-1} x_{i-1}) \geq x_{i+1} - \epsilon + \sum_{j=2}^{i+1} x_j - (i+1)\epsilon$. Tending ϵ to 0 [7] we get $\beta(x_i - f\sum_{j=1}^{i-1} x_{i-1}) \geq x_{i+1} + \sum_{j=2}^{i+1} x_j$
– Note that the sequence constructed in not the one we desired. The sum on the right hand side starts from $j = 2$. $\beta(x_i - f\sum_{j=1}^{i-1} x_{i-1}) \geq x_{i+1} + \sum_{j=2}^{i+1} x_j$ implies $\beta(x_i - f\sum_{j=2}^{i-1} x_{i-1}) \geq x_{i+1} + \sum_{j=2}^{i+1} x_j$. So from the sequence $\{x_1, x_2, ...\}$ deleting x_1 and rescaling x_2 to 1 we get the desired sequence.

The general case k is very similar to case $k = 2$ but instead involves construction of k-dimensional matching.

Acknowledgments. The author thanks Bobby Kleinberg, Renato Paes Leme and Hu Fu for helpful suggestions.

References

1. Adler, R., Azar, Y.: Beating the logarithmic lower bound: randomized preemptive disjoint paths and call control algorithms. In: SODA, pp. 1–10 (1999)
2. Ashwinkumar, B.V., Kleinberg, R.: Randomized online algorithms for the buyback problem. In: Leonardi, S. (ed.) WINE 2009. LNCS, vol. 5929, pp. 529–536. Springer, Heidelberg (2009)
3. Azar, Y., Blum, A., Mansour, Y.: Combining online algorithms for rejection and acceptance. In: SPAA, pp. 159–163 (2003)
4. Babaioff, M., Hartline, J.D., Kleinberg, R.: Selling ad campaigns: online algorithms with buyback. In: EC (2009)
5. Babaioff, M., Immorlica, N., Kempe, D., Kleinberg, R.: A knapsack secretary problem with applications. In: Charikar, M., Jansen, K., Reingold, O., Rolim, J.D.P. (eds.) RANDOM 2007 and APPROX 2007. LNCS, vol. 4627, pp. 16–28. Springer, Heidelberg (2007)
6. Babaioff, M., Immorlica, N., Kleinberg, R.: Matroids, secretary problems, and online mechanisms. In: SODA, pp. 434–443 (2007)

[7] We are ignoring some issues of convergence for ease of exposition.

7. Canetti, R., Irani, S.: Bounding the power of preemption in randomized scheduling. In: STOC, pp. 606–615 (1995)
8. Chawla, S., Hartline, J.D., Malec, D.L., Sivan, B.: Multi-parameter mechanism design and sequential posted pricing. In: STOC (2010)
9. Constantin, F., Feldman, J., Muthukrishnan, S., Pál, M.: An online mechanism for ad slot reservations with cancellations. In: SODA (2009)
10. Cormode, G., Muthukrishnan, S.: Space efficient mining of multigraph streams. In: PODS, pp. 271–282 (2005)
11. Sarma, A.D., Gollapudi, S., Panigrahy, R.: Estimating pagerank on graph streams. In: PODS, pp. 69–78 (2008)
12. Demetrescu, C., Finocchi, I., Ribichini, A.: Trading off space for passes in graph streaming problems. In: SODA, pp. 714–723 (2006)
13. Dynkin, E.B.: Optimal choice of the stopping moment of a Markov process. Dokl. Akad. Nauk SSSR 150, 238–240 (1963)
14. Epstein, L., Levin, A., Mestre, J., Segev, D.: Improved approximation guarantees for weighted matching in the semi-streaming model. In: STACS (2010)
15. Feige, U., Mirrokni, V.S., Vondrak, J.: Maximizing non-monotone submodular functions. In: FOCS, pp. 461–471 (2007)
16. Feigenbaum, J., Kannan, S., McGregor, A., Suri, S., Zhang, J.: Graph distances in the streaming model: the value of space. In: SODA, pp. 745–754 (2005)
17. Feigenbaum, J., Kannan, S., McGregor, A., Suri, S., Zhang, J.: On graph problems in a semi-streaming model. Theor. Comput. Sci. 348, 207–216 (2005)
18. Feldman, J., Korula, N., Mirrokni, V., Muthukrishnan, S., Pál, M.: Online ad assignment with free disposal. In: Leonardi, S. (ed.) WINE 2009. LNCS, vol. 5929, pp. 374–385. Springer, Heidelberg (2009)
19. Garay, J.A., Gopal, I.S.: Call preemption in communication networks. In: IEEE INFOCOM, pp. 1043–1050 (1992)
20. Garay, J.A., Gopal, I.S., Kutten, S., Mansour, Y., Yung, M.: Efficient on-line call control algorithms. J. Algorithms 23(1), 180–194 (1997)
21. Hajiaghayi, M.T., Kleinberg, R., Sandholm, T.: Automated online mechanism design and prophet inequalities. In: AAAI (2007)
22. Halava, V., Harju, T., Hirvensalo, M.: Positivity of second order linear recurrent sequences. Discrete Appl. Math. 154(3), 447–451 (2006)
23. Jenkyns, T.A.: The efficacy of the greedy algorithm. In: Proc. of 7th South Eastern Conference on Combinatorics, Graph Theory and Computing, pp. 341–350 (1976)
24. Kleinberg, R.: A multiple-choice secretary algorithm with applications to online auctions. In: SODA, pp. 630–631 (2005)
25. Korte, B., Hausmann, D.: An analysis of the greedy heuristic for independence systems. Annals of Discrete Math. 2, 65–74 (1978)
26. Korula, N., Pál, M.: Algorithms for secretary problems on graphs and hypergraphs. In: Albers, S., Marchetti-Spaccamela, A., Matias, Y., Nikoletseas, S., Thomas, W. (eds.) ICALP 2009. LNCS, vol. 5556, pp. 508–520. Springer, Heidelberg (2009)
27. Lee, J., Mirrokni, V.S., Nagarajan, V., Sviridenko, M.: Non-monotone submodular maximization under matroid and knapsack constraints. In: STOC (2009)
28. McGregor, A.: Finding graph matchings in data streams. In: Chekuri, C., Jansen, K., Rolim, J.D.P., Trevisan, L. (eds.) APPROX 2005 and RANDOM 2005. LNCS, vol. 3624, pp. 170–181. Springer, Heidelberg (2005)
29. Nemhauser, G.L., Wolsey, L.A., Fisher, M.L.: An analysis of approximations for maximizing submodular set functions ii. In: Mathematical Programming Study, pp. 73–87 (1978)

Tamper-Proof Circuits: How to Trade Leakage for Tamper-Resilience

Sebastian Faust[1,⋆], Krzysztof Pietrzak[2,⋆⋆], and Daniele Venturi[3]

[1] K.U. Leuven ESAT-COSIC/IBBT
[2] CWI Amsterdam
[3] Sapienza University of Rome

Abstract. Tampering attacks are cryptanalytic attacks on the implementation of cryptographic algorithms (e.g., smart cards), where an adversary introduces faults with the hope that the tampered device will reveal secret information. Inspired by the work of Ishai et al. [Eurocrypt'06], we propose a compiler that transforms any circuit into a new circuit with the same functionality, but which is resilient against a well-defined and powerful tampering adversary. More concretely, our transformed circuits remain secure even if the adversary can *adaptively* tamper with *every* wire in the circuit as long as the tampering fails with some probability $\delta > 0$. This additional requirement is motivated by practical tampering attacks, where it is often difficult to guarantee the success of a specific attack.

Formally, we show that a q-query tampering attack against the transformed circuit can be "simulated" with only black-box access to the original circuit and $\log(q)$ bits of additional auxiliary information. Thus, if the implemented cryptographic scheme is secure against $\log(q)$ bits of leakage, then our implementation is tamper-proof in the above sense. Surprisingly, allowing for this small amount of information leakage allows for much more efficient compilers, which moreover do not require randomness during evaluation. Similar to earlier works our compiler requires *small, stateless and computation-independent* tamper-proof gadgets. Thus, our result can be interpreted as reducing the problem of shielding arbitrary complex computation to protecting *simple* components.

1 Introduction

Modern security definitions usually consider some kind of game between an adversary and the cryptosystem under attack, where the adversary interacts with

⋆ Supported in part by Microsoft Research through its PhD Scholarship Programme, by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy), and FWO grant G.0225.07. Part of this work was done while the first and the third author were visiting CWI, Amsterdam.

⋆⋆ Supported by the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) / ERC Starting Grant (259668-PSPC).

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 391–402, 2011.

the system and finally must break it. A distinctive feature of such notions is that the adversary has only *black-box* access to the cryptosystem. Unfortunately, in the last decade it became evident that such black-box notions do not capture adversaries that attack the physical implementation of a cryptosystem. Recently, significant progress has been made to close this gap. With few exceptions most of this research is concerned with "side-channel attacks". These are attacks where the adversary measures information that is leaked from the cryptodevice during computation.

In this work we explore *active* physical attacks which so far got much less attention from the theory community (a few examples can be found in [10, 12, 7]). We study the security of cryptographic implementations when the adversary can not only measure, but actively tamper with the computation of the physical device, e.g. by introducing faults. Such attacks, often called *fault analysis* or *tampering attacks*, are a serious threat to the security of real-world implementations and often allow to completely break otherwise provably secure schemes. In this work we investigate the general question whether *any* cryptographic scheme can be implemented *efficiently* such that it resists a very powerful adversary tampering with the whole computation and the memory.

Many techniques to induce faults into the computation of a cryptodevice have been proposed. Important examples include heating up the device, expose it to infrared radiation or alter the internal power supply or clock [4, 6, 18]. One might think that an adversary that obtains the result of faulty computation will not be very useful. In a seminal paper Boneh et al. [6] show that already a single fault may allow to completely break an RSA based digital signature scheme. Different methods to counter such attacks have been proposed. Most such countermeasures have in common that they protect against specific adversarial strategies and come without a rigorous security analysis. This is different from the *provable security* approach followed by modern cryptography, where one first defines a precise adversarial model and then proves that no (resource-bounded) attacker exists who breaks the scheme.

An influential work on provable security against tampering attacks is the work on *private circuits* of Ishai et al. [13, 12]. Informally, such "private circuits" carry out a specific cryptographic task (e.g., signing), while protecting the internal secret state against a well-defined class of tampering attacks. The authors construct a general circuit compiler that transforms *any* Boolean circuit C into a functionally equivalent "private circuit" $\widehat{C}$. It is shown that an adversary who can tamper[1] with at most t wires in-between any two invocations of $\widehat{C}$, cannot learn anything more about the secret state than an adversary having just black-box access to C. Security is proven by a simulation argument: for any adversary $\mathcal{A}$ that can mount a tampering attack on the circuit, there exists an (efficient) simulator $\mathcal{S}$ having only black-box access to the circuit, such that the output distribution of $\mathcal{A}$ and $\mathcal{S}$ are statistically close.

[1] Tampering with a wire means permanently *set* its value to the constant 1 or 0 or *toggle* the wire, which means that whatever value is put on the wire gets inverted.

To achieve this goal their transformation uses techniques from multi-party computation and combines randomized computation with redundant encodings to detect faulty computation. If tampering is detected a self-destruction mechanism is triggered that overwrites the complete state, so that, from there on, regular and tampering queries to the circuit can be trivially simulated. One difficulty that needs to be addressed is that this self-destruction mechanism itself is exposed to tampering attacks. In particular, an adversary could just try to cancel any trigger for self-destruction and from then on apply arbitrary attacks without being in danger of detection. Ishai et al. face this problem by spreading and propagating errors that appear during the computation. As discussed later we will use similar techniques in our work.

Our Contribution. The "holy grail" in this line of research is to find an efficient general circuit compiler that provably resists *arbitrary* tampering attacks to an *unlimited number* of wires. This goal might be too ambitious since an adversary might just "reprogram" the circuit such that it outputs its complete secret state. Hence, to have any hope to formally analyze the security against tampering attacks we will need to limit the power of the adversary. As just discussed, Ishai et al. limit the adversary to tamper with at most t wires in each round. Their construction blows up the circuit size quite significantly and makes extensive use of randomness: for a statistical security parameter k, the size of the transformed circuit is by a factor of $O(k^3t)$ larger and requires $O(k^2)$ bits of randomness in each invocation.

"Noisy" tampering. In this work we consider a somewhat different (and incomparable) adversarial model, where the adversary can tamper with *every* wire (not just some small number t) in the circuit, but tampering with every wire will fail (independently) with some probability $\delta \geq 0$.[2] This model is partially motivated by existing attacks [17, 5, 6]. Concrete examples of attacks covered by our model are, e.g., optical attacks and Eddy currents attacks (cf. [17] for details).

Leakage. Another crucial difference between our work and [12] is that we use a relaxed security definition which allows the tampering adversary to learn a logarithmic amount of information about the secret state (in total, not per invocation). This relaxation buys us a lot in terms of efficiency and simplicity. In particular, for security parameter k we achieve statistical security by blowing up the circuit size by only $O(k)$ and requiring *no* randomness during run-time (and only $2k$ bits during production).

If q is the number of queries and n the size of the output of the original circuit, the amount of leakage that an adversary can learn by tampering with the circuit is $\log(q \cdot n)$ bits. Intuitively, the only advantage an adversary gets by being able to tamper with the transformed circuit is to "destroy" its internal state, but

[2] The adversary can tamper the same wire several times, but only once in-between every two invocations. As tampering is persistent, after a sufficiently large number of attempts the tampering will succeed almost certainly, i.e. with probability $1 - \delta^l$ after l rounds.

the point in the computation where the destruction happens can depend on the secret state. Hence, this "point of failure" may be leaked to the adversary.

If we apply our transformation to a particular cryptosystem in order to get a tamper-resilient version $\widehat{C}$ of it, it is crucial that the scheme C remains secure even given the leakage. Some primitives like public-key encryption [2] or digital signatures [10, 8] are always secure against a logarithmic amount of arbitrary leakage, but a logarithmic amount of leakage can decrease the security of the PKE or signature scheme by a polynomial factor. Recently signature schemes [15, 3] and public-key encryption schemes [2, 16] have been constructed where the security does not degrade at all, even if the amount of leakage is quite substantial. Using such schemes we can avoid the loss in security due to the leakage.

Overview of the construction. Starting with any Boolean (probabilistic) circuit C our transformation Φ outputs a transformed circuit $\widehat{C}$ that consists of k subcircuits (which we will call the *core*). Each subcircuit is made out of special constant size *tamper-proof* masked "Manchester gadgets". Instead of doing simple Boolean operations, these gadgets compute with encodings, so called *masked Manchester encodings* (which encode a single bit into 4 bits). If the inputs are not valid encodings, such gadgets output the invalid state, i.e., 0000. Since each of the k subcircuits is made solely out of such special gadgets, errors introduced by a tampering attack will propagate to the output of the core and are input to a self-destruction phase (essentially, identical to the one introduced by [12]). In contrast to the core of the circuit which is built with gadgets of constant size, the self-destruction phase (in the following also called cascade phase) will require (simple, stateless and deterministic) tamper-proof gadgets of linear (in k) size. Ishai et al. require tamper-proof gadgets of linear (in kt) size in the *entire* circuit, albeit simpler ones than we do.[3] Unlike [12], we do not require so called "reversible" NOT gates. Such gadgets propagate a fault on the output side to their input side and are highly non-standard.

It is not difficult to see that the transformation as outlined above is not secure in the setting of Ishai et al. (i.e. when $\delta = 0$ and the adversary can tamper with up to t wires in each round). Nevertheless, we show in the full version of this paper [1] how to adjust our compiler to achieve similar efficiency improvement when $\delta = 0$ and some small amount of leakage is allowed.

On tamper-proof gadgets. The assumption of the existence of simple components that withstand active physical attacks has been frequently made in the literature [12, 10] (and several others in the context of leakage [9, 14, 11]). Of course, the simpler the components are, the stronger the result is that one gets.

1. The components we use are fixed, standard and universal elements that can be added to once standard cell library. This is far better than designing over and over task specific tamper-proof components.
2. Our gadgets are simple, stateless and deterministic. In particular, the gadgets used in the core of the circuit have a small constant size.

[3] More precisely, they require tamper-proof AND gadgets that take $4kt$ bits as input.

3. Our transformation for the core is independent of the cascade phase, which uses linear size gadgets. Thus one can implement a universal "tamper-proof cascade phase", and only has to compile and implement the core.

Outline of the security proof. We construct an efficient simulator $\mathcal{S}$ that – having only black-box access to the circuit C – can simulate access to the transformed circuit $\widehat{C}$ including adversarial tampering queries. The main challenge here is consistency, that is, answers computed by $\mathcal{S}$ must have the same distribution as an adversary would see when tampering with $\widehat{C}$.

If an adversary tampers with $\widehat{C}$, the subsequent invocation of $\widehat{C}$ can have one of three different outcomes:

1. Nothing happens: the invocation goes through as if no tampering did happen (this is, e.g., the case if a wire is set to 0, but its value during the invocation is 0 anyway).
2. Self-destruct: the redundancy added to $\widehat{C}$ "detects" tampering, and the entire state is deleted.
3. Successful Tampering: the outcome of $\widehat{C}$ changes as a consequence of the tampering, and this tampering was not detected.

In a first step, we show that case 3 will not happen but with exponentially small probability. To show this, we use the fact that tampering with any particular wire fails with probability δ, and moreover that every bit carried by a wire in C is encoded in $\widehat{C}$ with a highly redundant and randomized encoding. This guarantees that the chance of an adversary to change a valid encoding of a bit to its complement is tiny: either she has to be lucky – in the sense that she tampers with many wires at once and all attacks succeed – or she has to guess the randomness used in the encoding.

As we ruled out case 3., we must only build a simulator $\mathcal{S}$ that simulates $\widehat{C}$ as if no tampering has happened (i.e., case 1.). This is easy as $\mathcal{S}$ has access to C which is functionally equivalent. Moreover, at some point $\mathcal{S}$ has to simulate a self-destruct (i.e., case 2.). Unfortunately, there is no way for $\mathcal{S}$ to know when the self-destruct happens (as the probability of this event can be correlated with the secret state). As explained before, we provide the exact point of failure as auxiliary input to $\mathcal{S}$.

The simulator has to continue the simulation even after the self-destruct. This seems easy, as now all the secret state has been deleted. There is one important technicality though. As tampering is *permanent*, even after self-destruct the simulator $\mathcal{S}$ must simulate a circuit in a way that is consistent with the simulation so far. A priori the simulator only knows which wires the adversary tried to tamper, but recall that each tampering is only successful with probability $1 - \delta$. For this reason, we let the simulator choose all the randomness used, including the randomness of the compiler (which generates $\widehat{C}$ from C) and the randomness that determines the success of the tampering attacks. Knowledge of this randomness, allows the simulator to continue simulation after self-destruct.

Note that the above-mentioned auxiliary information (i.e., the point at which self-destruct is triggered) can be computed as a function of this randomness, and the randomness used by the adversary.

2 Definitions

Notation. If D is a distribution over a set $\mathcal{D}$, then $x \leftarrow \mathsf{D}$ means a random variable x is drawn from D (if $\mathcal{D}$ is a set with no distribution specified, then $x \leftarrow \mathcal{D}$ denotes a random variable with uniform distribution over $\mathcal{D}$). If D is an algorithm, then $y \leftarrow \mathsf{D}(x)$ means that y is the output of D on input x; in particular when D is probabilistic, y is a random variable. Two distributions D and D' are ϵ-*close*, written $\mathsf{D} \approx_\epsilon \mathsf{D}'$, if their statistical distance $\frac{1}{2}\sum_{x\in\mathcal{D}} |\mathsf{D}(x) - \mathsf{D}'(x)|$ is less than or equal to ϵ. We write $\mathcal{A}^{\mathcal{O}(\cdot)}$ to denote an algorithm $\mathcal{A}$ with oracle access to $\mathcal{O}(\cdot)$. Given two codewords $x, y \in \{0,1\}^n$ their Hamming distance, $0 \le d_H(x,y) \le n$, is the number of positions in which x and y differ.

Our Model. Our physical model of computation is very similar to [12]. We consider (probabilistic) stateful Boolean circuits C and present circuit compilers Φ that transform any such circuit into a new circuit $\widehat{C}$ resistant against a well-defined class of tampering attacks. Details follow below.

Circuits. A Boolean circuit C is a directed acyclic graph whose vertices are standard Boolean gates and whose edges are the wires. The depth of C, denoted $depth(C)$, is the longest path from an input to an output. A circuit is *clocked* if it evolves in clock cycles (or rounds). The input and output values of the circuit C in clock cycle i are denoted by X_i and Y_i, respectively. A circuit is *probabilistic* if it uses internal randomness as part of its logic. We call such probabilistic logic *randomness gates* and denote them with \$. In each clock cycle \$ outputs a fresh random bit.

Additionally, a circuit may contain *memory gates*. Memory gates, which have a single incoming edge and any number of outgoing edges, maintain state: at any clock cycle, a memory gate sends its current state down its outgoing edges and updates it according to the value of its incoming edge. Any cycle in the circuit graph must contain at least one memory gate. The state of all memory gates at clock cycle i is denoted by M_i, with M_0 denoting the initial state. When a circuit is run in state M_{i-1} on input X_i, the circuit will output Y_i and the memory gates will be in a new state M_i. We will denote this by $(Y_i, M_i) \leftarrow C[M_{i-1}](X_i)$.

Adversarial model. We consider adversaries that can adaptively tamper in q clock cycles with up to t wires. In this paper we are particularly interested in the case where t is unbounded, i.e., the adversary can tamper with an *arbitrarily large* number of wires in the circuit in *every* round. For each wire we allow the adversary to choose between the following types of attacks: *set*, i.e., setting a wire to 1, *reset*, i.e., setting a wire to 0 and *toggle*, i.e., flipping the value on the wire. For each wire such an attack fails independently with some probability. This is captured by the global parameter δ, where $\delta = 0$ means that the attack succeeds always, and $\delta = 1$ that no tampering takes place. The model of [12] considers the case in which t is a small integer and tampering is always successful, i.e., $\delta = 0$.

When an attack fails for one wire the computation continues with the original value on that wire. Notice that once a fault is successfully placed it stays

permanently. Let us stress that we do allow the adversary to "undo" (with zero error probability) persistent attacks induced in previous rounds (this captures so called transient faults). We call such an adversary, that can adaptively tamper with a circuit for up to q clock cycles attacking up to t wires per round, an (t, δ, q)-adversary and denote the attack strategy for each clock cycle as $\mathsf{W} = \{(w_1, a_1), \ldots, (w_t, a_t)\}$. The first element in each such tuple specifies which wire in the circuit is attacked and the second element specifies the type of attack (i.e., *set*, *reset* or *toggle*). When the number of faults per clock cycle is unbounded, we will explicitely write $t = \infty$.

Tamper-Proof Security. The definitions below are given for (∞, δ, q)-adversaries, but can be adapted to the case where the number t of faults in every clock cycle is bounded in a straight forward way.

Transformation. A circuit transformation Φ takes as input a security parameter k, a (probabilistic) circuit C and an initial state M_0 and produces a transformed initial state $\widehat{M}_0$ and a transformed (probabilistic) circuit $\widehat{C}$. This is denoted by $(\widehat{C}, \widehat{M}_0) \leftarrow \Phi(C, M_0)$. The compiled circuit can use a different set of gates, and this will be the case for the compiler we construct. The transformation itself can be randomized and we let ρ_Φ denote the random coins of the transformation. We say that the transformation Φ is *functionality preserving* if for all C, M_0 and any set of public inputs $X_1, X_2, \ldots, X_q$ the original circuit C starting with state M_0 and the transformed circuit $\widehat{C}$ starting with state $\widehat{M}_0$ result in an identical output distribution.

Following [12], we define security of circuit transformations against tampering attacks by a simulation-based argument, but we augment the simulation by allowing auxilliary input. Loosely speaking, for every (∞, δ, q)-adversary $\mathcal{A}$ tampering with $\widehat{C}$, there exists a simulator $\mathcal{S}_\lambda$, that gets as input some λ-bounded auxiliary information Λ and only has black-box access to the original circuit C, such that the output distribution of $\mathcal{A}$ and $\mathcal{S}_\lambda$ are close. We will specify the nature of the auxiliary information below.

Real world experiment. The adversary $\mathcal{A}$ can in each round i adaptively specify an input X_i and an attack strategy W_i that is applied to the transformed circuit $\widehat{C}$ when run on input X_i with secret state $\widehat{M}_{i-1}$. The output Y_i resulting from the (possibly) faulty computation is given to the adversary and the state is updated to $\widehat{M}_i$ for the next evaluation. To formally describe such a process we introduce a special oracle, Tamper, that can be queried on (X_i, W_i) to return the result Y_i. More precisely, for any (∞, δ, q)-adversary $\mathcal{A}$, any circuit C and any initial state M_0, we define the following experiment:

Experiment. $\mathsf{Exp}_\Phi^{\mathrm{Real}}(\mathcal{A}, C, M_0)$:
$(\widehat{C}, \widehat{M}_0) \leftarrow \Phi(C, M_0)$
Output $\mathcal{A}^{\mathsf{Tamper}(\widehat{C}, \widehat{M}_0, \cdot, \cdot)}(C)$

Simulation. The simulator $\mathcal{S}_\lambda$ simulates the adversary's view, however, she has to do so without having tampering access to the transformed circuit. More precisely,

the simulator only has oracle access to $C[M_0](\cdot)$. Additionally, we will give the simulator some λ-bounded auxiliary information. This is described by letting $\mathcal{S}_\lambda$ choose an arbitrary function $f : \{0,1\}^* \to \{0,1\}^\lambda$ and returning the result of f evaluated on input the secret state M_0, i.e., $\Lambda \stackrel{\mathsf{def}}{=} f(M_0)$. For a simulator $\mathcal{S}_\lambda$ we define the following experiment for any circuit C, any initial state M_0 and any (∞, δ, q)-adversary $\mathcal{A}$:

Experiment. $\mathsf{Exp}^{\mathrm{Sim}}_{\Phi}(\mathcal{S}_\lambda, C, M_0, \mathcal{A})$:
$f \leftarrow \mathcal{S}_\lambda(\mathcal{A}, C)$ where $f : \{0,1\}^* \to \{0,1\}^\lambda$
Output $\mathcal{S}_\lambda^{C[M_i](\cdot)}(\Lambda)$ where $\Lambda = f(M_0)$

Tamper-resilient circuit transformations. A circuit transformation is said to be tamper-resilient if the outputs of the two experiments are statistically close.

Definition 1. (**Tamper-Resilience of Circuit Transformation**). *A circuit transformation Φ is $(\infty, \delta, q, \lambda, \epsilon)$*-tamper-resilient *if for any (∞, δ, q)-adversary $\mathcal{A}$, for any circuit C and any initial state M_0, there exists a simulator $\mathcal{S}_\lambda$ such that*

$$\mathsf{Exp}^{\mathrm{Sim}}_{\Phi}(\mathcal{S}_\lambda, C, M_0, \mathcal{A}) \approx_\epsilon \mathsf{Exp}^{\mathrm{Real}}_{\Phi}(\mathcal{A}, C, M_0),$$

where the probabilities are taken over all the random coin tosses involved in the experiments.[4]

3 A Compiler Secure against (∞, δ, q)-Adversaries

We describe a compiler Φ which is secure against (∞, δ, q)-adversaries. A different construction for the case of a small t and $\delta = 0$ – i.e. when the number of faults per round is *bounded* but attacks succeed always – is given in the full version [1].

Instead of computing with bits the compiled circuit $\widehat{C}$ will operate on redundant and randomized encodings of bits.

Encodings. Our transformation is based on three encoding schemes, where each is used to encode the previous one. The first encoding, so called *Manchester encoding*, can be described by a deterministic function that takes as input a bit $b \in \{0,1\}$ and has the following output: $\mathsf{MC}(b) \stackrel{\mathsf{def}}{=} (b, \bar{b})$. Decoding is done just by outputting the first bit. The output $(b, \bar{b})$ is given as input to the next level of our encoding procedure where we use a probabilistic function $mask : \{0,1\}^2 \times \{0,1\}^2 \to \{0,1\}^4$. Such a function uses as input additionally two random bits for *masking* its output. More precisely, we have $mask(\mathsf{MC}(b), (r, r')) \stackrel{\mathsf{def}}{=} (b \oplus r, r, \bar{b} \oplus r', r')$, with $(r, r') \leftarrow \{0,1\}^2$. We denote with $\mathsf{MMC} \subset \{0,1\}^4$ the set of valid masked Manchester encoded bits, and with $\overline{\mathsf{MMC}} \stackrel{\mathsf{def}}{=} \{0,1\}^4 \setminus \mathsf{MMC}$ the non-valid encodings. Our final encoding consists of k independent masked Manchester encodings:

$$\mathsf{Enc}(b, \vec{r}) \stackrel{\mathsf{def}}{=} mask(\mathsf{MC}(b), (r_1, r'_1)), \ldots, mask(\mathsf{MC}(b), (r_k, r'_k)), \tag{1}$$

[4] The parameters δ, q, λ and ϵ are all parameterized by the security parameter k.

with $\vec{r} = (r_1, r'_1, r_2, r'_2, \ldots, r_k, r'_k) \in \{0,1\}^{2k}$. Thus it has length $4k$ bits and uses $2k$ bits of randomness. When the randomness in an encoding is omitted, it is uniformly sampled, e.g. $\mathsf{Enc}(b)$ denotes the random variable $\mathsf{Enc}(b, \vec{r})$ where $\vec{r} \in \{0,1\}^{2k}$ is sampled uniformly at random.

We denote with $\mathsf{Enc} \subset \{0,1\}^{4k}$ the set of all valid encodings and with $\overline{\mathsf{Enc}} \stackrel{\mathsf{def}}{=} \{0,1\}^{4k} \setminus \mathsf{Enc}$ the non-valid ones.

The Compiler. Consider any (probabilistic) Boolean circuit C that consists of Boolean NAND gates, randomness gates \$ and memory cells. We assume that the original circuit handles fanout through special copy gates taking one bit as input and outputting two copies. If k copies are needed, the original value is passed through a subcircuit of $k-1$ copy gadgets arranged in a tree structure. Let us first describe the transformation for the secret state. On factory setup $2k$ random bits $\rho_\Phi = (r_1, r'_1, \ldots, r_k, r'_k)$ are sampled uniformly. Then, each bit of the secret state m_i is encoded by $\mathsf{Enc}(m_i, \rho_\Phi)$. Putting all these encodings together we get the initial transformed secret state $\widehat{M}_0$. The encoded secret state will be stored in the memory cells of $\widehat{C}$, but we will discuss this below. Notice that we use the same randomness for each encoding.

The global picture of our transformation consists of four different stages: the encoder, the input/output cascade phase, the transformation for the core and the decoder. These stages are connected as shown in Figure 1 and are described below.

The encoder and the decoder. Since the compiled circuit computes with values in encoded form, we need to specify how to encode and decode the public inputs and outputs of $\widehat{C}$. The encoder (which is deterministic and build from copy and negation gates) encodes every bit of the input using randomness ρ_Φ:

$$\mathsf{Encoder}(x_1, \ldots, x_t) \stackrel{\mathsf{def}}{=} \mathsf{Enc}(x_1, \rho_\Phi), \ldots, \mathsf{Enc}(x_t, \rho_\Phi) \quad \text{where} \quad x_1, \ldots, x_t \in \{0,1\}.$$

The decoding phase simply outputs the XORs of the first two bits of every encoding:

$$\mathsf{Decoder}(X_1, \ldots, X_{t'}) \stackrel{\mathsf{def}}{=} X_1[1] \oplus X_1[2], \ldots, X_{t'}[1] \oplus X_{t'}[2] \quad \text{where} \quad X_i \in \{0,1\}^{4k}.$$

The input and output cascade phases. For self-destruction we use a tool already introduced by Ishai et al. – the *cascade phase* (cf. Figure 2). In our construction we make use of two cascade phases: an *input* cascade phase and an *output* cascade phase. As shown in Figure 1 the input cascade phase takes as input the output of the encoder and the encoded secret state. The output cascade phase takes as inputs the output of the core and the updated secret state.[5] For technical reasons we require that the secret state is always in the top part and the public output is always on the bottom part of the cascade phase. For ease of description

[5] Notice that the input and the output cascade phases might have a different number of inputs/outputs.

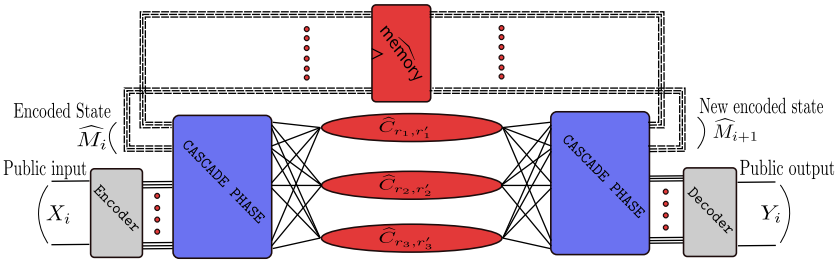


Fig. 1. A global picture of our compiler in the case $k = 3$. In the red-coloured parts we rely on gadgets of constant size, whereas in the blue-coloured parts gadgets of linear size (in the security parameter k) are used.

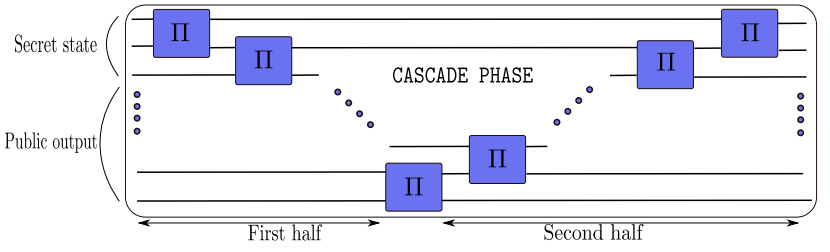


Fig. 2. The cascade phase for error propagation and self-destruction

we call the part of the cascade phase that takes the inputs as the first half and the part that produces the outputs as the second half (cf. Figure 2).

Inside the cascade phase we make use of special *cascade gadgets* $\Pi : \{0,1\}^{8k} \to \{0,1\}^{8k}$. The gadgets behave like the identity function if the inputs are valid encodings using randomness ρ_Φ, and output 0^{8k} otherwise, i.e.

$$\Pi(A,B) = \begin{cases} A, B \text{ if } A, B \in \{\mathsf{Enc}(0,\rho_\Phi), \mathsf{Enc}(1,\rho_\Phi)\} \\ 0^{8k} \quad \text{otherwise.} \end{cases}$$

The gadgets are assumed to be *tamper-proof*, i.e. the adversary is allowed to tamper with their inputs and outputs, but she cannot modify their internals.

The core. As outlined in the introduction, the core of the circuit is made out of k sub-circuits each using special tamper-proof gadgets of constant size. Let us describe these gadgets in more detail. With $\widehat{\mathsf{NAND}}_{r,r'} : \{0,1\}^{2\times 4} \to \{0,1\}^4$ we define a NAND gate which works on masked Manchester encodings using randomness r, r' (on input and output). If the input contains anything else than a valid masked Manchester encoding, the output is $0^4 \in \overline{\mathsf{MMC}}$. The truth table of these gadgets is given in Figure 3. Similarly we denote with $\widehat{\mathsf{copy}}_{r,r'} : \{0,1\}^4 \to \{0,1\}^{2\times 4}$ a copy gate which takes as input a masked Manchester encoding using randomness r, r' and outputs two copies of it. Whenever the input contains anything else than a masked Manchester encoding using randomness r, r', the output is $0^8 \in \overline{\mathsf{MMC}}$.

1st Input	2nd Input	Output
$mask(\mathsf{MC}(0,r,r'))$	$mask(\mathsf{MC}(0,r,r'))$	$mask(\mathsf{MC}(1,r,r'))$
$mask(\mathsf{MC}(0,r,r'))$	$mask(\mathsf{MC}(1,r,r'))$	$mask(\mathsf{MC}(1,r,r'))$
$mask(\mathsf{MC}(1,r,r'))$	$mask(\mathsf{MC}(0,r,r'))$	$mask(\mathsf{MC}(1,r,r'))$
$mask(\mathsf{MC}(1,r,r'))$	$mask(\mathsf{MC}(1,r,r'))$	$mask(\mathsf{MC}(0,r,r'))$
$\star$	$\star$	0^4

Fig. 3. Truth table of $\widehat{\mathsf{NAND}}_{r,r'} : \{0,1\}^{2\times 4} \to \{0,1\}^4$

$$\widehat{\mathsf{copy}}_{r,r'}(A) = \begin{cases} A, A \text{ if } A \in \{mask(\mathsf{MC}(0,r,r')), mask(\mathsf{MC}(1,r,r'))\} \\ 0^8 \quad \text{otherwise.} \end{cases}$$

Finally we let $\widehat{\$}_{r,r'}$ denote a randomness gadget outputting a fresh masked Manchester encoded random bit.

With $\widehat{C}_{r,r'}$ we denote the circuit we get by replacing every wire in C with 4 wires (carrying an encoding in MMC using randomness r, r') and every NAND gate (resp. copy gate, \$ gate) in C with a $\widehat{\mathsf{NAND}}_{r,r'}$ (resp. $\widehat{\mathsf{copy}}_{r,r'}$, $\widehat{\$}_{r,r'}$). Similar to the Π gadgets, we require the $\widehat{\mathsf{NAND}}_{r,r'}$, $\widehat{\mathsf{copy}}_{r,r'}$, $\widehat{\$}_{r,r'}$ gadgets to be *tamper-proof*, i.e. the adversary is allowed to tamper with their inputs and outputs, but cannot modify the internals. Note that if we want to avoid the use of $\widehat{\$}_{r,r'}$ gadgets we can derandomize the original circuit C replacing the \$ gates with the output of a PRG. The core of the transformed circuit consists of the k circuits $\widehat{C}_{r_1,r'_1}, \ldots, \widehat{C}_{r_k,r'_k}$ (where the r_i, r'_i are from ρ_Φ).

Security. We can now state our main theorem.

Theorem 1 (Tamper-resilience against (∞,δ,q)-adversaries). *Let $0 < \delta < 1/2$, $k > 0$. The compiler Φ of Section 3 is $(\infty,\delta,q,\lambda,\epsilon)$-tamper resilient, where $\lambda = \log(q) + \log(n+1) + 1$ and $\epsilon = 3(1-\delta/2)^k$.*

Proof. Due to lack of space the proof has been moved to the full version [1].

Acknowledgments. We thank Yuval Ishai and Manoj Prabhakaran for helpful discussions on their work in [12].

References

[1] The full version of this paper will be posted on the Cryptology ePrint Archive, http://eprint.iacr.org/

[2] Akavia, A., Goldwasser, S., Vaikuntanathan, V.: Simultaneous hardcore bits and cryptography against memory attacks. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 474–495. Springer, Heidelberg (2009)

[3] Alwen, J., Dodis, Y., Wichs, D.: Leakage-resilient public-key cryptography in the bounded-retrieval model. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 36–54. Springer, Heidelberg (2009)

[4] Anderson, R., Kuhn, M.: Tamper resistance: a cautionary note. In: WOEC 1996, p. 1. USENIX Association, Berkeley (1996)
[5] Blömer, J., Seifert, J.-P.: Fault based cryptanalysis of the advanced encryption standard (AES). In: Wright, R.N. (ed.) FC 2003. LNCS, vol. 2742, pp. 162–181. Springer, Heidelberg (2003)
[6] Boneh, D., DeMillo, R.A., Lipton, R.J.: On the importance of eliminating errors in cryptographic computations. J. Cryptology 14(2), 101–119 (2001)
[7] Dziembowski, S., Pietrzak, K., Wichs, D.: Non-malleable codes. In: ICS 2010, pp. 434–452 (2010)
[8] Faust, S., Kiltz, E., Pietrzak, K., Rothblum, G.N.: Leakage-resilient signatures. In: Micciancio, D. (ed.) TCC 2010. LNCS, vol. 5978, pp. 343–360. Springer, Heidelberg (2010)
[9] Faust, S., Rabin, T., Reyzin, L., Tromer, E., Vaikuntanathan, V.: Protecting circuits from leakage: the computationally-bounded and noisy cases. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 135–156. Springer, Heidelberg (2010)
[10] Gennaro, R., Lysyanskaya, A., Malkin, T., Micali, S., Rabin, T.: Algorithmic tamper-proof (ATP) security: Theoretical foundations for security against hardware tampering. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 258–277. Springer, Heidelberg (2004)
[11] Goldwasser, S., Rothblum, G.N.: Securing computation against continuous leakage. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 59–79. Springer, Heidelberg (2010)
[12] Ishai, Y., Prabhakaran, M., Sahai, A., Wagner, D.: Private circuits II: Keeping secrets in tamperable circuits. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 308–327. Springer, Heidelberg (2006)
[13] Ishai, Y., Sahai, A., Wagner, D.: Private circuits: Securing hardware against probing attacks. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 463–481. Springer, Heidelberg (2003)
[14] Juma, A., Vahlis, Y.: Protecting cryptographic keys against continual leakage. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 41–58. Springer, Heidelberg (2010)
[15] Katz, J., Vaikuntanathan, V.: Signature schemes with bounded leakage resilience. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 703–720. Springer, Heidelberg (2009)
[16] Naor, M., Segev, G.: Public-key cryptosystems resilient to key leakage. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 18–35. Springer, Heidelberg (2009)
[17] Otto, M.: Fault Attacks and Countermeasures. PhD thesis, University of Paderborn, Germany (2006)
[18] Skorobogatov, S.P., Anderson, R.J.: Optical fault induction attacks. In: Kaliski Jr., B.S., Koç, Ç.K., Paar, C. (eds.) CHES 2002. LNCS, vol. 2523, pp. 2–12. Springer, Heidelberg (2003)

New Algorithms for Learning in Presence of Errors*

Sanjeev Arora and Rong Ge

Department of Computer Science, Princeton University
and Center for Computational Intractability
{arora,rongge}@cs.princeton.edu

Abstract. We give new algorithms for a variety of randomly-generated instances of computational problems using a *linearization* technique that reduces to solving a system of linear equations.

These algorithms are derived in the context of learning with *structured* noise, a notion introduced in this paper. This notion is best illustrated with the *learning parities with noise* (LPN) problem —well-studied in learning theory and cryptography. In the standard version, we have access to an oracle that, each time we press a button, returns a random vector $\mathbf{a} \in \mathrm{GF}(2)^n$ together with a bit $b \in \mathrm{GF}(2)$ that was computed as $\mathbf{a}\cdot\mathbf{u}+\eta$, where $\mathbf{u} \in \mathrm{GF}(2)^n$ is a *secret* vector, and $\eta \in \mathrm{GF}(2)$ is a *noise* bit that is 1 with some probability p. Say $p = 1/3$. The goal is to recover $\mathbf{u}$. This task is conjectured to be intractable.

In the structured noise setting we introduce a slight (?) variation of the model: upon pressing a button, we receive (say) 10 random vectors $\mathbf{a_1}, \mathbf{a_2}, \ldots, \mathbf{a_{10}} \in \mathrm{GF}(2)^n$, and corresponding bits $b_1, b_2, \ldots, b_{10}$, of which at most 3 are noisy. The oracle may arbitrarily decide which of the 10 bits to make noisy. We exhibit a polynomial-time algorithm to recover the secret vector $\mathbf{u}$ given such an oracle. We think this structured noise model may be of independent interest in machine learning.

We discuss generalizations of our result, including learning with more general noise patterns. We also give the first nontrivial algorithms for two problems, which we show fit in our structured noise framework.

We give a slightly subexponential algorithm for the well-known *learning with errors* (LWE) problem over $\mathrm{GF}(q)$ introduced by Regev for cryptographic uses. Our algorithm works for the case when the gaussian noise is small; which was an open problem. Our result also clarifies why existing hardness results fail at this particular noise rate.

We also give polynomial-time algorithms for learning the MAJORITY OF PARITIES function of Applebaum et al. for certain parameter values. This function is a special case of Goldreich's pseudorandom generator.

The full version is available at http://www.eccc.uni-trier.de/report/2010/066/.

* Research supported by NSF Grants CCF-0832797, 0830673, and 0528414.

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 403–415, 2011.

1 Introduction

Sometimes, generating difficult instances of computational problems seems too easy. Of course this can be a boon for cryptography, but on the other hand can frustrate machine learning theorists, since it implies that hard-to-learn instances of learning problems can arise easily.

Consider for instance Goldreich's simple random number generator: it takes a random string $\mathbf{u}$ of length n and a fixed predicate f on $d = O(1)$ bits that is balanced (i.e., $|f^{-1}(0)| = |f^{-1}(1)|$), and applies f on m random (or pseudorandom) subsets of $\mathbf{u}$. Can the resulting string of m bits be distinguished from a random m-bit string? If not, we have a pseudorandom generator, and in particular can conclude that the task of *learning* $\mathbf{u}$ from the resulting string is also hard.

Or to take another example, consider the *Learning parities with noise* (LPN) problem: we are given a set of data points $(\mathbf{a_1}, b_1), (\mathbf{a_2}, b_2) \ldots$, where $\mathbf{a_i} \in \mathrm{GF}(2)^n$ and $b_i \in \mathrm{GF}(2)$. Each $\mathbf{a_i}$ was chosen randomly. The corresponding b_i was computed as $\mathbf{a_i} \cdot \mathbf{u} + \eta_i$, where $\mathbf{u} \in \mathrm{GF}(2)^n$ is a *secret* vector, and $\eta_i \in \mathrm{GF}(2)$ is a *noise* bit that is 1 with some probability p. (All $\mathbf{a_i}$'s and η_i's are iid.) The goal is to distinguish these m bits from a truly random string. It can be shown that such a distinguisher also allows us to recover $\mathbf{u}$ with high probability. All evidence suggests that this recovery problem is hard, and this conjectured hardness is the basis of some cryptosystems [4,7,13]. The best algorithm to recover $\mathbf{u}$ runs in $2^{O(n/\log n)}$ time (Blum et al. [8]), only very slightly better than the trivial $2^{O(n)}$. The hardness of the LPN problem has important implications for machine learning. A standard technique to make learning algorithms noise-tolerant is using Kearns [14]'s *statistical query* (SQ) model, but parity is perhaps the simplest concept class for which the approach fails (and must fail because parity has exponential SQ dimension). This is frustrating, because the algorithm for learning parities in the noise-free setting is so simple and canonical: gaussian elimination (i.e., solving linear equations). If even this algorithm cannot be made noise-tolerant then the prospects for more complicated learning algorithms seem bleak.

Clearly, we need more algorithmic ideas for this type of problems. Even if they do not always work, they may help us understand parameter ranges where the problem transitions from hard to easy. Sometimes the success/failure of specific algorithms can lead to interesting new research questions, as happened with phase transitions for Random3SAT. If the problem is being used in cryptography, then having several candidate algorithms —and the parameter values for which they fail—is useful for setting parameter values in practice. Regev's *Learning with Errors* (LWE) problem, a generalization of LPN, provides a good example. This problem has become ubiquitous in new cryptographic research in the last few years, since it shares similar hardness properties as lattice-based cryptosystems a la Ajtai [1], Ajtai-Dwork [2] and others, while having more algebraic structure which enables new uses (e.g., oblivious transfer protocol by Peikert et al. [17], and a leakage-resistant crytosystem by Akavia et al. [3]; see Micciancio and Regev [15] for a survey). There are no nontrivial algorithms for this problem, and it is known to become as hard as (worst-case) approximate lattice vector

problems when the *noise* parameter is more than $\sqrt{n}$. Does it have any nontrivial algorithms for noise below $\sqrt{n}$ (or for that matter, any other noise rate)? This is not an idle question since the problem structure could be quite different for different noise rates; for example in case of LPN the known construction for public-key cryptosystems from LPN [4] need to assume the hardness of LPN for noise rate below $1/\sqrt{n}$.

In this paper we use an algorithmic technique called *linearization* to make progress on some of these questions. The main idea is an obvious one and similar to the linearization technique used in practical cryptanalysis [6]: given a nonlinear equation that holds among some variables, introduce new variables for the monomials and obtain a linear system in the new variables. Similar linearization ideas underlie Sherali-Adams lift and project methods in linear programming. Also, Friedl et. al.[11] used a similar approach to solve special polynomial equations in GF(q). Our contribution is to fruitfully apply this idea to the above settings and introduce new ways of analysis that lead to provable running times. We hope this will inspire new results.

To illustrate our idea in the simplest way we introduce a variant on LPN that does not appear to have been considered before and that, surprisingly, turns out to be tractable using our methods. We call it the *LPN problem with structured noise*.

In the standard noise model, pushing an oracle button gives us a random $\mathbf{a} \in \mathrm{GF}(2)^n$ and a corresponding noisy value $b \in \mathrm{GF}(2)$ that is wrong with probability p. In our model, pushing the oracle button returns m random vectors $\mathbf{a_1}, \mathbf{a_2}, \ldots, \mathbf{a_m} \in \mathrm{GF}(2)^n$ and m bits $b_1, b_2, \ldots, b_m$, of which at most p fraction are noisy (in other words, for at least $(1-p)m$ of the indices i, we are guaranteed $\mathbf{a_i} \cdot \mathbf{u} = b_i$). This may be viewed as a guarantee that the data errors, though still happening p fraction of the time, are not arbitrarily bursty. Every sample returned by the oracle contains untainted data. Perhaps this and other structured noise models should be considered in machine learning since the standard noise model has often led to intractable problems. Our algorithm for LPN with structured noise runs in time that is polynomial in n^m, which is polynomial if m is a constant. Interestingly, it works even if the oracle is allowed to look at $\mathbf{a_i}$'s while deciding where to add noise. It also works for more general "noise patterns" as described later.

The idea in our algorithm is fairly simple: the above structured noise model implies that the hidden secret $\mathbf{u}$ is the solution to a system of degree-m constraints. These degree-m constraints are *linearized* by replacing monomials $\prod_{i \in S} \mathbf{u}_i$ with a new variables y_S, thus obtaining a linear constraint in $\binom{n}{m}$ variables. Then we prove that the resulting system has a unique solution. We also give an exact characterization of noise-models for which our technique works.

We also apply the linearization technique to give a subexponential algorithm for the LWE problem with noise rate below $\sqrt{n}$, which clarifies why existing hardness results (which show that LWE is as hard as lattice approximation [18,16]) did not extend to this case.

Then we apply the linearization technique to an important subcase of Goldreich's generator, whereby f = MAJORITY of three XORs. This was proposed by Applebaum, Barak and Wigderson [5] as a suitable choice of f following some earlier attacks on Goldreich's generator for "structured" f's (Bogdanov and Qiao [9]). We show that if $m = \tilde{\Omega}(n^2)$ then the secret vector $\mathbf{u}$ can be efficiently learned from the output of the generator. (Note that Applebaum et al. had proposed $m = n^{1.1}$ as a safe parameter choice, which is not contradicted by our results. However, thus far there was no result indicating that even $m = n^{d/3}$ is an unsafe choice.)

2 Learning Parities with Structured Noise

2.1 Problem Definition

This section introduces our linearization technique in context of the LPN problem. The following is the most general kind of oracle our algorithm can work with. The oracle has an associated nonzero polynomial P in m variables over GF(2). Any $\eta \in \mathrm{GF}(2)^m$ such that $P(\eta) = 0$ is an *acceptable noise pattern*. Each time a button is pressed the oracle picks m random $\mathbf{a_1}, \mathbf{a_2}, \ldots, \mathbf{a_m} \in \mathrm{GF}(2)^n$, and then picks an arbitrary acceptable noise pattern η. Then it outputs $\mathbf{a_1}, \mathbf{a_2}, \ldots, \mathbf{a_m}$ together with m bits $b_1, b_2, \ldots, b_m$ defined as $b_i = \mathbf{a_i} \cdot \mathbf{u} + \eta_i$. For example, the polynomial such that $P(\eta) = 0$ iff η is a vector of hamming weight at most $m/3$ corresponds to the intuitive notion of "noise rate 1/3." This polynomial also satisfies the hypothesis of the main theorem below.

Note that this noise model allows the oracle to look at the $\mathbf{a_i}$'s in choosing the noise pattern (this is analogous to the *agnostic-learning* or *worst-case* noise model of the standard LPN, which Feldman et al. [10] have shown to be equivalent to the *white noise* case.)

Our algorithm requires $P(\cdot)$ to be such that there is at least one $\rho \in \mathrm{GF}(2)^m$ such that $\rho \neq \eta + \eta'$ for all η, η' satisfying $P(\eta) = P(\eta') = 0$. For example, we could have $P(\eta) = 0$ iff the number of 1's in η is fewer than $m/2 - 1$, corresponding to a noise rate of less than 1/2. But one can conceive of more exotic noise polynomials.

Theorem 1 (Main). *Suppose $P(\cdot)$ is such that there is at least one $\rho \in GF(2)^m$ such that $\rho \neq \eta + \eta'$ for all η, η' satisfying $P(\eta) = P(\eta') = 0$. Then there is an algorithm that learns the secret $\mathbf{u}$ in time $\mathrm{poly}(n^m)$ time.*

In this section we do a simpler version of Theorem 1 where the oracle's noise pattern η is not allowed to depend upon the $\mathbf{a}$ vectors. We think of this as *white noise*. The proof of the general result appears in the full version.

The learning algorithm has access to an oracle $Q(\mathbf{u}, m, P, \mu)$, where $\mathbf{u} \in \mathrm{GF}(2)^n$ is the secret vector. Let m be an integer and $P(\eta_1, \eta_2, ..., \eta_m)$ be a multilinear polynomial over the vector $\eta \in \mathrm{GF}(2)^m$ of degree d (P cannot be

identically 0). Let μ be a distribution over the values of η that satisfies $P(\eta) = 0$. The algorithm knows m and P, and tries to compute the secret $\mathbf{u}$ by querying the oracle. (The algorithm does not know the distribution μ).

When queried by the algorithm, the oracle with secret $\mathbf{u} \in \mathrm{GF}(2)^n$ returns m random vectors $\mathbf{a_1}, \mathbf{a_2}, \ldots, \mathbf{a_m} \in \mathrm{GF}(2)^n$ together with m bits $b_1, b_2, \ldots, b_m \in \mathrm{GF}(2)$. The b values are determined by first choosing a random vector η that satisfies $P(\eta) = 0$ using the distribution μ, then computing $b_i = \mathbf{a_i} \cdot \mathbf{u} + \eta_i$.

Many structures can be expressed in this framework, for example, "at least one equation $\mathbf{a_i} \cdot \mathbf{u} = b_i$ is satisfied" can be expressed by $P(\eta) = \prod_{i=1}^{m} \eta_i$. In fact, any "structure" within the group of size m that excludes at least one value of η can be expressed by a non-zero polynomial.

2.2 Constraints and Linearization

Now we describe the linearization in the context of LPN. Let $\mathbf{x}$ be an n dimensional vector, where $\mathbf{x}_i$'s are considered to be variables. The algorithm uses the sample returned by the oracle to write a polynomial constraint for the variables $\mathbf{x}_i$'s. Ideally the solution to the equations will be $\mathbf{x} = \mathbf{u}$. First write the formula for η that is known to be satisfied given the oracle Q: $P(\eta_1, \eta_2, ..., \eta_m) = 0$. Then substitute $\eta_i = \mathbf{a_i} \cdot \mathbf{x} + b_i$ to obtain

$$P(\mathbf{a_1} \cdot \mathbf{x} + b_1, \mathbf{a_2} \cdot \mathbf{x} + b_2, \ldots, \mathbf{a_m} \cdot \mathbf{x} + b_m) = 0, \tag{1}$$

which is a degree d polynomial constraint over the variable vector $\mathbf{x}$ that is always satisfied by the samples obtained from the oracle when $\mathbf{x} = \mathbf{u}$. Since $\mathbf{x}_i^2 = \mathbf{x}_i$ in GF(2), without loss of generality such a polynomial is multilinear.

Now convert this polynomial constraint to a linear one by the obvious linearization: For each $S \subseteq [n]$ and $|S| \leq d$ replace the monomial $\prod_{i \in S} \mathbf{x}_i$ by the new variable y_S. Thus (1) turns into a linear constraint in a vector y of $N = \sum_{i=1}^{d} \binom{n}{i}$ new variables. Ideally the solution should have y being the "tensor" of $\mathbf{u}$, that is, $y_S = \prod_{i \in S} \mathbf{u}_i$, and this is what we will show.

We define the linearization process more formally:

Definition 2 (Linearization) *Let $p(\mathbf{x}) = \sum_{S \subseteq [n], |S| \leq d} c_S \prod_{i \in S} \mathbf{x}_i$, be a multilinear polynomial of degree d in n variables where the coefficients c_S's are in GF(2). The* linearization *of p, denoted $L(p)$, is a linear function over the variables y_S, where S ranges over subsets of $[n]$ of size at most d:*

$$L(p) = \sum_{S \subseteq [n], |S| \leq d} c_S y_S.$$

We will assume there is a variable $y_\emptyset$ that is always 1, so the number of new variables is $N + 1 = \sum_{i=0}^{d} \binom{n}{i}$

The linearization procedure has a few nice properties because it relies on the unique representation of multilinear polynomials using monomials. In particular, if two polynomials P, Q are the same, their linearizations $L(P)$ and $L(Q)$ are also identical. Also, the linearization for $P + Q$ is the sum of linearizations of P and Q ($L(P + Q) = L(P) + L(Q)$).

Our learning algorithm will take poly(N) samples from the oracle, thus obtaining a linear system in N variables and poly(N) constraints. Each constraint is the linearization of Equation (1):

$$L(P(\mathbf{a_1} \cdot \mathbf{x} + b_1, \mathbf{a_2} \cdot \mathbf{x} + b_2, \ldots, \mathbf{a_m} \cdot \mathbf{x} + b_m)) = 0. \quad (2)$$

2.3 Proof for White Noise Case

Note that the set of constraints always has at least one solution, namely, the vector y obtained from the hidden vector $\mathbf{u}$ ($y_S = \prod_{i \in S} \mathbf{u}_i$). Here we will show a simplified proof which relies heavily on the structure of the learning parities problem; a proof that is more generalizable to other settings can be found in the full version. The key lemma in the proof states that when the $\mathbf{a}$ vectors and b values are chosen uniformly at random, the constraint is violated with significant probability no matter what value y takes.

Lemma 1. *When the vectors* $\mathbf{a_i}$ *and values* b_i *are all independent and uniformly random, the probability that any particular solution* y *satisfies a single constraint is at most* $1 - 2^{-d}$.

Proof. To prove this lemma, we observe that once we fix the solution vector y, the left hand side of Constraint (2) becomes a polynomial of degree d over the components of $\mathbf{a_i}$ and values b_i. By assumption these variables are uniformly and independently chosen at random. For such polynomials, we have the following version of Schwartz-Zippel Lemma, which is essentially the fact that degree d Reed-Muller Codes over GF(2) have distance 2^{-d}. We call it "Schwartz-Zippel" because that is the name given to a similar lemma over GF(q).

Lemma 2 (Binary Schwartz-Zippel). *If* P *is a nonzero multilinear polynomial over GF*(2) *of degree* d, *then* $\mathbf{Pr}_{\mathbf{x} \in U}[P(\mathbf{x}) \neq 0] \geq 2^{-d}$.

If the polynomial we get after fixing y is always nonzero, by Schwartz-Zippel we can already conclude that the probability of satisfying Constraint (2) is at most $1 - 2^{-d}$. Hence we only need to show the polynomial is never constant 0.

We write the polynomial $P(\eta)$ as the sum of monomials:$P(\eta) = \sum_{S \subseteq [n], |S| \leq d} c_S \cdot \prod_{i \in S} \eta_i$. Here $c_S \in$ GF(2) are indicator variables showing whether the corresponding monomials appear in the polynomial P. By linearity, the left hand side of Constraint (2) can be written as: $\sum_{S \subseteq [n], |S| \leq d} c_S L(\prod_{i \in S}(\mathbf{a_i} \cdot \mathbf{x} + b_i))$.

Let S be one of the largest sets with $c_S = 1$, then the monomial $\prod_{i \in S} b_i$ will appear in the linearization of the corresponding term $\prod_{i \in S}(\mathbf{a_i} \cdot \mathbf{x} + b_i)$. Notice that the same monomial cannot appear anywhere else. If it appears in the linearization for another monomial $\prod_{i \in T}(\mathbf{a_i} \cdot \mathbf{x} + b_i)$, then we must have $S \subseteq T$ and $S \neq T$, so $|T| > |S|$, which contradicts with the assumption that S is the largest set. So the polynomial is always nonzero no matter how the vector y is chosen, and by Binary Schwartz-Zippel we know the probability of Constraint (2) being satisfied is at most $1 - 2^{-d}$.

Based on this lemma, the following algorithm can learn the secret vector $\mathbf{u}$.

Algorithm 1

1. *Query the oracle* $10N2^d$ *times*
2. *Repeat for all index* i *from 1 to* n
3. *Replace the* i*-th component of* $\mathbf{a}$ *vectors with fresh random bits.*
4. *Try to solve the equations generated by the new* $\mathbf{a}$*'s and original* b*'s*
5. *If there's no solution,* $\mathbf{u}_i = 1$*, otherwise* $\mathbf{u}_i = 0$*.*
6. *Restore the original values for* $\mathbf{a}$ *vectors*

It's clear that the algorithm runs in poly(N) time since $2^d \leq N$, we shall prove that it computes the correct vector $\mathbf{u}$ with high probability.

Proof. First suppose $\mathbf{u}_j = 0$. Then let $\hat{\mathbf{a}}$ be the modified $\mathbf{a}$ vectors. Even after we replace the j-th component of $\mathbf{a}$ vectors with fresh random bits, we still have $b_j = \hat{\mathbf{a}}_\mathbf{i} \cdot \mathbf{u} + \eta_j$ because $\hat{\mathbf{a}}_\mathbf{i}$ and $\mathbf{a}_\mathbf{i}$ only differ in the j-th component. As we discussed before Constraint (2) is always satisfied by the solution $y_S = \prod_{j \in S} \mathbf{u}_j$, so the algorithm will find a solution and conclude $\mathbf{u}_j = 0$.

Now let $\mathbf{u}_j = 1$ and let t be the j-th component of $\mathbf{a}_\mathbf{i}$, and $\hat{t}$ be the j-th component of $\hat{\mathbf{a}}_\mathbf{i}$. Then we have $b_i = \hat{\mathbf{a}}_\mathbf{i} \cdot \mathbf{u} + t + \hat{t} + \eta_i$. Notice that t is independent from anything in $\hat{\mathbf{a}}$, and is uniformly $0, 1$ with probability $1/2$. Thus in this case the vectors $\hat{\mathbf{a}}_\mathbf{i}$ and values b_i are all independent and uniformly chosen at random. By Lemma 1 we know for any fixed assignment of y, the probability that it satisfies one constraint is at most $1 - 2^{-d}$. But we have $10N2^d$ independent constraints, the probability that all those constraints are satisfied simultaneously is at most $(1 - 2^{-d})^{10N2^d} \ll e^{-N}$. There are 2^N different values for y, by union bound, the probability that at least one of them satisfies all the constraints (so the system of linear equations has solution) is at most $2^N e^{-N} \ll 1/n$. Therefore the algorithm will correctly decide $\mathbf{u}_j = 1$.

3 Learning with Errors

The learning with errors problem is a generalization of LPN to a larger field. There is an unknown vector $\mathbf{u} \in \mathrm{GF}(q)^n$ where q is a prime number that is usually a polynomial of n (but can be large in some applications). The algorithm is given noisy samples of the type $\mathbf{a} \cdot \mathbf{u} + \eta$ where the noise η is generated according to the *Discrete Gaussian* distribution Ψ_α with standard deviation αq, which consists of picking a random real number r from the (usual) Gaussian distribution with standard deviation $\sigma = \alpha q$ (i.e., with density function $D_\sigma(r) = 1/\sigma \exp(-(\pi r/\sigma)^2)$), then rounding it up to the nearest integer $\lfloor r \rceil$ and outputting $\lfloor r \rceil \pmod q$. The algorithm is given access to an oracle $Q(\mathbf{u}, \Psi_\alpha)$. which picks $a \in \mathrm{GF}(q)^n$ uniformly at random, then picks η independent of a from distribution Ψ_α, and returns $\mathbf{a}, b$ where $b = \mathbf{a} \cdot \mathbf{u} + \eta$. It is conjectured that no polynomial time algorithm can learn the secret $\mathbf{u}$ for some specific parameters, and the best known algorithm works in exponential time. We give the first subexponential algorithm for LWE:

Theorem 3. *When $\alpha q = n^{\varepsilon}$ where ε is a constant strictly smaller than $1/2$, and $q \gg (\alpha q \log n)^2$, there is a $2^{\tilde{O}(n^{2\varepsilon})}$ time algorithm algorithm that learns* $\mathbf{u}$ *when given access to the oracle $Q(\mathbf{u}, \Psi_\alpha)$.*

The theorem is derived by showing that the low error case fits in our "structured noise" setting, for which we show a subexponential algorithm. In the structured noise model the algorithm has access to a different oracle $Q(\mathbf{u}, \Psi_\alpha, d)$. The new parameter d is an integer that is considered to be the "bound" of the error η and satisfies $4d + 1 < q$. The oracle ensures that the noise η is picked uniformly from Ψ_α conditional on it being at most d in magnitude. (We always interpret η as an integer between $-(q-1)/2$ and $(q-1)/2$.)

We will rely on the well known fact about the gaussian distribution:

Lemma 3. *For $\eta \in [-(q-1)/2, (q-1)/2]$, we have $\mathbf{Pr}_{\eta \sim \Psi_\alpha}[|\eta| > k\alpha q] \leq e^{-O(k^2)}$.*

This implies (we omit the simple calculation) that if we draw fewer than $e^{o(k^2)}$ samples from the standard oracle then these are statistically very close to samples from the structured noise oracle $Q(\mathbf{u}, \Psi_\alpha, d)$ for $d = k\alpha q$. Thus from now on we will assume that samples are drawn from the latter.

The algorithm works similarly as the algorithm for LPN, with an additional twist needed because the field is not GF(2), so the underlying polynomials are non-multilinear. We first write a univariate degree $2d + 1$ polynomial P such that $P(\eta) = 0$ whenever η is drawn from oracle $Q(\mathbf{u}, \Psi_\alpha, d)$ (and thus $|\eta| \leq d$): $P(\eta) = \eta \prod_{i=1}^{d} (\eta + i)(\eta - i)$.

From now on we use similar notation as in the LPN section. We use $\mathbf{x}$ (an n-dimensional vector) as variables, and try to write a system of equations which whp have solutions that allow us to recover $\mathbf{u}$. To do this we substitute $\eta = \mathbf{a} \cdot \mathbf{x} + b$ in the polynomial $P(\eta)$ to obtain a degree $2d + 1$ polynomial over the variables $\mathbf{x}_i$: $(\mathbf{a} \cdot \mathbf{x} + b) \prod_{i=1}^{d} (\mathbf{a} \cdot \mathbf{x} + b + i)(\mathbf{a} \cdot \mathbf{x} + b - i) = 0$.

This constraint is always satisfied if $\mathbf{x} = \mathbf{u}$. Let $D = 2d + 1$ denote the degree of the polynomial $P(\eta)$. Finally we linearize this equation using variable vector y that is indexed by vectors $\mathbf{v} \in \mathbf{Z}^n$ such that $1 \leq \sum_{i=1}^{n} \mathbf{v}_i \leq D$. (Although v also has dimension n, we will not use bold font because it usually appears in the subscript.) The variable $y_{\mathbf{v}}$ corresponds to the monomial $\prod_{i=1}^{n} \mathbf{x}_i^{\mathbf{v}_i}$. We denote the degree of this monomial namely $\sum_{i=1}^{n} \mathbf{v}_i$ as $deg(\mathbf{v})$ or $deg(y_{\mathbf{v}})$. For simplicity we add one component $y_{\mathbf{0}}$, which always has the value 1. The number of variables is $N = \binom{n+D}{n}$. We define y^k to be the vector of all the variables with degree k. Thus $y = (1, y^1, y^2, ..., y^D)$.

The new linearization operator L replaces each monomial in the polynomial with the corresponding y variable. The linearized equation will be a linear constraint on the y variables, $L\left(P(\mathbf{a} \cdot \mathbf{x} + b)\right) = 0$.

Again we observe that this constraint is always satisfied by the solution generated by $\mathbf{u}$: $y_{\mathbf{v}} = \prod_{i=1}^{n} \mathbf{u}_i^{\mathbf{v}_i}$. Since this is similar to the white noise case in LPN, we show a lemma similar to Lemma 1: when $\mathbf{a}$ and b are independent and uniformly random, a large number of constraints of this form cannot be satisfied simultaneously.

Lemma 4. *When the vector* $\mathbf{a}$ *and the values* b *are independent and uniformly random, the probability that any particular solution* y *satisfies a single constraint is at most* $1 - 1/q$.

Proof. Similar to the proof of Lemma 1, we again observe that once y is fixed, the left hand side of the equation $(L\,(P(\mathbf{a} \cdot \mathbf{x} + b)))$ is a polynomial over $\mathbf{a}(i)$'s and b of degree at most D. This polynomial is clearly nonzero because no matter what value y has, the coefficient for b^D is always 1 (to get b^D, it's essential to choose b in all factors of the product $(\mathbf{a} \cdot \mathbf{x} + b) \prod_{i=1}^{d} (\mathbf{a} \cdot \mathbf{x} + b + i)(\mathbf{a} \cdot \mathbf{x} + b - i)$). So we can always apply the standard version of Schwartz-Zippel Lemma: If p is a nonzero polynomial over $\mathrm{GF}(q)$ of degree d, then $\mathbf{Pr}_{x \in U}[p(x) = 0] \leq d/q$.

Since the polynomial we get after fixing y is always nonzero, and $D < q$, the probability that the constraint is satified is at most $D/q \leq 1 - 1/q$.

According to this lemma, we use the following algorithm to learn the secret vector $\mathbf{u}$.

Algorithm 2

1. *Query the oracle* $10Nq \log q$ *times*
2. *Repeat for all index* i *from 1 to* n
3. *Try for all* k *from 0 to* $q - 1$
4. *Guess* $\mathbf{x}_i = k$.
5. *Let* $b = b - \mathbf{a}(i) \cdot k$ *for all oracle answers*
6. *Replace the* i*-th component of* $\mathbf{a}$ *vectors with new independent uniform random value in GF*(q).
7. *Try to solve the equations generated by the modified* $\mathbf{a}$*'s and* b*'s*
8. *If there's no solution, the guess is incorrect; otherwise it is correct*
9. *Restore the original values for* $\mathbf{a}$ *vectors and* b *values*

We will show that with high probability the algorithm will get the correct value of $\mathbf{u}$.

Proof. Let $\hat{\mathbf{a}}$ be the modified a vector and $\hat{b}$ be the modified b value. Let $\hat{\mathbf{u}}$ be the vector $\mathbf{u}$ except that the i-th component is changed to 0.

If $\mathbf{x}_i = k$, we have $\hat{b} = \mathbf{a} \cdot \mathbf{u} + \eta - \mathbf{u}(i)\mathbf{a}(i) = \hat{\mathbf{a}}\hat{\mathbf{u}} + \eta$. Therefore $\hat{\mathbf{a}}$ and $\hat{b}$ form a valid oracle answer, the system of linear equations always has the solution $y_{\mathbf{v}} = \prod_{j=1}^{n} \hat{\mathbf{u}}_j^{\mathbf{v}_j}$.

If $\mathbf{x}_i \neq k$, we have $\hat{b} = \mathbf{a} \cdot \mathbf{u} + \eta - k\mathbf{a}(i) = \hat{\mathbf{a}}\hat{\mathbf{u}} + \eta + (\mathbf{u}(i) - k)\mathbf{a}(i)$. Since $\mathbf{u}(i) - k$ is nonzero and $\mathbf{a}(i)$ is independent with everything in $\hat{\mathbf{a}}$, the value $\hat{b}$ is independent of $\hat{\mathbf{a}}$. For any solution vector y, by Lemma 4 the probability that it satisfies a single constraint is at most $1 - 1/q$. Since we have $10Nq \log q$ independent constraints, the probability that all of them are satisified is at most $(1-1/q)^{10Nq \log q} \ll q^{-2N}$. There are q^N different solutions, by union bound that at least one of them satisfies all the constraints is at most $q^{-2N} q^N = q^{-N} \ll 1/nq$. Thus with high probability the algorithm will be able to learn the secret vector $\mathbf{u}$.

The running time of this algorithm is proportional to q, which in some cases can be quite large. In the full version of the paper we give a better algorithm whose running time only depends on $\log q$.

4 Learning the Majority of Parities Function

Applebaum et al. [5] proposed the"DSF assumption" and showed how to use it (with other assumptions) to build public-key cryptosystems. Let $\mathcal{M}_{m,n,d}$ be a random bipartite graph with m vertices on top, n vertices at the bottom, and d edges from each top vertex. If G is such a graph, f is a function $f : \{0,1\}^d \to \{0,1\}$, define the function $G_f : \{0,1\}^n \to \{0,1\}^m$ to be the function obtained by mapping every $\mathbf{u} \in \{0,1\}^n$ to $(f(\mathbf{u}_{\Gamma(1)}), \ldots, f(\mathbf{u}_{\Gamma(m)}))$, where $\Gamma(i)$ denotes the neighbors of the i-th "top" vertex. The DSF assumption asserts the existence of a function f such that $(G, G_f(U_n))$ is ε-indistinguishable from the distribution (G, U_m) when $G \in_R \mathcal{M}_{m,n,d}$. Here d is a large constant. When $m = n$ this is conjectured by Goldreich [12], but in [5] m is required to be super-linear.

To avoid known attacks by Bogdanov and Qiao [9], Applebaum et al. suggested the "majority of three parities" (that is, f is the majority of three parities on $d/3$ bits each) as a candidate function for f. Indeed, they showed when $m = O(n^{1.1})$, this function has nice properties such as looking pseudorandom for AC^0 circuits (and the proofs can be generalized when $m = o(n^2)$). However, using our algorithm we can show when $m = \Omega(n^2 \log n)$, the function G_f fails to be a pseudorandom generator in a really severe way: not only is the output no longer indistinguishable from uniform distribution, but the function G_f is also invertible with high probability. Notice the claim is that $(G, G_f(U_n))$ is indistinguishable with (G, U_m), if instead we consider $(\mathbf{u}, G_f(\mathbf{u}))$ then clearly it cannot be pseudorandom by simple dimension arguments.

Our algorithm is designed by noting that the "majority of parities" function can actually be viewed as an answer given by a learning parities with structured noise oracle. Given $Maj(\mathbf{a_1} \cdot \mathbf{u}, \mathbf{a_2} \cdot \mathbf{u}, \mathbf{a_3} \cdot \mathbf{u}) = b$, where $\mathbf{u} \in \{0,1\}^n$ is the input, and $\mathbf{a_1}$, $\mathbf{a_2}$, $\mathbf{a_3}$ are vectors with exactly $d/3$ 1's that are obtained when applying G_f, we can write a group of linear equations: $\mathbf{a_1} \cdot \mathbf{u} = b$, $\mathbf{a_2} \cdot \mathbf{u} = b$, $\mathbf{a_3} \cdot \mathbf{u} = b$. Since b is the $m$ajority of these three parities, we know that at least two out of the three equations are satisfied. This is exactly the kind of structure our LPN algorithm can work with. We can represent this structure by $P(\eta) = \eta_1\eta_2 + \eta_1\eta_3 + \eta_2\eta_3$ where $\eta_i = \mathbf{a_i} \cdot \mathbf{u} + b$. We could try to use our earlier LPN algorithm, except the earlier analysis does not apply because the a vectors are not uniformly random —each of $\mathbf{a_1}, \mathbf{a_2}, \mathbf{a_3}$ is randomly chosen from all vectors that have $d/3$ 1's, and have disjoint support. Also, once we fix the input $\mathbf{u}$, the error η is dependent on the a vectors, which we will need to deal with it differently than in Section 2. We will see that all our calculations become easier when $m = n^4$ instead of $m = n^2 \log n$.

The algorithm will be analogous to our LPN algorithm. We expand $P(\mathbf{a_1} \cdot (\mathbf{x}+\mathbf{u}) + \eta_1, \mathbf{a_1} \cdot (\mathbf{x}+\mathbf{u}) + \eta_2, \mathbf{a_1} \cdot (\mathbf{x}+\mathbf{u}) + \eta_3)$ explicitly and treat $(\mathbf{x}+\mathbf{u})$ as a formal variable:

$$(\mathbf{a_1}\otimes\mathbf{a_2}+\mathbf{a_1}\otimes\mathbf{a_3}+\mathbf{a_3}\otimes\mathbf{a_2})\cdot Y^2+((\eta_1+\eta_2)\mathbf{a_3}+(\eta_2+\eta_3)\mathbf{a_1}+(\eta_1+\eta_3)\mathbf{a_2})\cdot Y^1=0, \tag{3}$$

where Y^2, Y^1 are linearizations of $(\mathbf{x}+\mathbf{u})\otimes(\mathbf{x}+\mathbf{u})$ and $(\mathbf{x}+\mathbf{u})$ respectively. Unlike in case of LPN, having enough equations of this form *does not* guarantee a unique solution. Nevertheless we can show that the solutions to the system of linear equations allow us to learn the secret $\mathbf{u}$.

A key observation is $\eta_1+\eta_2=(\mathbf{a_1}+\mathbf{a_2})\cdot\mathbf{u}$. Indeed, when $(\mathbf{a_1}+\mathbf{a_2})\cdot\mathbf{u}=0$, it means $\mathbf{a_1}\cdot\mathbf{u}=\mathbf{a_2}\cdot\mathbf{u}$, so $\eta_1=\eta_2=0$; when $(\mathbf{a_1}+\mathbf{a_2})\cdot\mathbf{u}=1$, it means $\mathbf{a_1}\cdot\mathbf{u}\neq\mathbf{a_2}\cdot\mathbf{u}$, so exactly one of the equations is incorrect, and we have $\eta_1+\eta_2=1$. Applying this observation to (3), we get

$$(\mathbf{a_1}\otimes\mathbf{a_2}+\mathbf{a_1}\otimes\mathbf{a_3}+\mathbf{a_3}\otimes\mathbf{a_2})\cdot(Y^2+\mathbf{u}\otimes Y^1+Y^1\otimes\mathbf{u})=0. \tag{4}$$

Let W denote $(Y^2+\mathbf{u}\otimes Y^1+Y^1\otimes\mathbf{u})$. Since the diagonal entries of W do not affect the equation we will assume $W_{i,i}=0$ below. For distinct $p,q,s,t\in[n]$, let $W_{(p,q)\times(s,t)}$ denote the sum $W_{p,s}+W_{p,t}+W_{q,s}+W_{q,t}$.

Claim 1: *There is a polynomial-time algorithm that, given any solution Y^2, Y^1 for which $W_{(p,q)\times(s,t)}=0$ for all p,q,s,t, finds the secret $\mathbf{u}$.*

The proof is simple after observing $y_{\{p,s\}}+y_{\{p,t\}}+y_{\{q,s\}}+y_{\{q,t\}}=\mathbf{u}_p\mathbf{u}_s+\mathbf{u}_p\mathbf{u}_t+\mathbf{u}_q\mathbf{u}_s+\mathbf{u}_q\mathbf{u}_t$. Due to space constraint we leave the full proof in the full version of the paper. Now we give the simpler analysis for $m=O(n^4)$.

Claim 2: *$O(n^4)$ equations suffice whp to rule out all solutions in which $W_{(p,q)\times(s,t)}=1$ for some p,q,s,t.*

Proof. We show that if p,q,s,t are such that $W_{(p,q)\times(s,t)}=1$, then Equation (4) is violated with probability $\Omega(1/n^2)$. Since the number of possible solutions W is 2^{n^2} a simple union bound yields the claim.

Let A_i denote the set whose indicator vector is $\mathbf{a_i}$; thus A_i is a random set of size $d/3$ and A_1, A_2, A_3 are disjoint. Consider the event E that $|A_1\bigcap\{p,q,s,t\}|=0$, $|A_2\bigcap\{p,q\}|=|A_3\bigcap\{s,t\}|=1$, $|A_2\bigcap\{s,t\}|=|A_3\bigcap\{p,q\}|=0$. That is, exactly one of p,q appears in A_2 and exactly one of s,t appears in A_3. It's easy to see that this happens with probability $\Omega(1/n^2)$. Now fix A_1, $A_2\backslash\{p,q\}$, $A_3\backslash\{s,t\}$ and let the corresponding indicator variables be $\mathbf{a_1}$, $\mathbf{a_2}^*$, $\mathbf{a_3}^*$. Now and consider the four possibilities depending upon which of p,q appears in A_2 and which of s,t appear in A_3: either $\mathbf{a_2}=\mathbf{a_2}^*+e_p$ or $\mathbf{a_2}=\mathbf{a_2}^*+e_q$, and either $\mathbf{a_3}=\mathbf{a_3}^*+e_s$ or $\mathbf{a_3}=\mathbf{a_3}^*+e_t$ (e_i is the vector which is 1 only at position i). The sum of the expression $(\mathbf{a_1}\otimes\mathbf{a_2}+\mathbf{a_1}\otimes\mathbf{a_3}+\mathbf{a_3}\otimes\mathbf{a_2})$ over these four possibilities evaluates to exactly the matrix $(e_p+e_q)\otimes(e_s+e_t)$, because all other terms appear even number of times.Therefore the sum of LHS of Equation (4) is exactly $W_{(p,q)\times(s,t)}$. Since $W_{(p,q)\times(s,t)}$ is 1, in at least one of the four cases Equation (4) is violated, and this happens with probability 1/4 conditioned on the event E.

We leave the proof that $O(n^2\log n)$ equations actually suffice for Claim 2 to the full version of this paper.

5 Conclusions

Linearization techniques have been applied in various contexts, and in this paper we manage to give provable bounds for this idea in several contexts such as LPN, LWE, and learning the MAJORITY of PARITIES function.

We also introduced a new structured noise model for LPN problems (and by analogy, for related problems such as learning low-depth decision trees) which is a natural modification to the original LPN problem and seems more tractable. We think such structured noise models should be studied more in machine learning since standard models led to intractable problems.

It should be interesting to apply our techniques to other problems and settings, and to investigate the optimality of our parameter choices. Our algorithm for $m = n^2 \log n$ for the MAJORITY of PARITIES function shows that analysis can sometimes be tightened beyond what a first glance suggests.

An obvious open problem is to relate the new structured noise model with the original LPN problem. This seems difficult, though it worked in the special case of LWE with low noise.

Acknowledgements. We thank several people for useful conversations: Boaz Barak, Avrim Blum, Daniele Micciancio, Oded Regev, David Steurer, Avi Wigderson. We thank an anonymous referee for suggesting the simplified proof in Section 2.3.

References

1. Ajtai, M.: Generating hard instances of lattice problems (extended abstract). In: Proceedings of the 28th Annual ACM Symposium on Theory of Computing (1996)
2. Ajtai, M., Dwork, C.: A public-key cryptosystem with worst-case/average-case equivalence. In: Proceedings of the 29th Annual ACM Symposium on Theory of Computing (1997)
3. Akavia, A., Goldwasser, S., Vaikuntanathan, V.: Simultaneous hardcore bits and cryptography against memory attacks. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 474–495. Springer, Heidelberg (2009)
4. Alekhnovich, M.: More on average case vs approximation complexity. In: Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science (2003)
5. Applebaum, B., Barak, B., Wigderson, A.: Public key cryptography from different assumptions. In: Proceedings of the 42nd Annual ACM Symposium on Theory of Computing (2010)
6. Bard, G.V.: Algebraic Cryptanalysis. Springer, Heidelberg (2009)
7. Blum, A., Furst, M.L., Kearns, M.J., Lipton, R.J.: Cryptographic primitives based on hard learning problems. In: Proceedings of the 13th Annual International Cryptology Conference on Advances in Cryptology (1994)
8. Blum, A., Kalai, A., Wasserman, H.: Noise-tolerant learning, the parity problem, and the statistical query model. Journal of ACM (2003)
9. Bogdanov, A., Qiao, Y.: On the security of goldreich's one-way function. In: Dinur, I., Jansen, K., Naor, J., Rolim, J. (eds.) APPROX 2009. LNCS, vol. 5687, pp. 392–405. Springer, Heidelberg (2009)

10. Feldman, V., Gopalan, P., Khot, S., Ponnuswami, A.K.: New results for learning noisy parities and halfspaces. In: Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (2006)
11. Friedl, K., Ivanyos, G., Magniez, F., Santha, M., Sen, P.: Hidden translation and orbit coset in quantum computing. In: Proceedings of the 35th Annual ACM Symposium on Theory of Computing (2003)
12. Goldreich, O.: Candidate one-way functions based on expander graphs. technical report. TR00-090, Electronic Colloquium on Computational Complexity, ECCC (2000)
13. Hopper, N.J., Blum, M.: Secure human identification protocols. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, p. 52. Springer, Heidelberg (2001)
14. Kearns, M.: Efficient noise-tolerant learning from statistical queries. Journal of ACM (1998)
15. Micciancio, D., Regev, O.: Lattice-based cryptography. In: Post Quantum Cryptography (2009)
16. Peikert, C.: Public-key cryptosystems from the worst-case shortest vector problem. In: Proceedings of 41st ACM Symposium on Theory of Computing (2009)
17. Peikert, C., Vaikuntanathan, V., Waters, B.: A framework for efficient and composable oblivious transfer. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 554–571. Springer, Heidelberg (2008)
18. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. Journal of ACM (2009)

Exact Learning Algorithms, Betting Games, and Circuit Lower Bounds*

Ryan C. Harkins and John M. Hitchcock

Department of Computer Science,
University of Wyoming

Abstract. This paper extends and improves work of Fortnow and Klivans [5], who showed that if a circuit class $\mathcal{C}$ has an efficient learning algorithm in Angluin's model of exact learning via equivalence and membership queries [2], then we have the lower bound $\mathrm{EXP}^{\mathrm{NP}} \not\subseteq \mathcal{C}$. We use entirely different techniques involving betting games [4] to remove the NP oracle and improve the lower bound to $\mathrm{EXP} \not\subseteq \mathcal{C}$. This shows that it is even more difficult to design a learning algorithm for $\mathcal{C}$ than the results of Fortnow and Klivans indicated.

1 Introduction

We continue the line of research basing hardness of learning results on computational complexity and cryptography (see for example [16,8,1,9]). Fortnow and Klivans [5] consider Angluin's model of exact learning from equivalence and membership queries [2]. In an equivalence query, the learner presents a hypothesis and asks if it is equivalent to the unknown target concept. If the hypothesis is equivalent, the learner has succeeded; otherwise, the learner is given an example on which the hypothesis is incorrect. In a membership query, the learner chooses an example and asks the value of the target concept on that example. To succeed, the learner must exactly identify the target concept. Fortnow and Klivans show that learning algorithms for a circuit class give lower bounds for that type of circuit in the class $\mathrm{EXP}^{\mathrm{NP}}$. Throughout this introduction, $\mathcal{C}$ denotes any nonuniform class of polynomial-size circuits. (Several results will be stated informally here but made precise in the body of the paper.)

Theorem 1.1. (Fortnow and Klivans [5]) *If there is an efficient exact learning algorithm for $\mathcal{C}$ using equivalence and membership queries, then* $\mathrm{EXP}^{\mathrm{NP}} \not\subseteq \mathcal{C}$.

While "efficient" typically means a polynomial-time algorithm, the result of Fortnow and Klivans as well as our results allow for exponential time and subexponentially many queries (i.e. $(2^{O(n)})$ time and $2^{n^{o(1)}}$ queries).

* This research was supported in part by NSF grants 0652601 and 0917417 and by an NWO travel grant. Part of this research was done while Hitchcock was on sabbatical at CWI.

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 416–423, 2011.

For the case of exact learning algorithms that only make equivalence queries, a stronger lower bound follows immediately from results [10,6] connecting resource-bounded measure and dimension with Littlestone's model of online mistake-bound learning [11]. Combining these results with the fact that exact learning with equivalence queries is the same as online mistake-bound learning, we have the following, which was also noted in [5].

Theorem 1.2. (Hitchcock [6]) *If there is an efficient exact learning algorithm for* $\mathcal{C}$ *using equivalence queries, then* $\mathrm{EXP} \not\subseteq \mathcal{C}$.

Given these two theorems, it is natural to ask whether we can prove a lower bound in EXP assuming the learning algorithm makes both equivalence and membership queries. Where does the NP oracle come from in Theorem 1.1? The proof of Theorem 1.1 separates into two parts, giving an indirect diagonalization. Assume that $\mathrm{EXP}^{\mathrm{NP}} \subseteq \mathcal{C}$.

(i) Because $\mathcal{C} \subseteq \mathrm{P/poly}$, $\mathrm{EXP}^{\mathrm{NP}}$ collapses and reduces to the Permanent [3,7,15].
(ii) Use the exact learning algorithm to learn the $\mathcal{C}$ circuits for the Permanent. An NP oracle is used to answer the equivalence queries. This yields a P^{NP} algorithm for the Permanent.

Combining (i) and (ii) gives $\mathrm{EXP}^{\mathrm{NP}} \subseteq \mathrm{P}^{\mathrm{NP}}$, a contradiction. Therefore we see that the NP oracle in Theorem 1.1 is for the equivalence queries. In contrast, no NP oracle is required in Theorem 1.2 where equivalence queries are allowed. The proof of Theorem 1.2 relies on martingales and a more direct measure-theoretic diagonalization, which is very different than the double-collapse argument and indirect diagonalization used for Theorem 1.1. This suggests hope that a more direct diagonalization approach may yield the desired improvement.

To improve Theorems 1.1 and 1.2, we must simulate the learning algorithm's queries while performing our diagonalization. Following [6], consider implementing this in the form of a martingale. The equivalence queries are no problem. We simply use the transformation in [11] that converts an equivalence query algorithm to a mistake-bound learning algorithm and apply the technique from [6]. We can therefore focus our effort on the membership queries. Unfortunately, we have been unable to discover a method for simulating membership queries in a martingale, due to the stringent requirement that a martingale must bet on all strings in lexicographic order. However, there is an extension of martingales called betting games that do help.

Buhrman et al. [4] introduced betting games to define a generalization of resource-bounded measure with applications to autoreducibility and the BPP vs. EXP problem. Betting games and martingales are similar; the difference is how the betting strategy is allowed to operate. A martingale is required to consider the strings at each length in lexicographic order. Betting games lift this requirement by allowing the betting strategy to pick the next string that it will bet on. We can easily simulate a membership query in a betting game – the strategy simply asks to make a prediction on the queried string and then it gets

to see the answer for that string. This allows for a more powerful diagonalization and it suffices for our purposes:

Theorem 1.3. *If there is an exact learning algorithm for $\mathcal{C}$ using equivalence and membership queries, then there is a betting game which succeeds on $\mathcal{C}$ in the sense of [4].*

Buhrman et al. showed that it is also possible to diagonalize within EXP against betting games [4], just as is the case for martingales [12]. Formally, no betting game can succeed on all of EXP. Hence the desired improvement, our main result, follows from Theorem 1.3:

Theorem 1.4. *If there is an exact learning algorithm for $\mathcal{C}$ using equivalence and membership queries, then* $\text{EXP} \not\subseteq \mathcal{C}$.

This results shows that designing an exact learning algorithm for many circuit classes $\mathcal{C}$ will be difficult, as for most classes it is an open problem whether $\text{EXP} \subseteq \mathcal{C}$.

This paper is organized as follows. Precise technical definitions for betting games and exact learning are given in section 2. We construct betting games from exact learning algorithms in section 3.

2 Preliminaries

We use standard notation. The binary alphabet is $\Sigma = \{0, 1\}$, the set of all binary strings is Σ^*, the set of all binary strings of length n is Σ^n, and the set of all infinite binary sequences is Σ^∞. The empty string is denoted by λ. We use the standard enumeration of strings, $\lambda, 0, 1, 00, 01, \ldots$, and a total ordering of strings corresponding to this enumeration. A language A can alternatively be seen as a subset of Σ^*, or as an element of Σ^∞ via identification with its characteristic sequence.

2.1 Betting Games

Betting games, which are also called nonmonotonic martingales, originated in the field of algorithmic information theory. In that setting they yield the notion of Kolmogorov-Loveland randomness (generalizing Kolmogorov-Loveland stochasticity) [14,13]. The concept was introduced to computational complexity by Buhrman et al. [4]. Our notation for betting games is taken predominantly from Merkle et al. [13]. First, for comparison, we recall the definition of a martingale:

Definition 2.1. *A* martingale *is a function $d : \Sigma^* \to [0, \infty)$ such that for all $w \in \Sigma^*$, we have the following averaging condition:*

$$d(w) = \frac{d(w0) + d(w1)}{2}.$$

Intuitively, a martingale is betting in order on the characteristic sequence of an unknown language. The martingale starts with finite initial capital $d(\lambda)$. The quantity $d(w)$ represents the current capital the martingale has after betting on the first $|w|$ bits of a sequence that begins with w. The quantities $\pi(w,0) = d(w0)/2d(w)$ and $\pi(w,1) = d(w1)/2d(w)$ represent the fraction of its current capital that the martingale is waging on 0 and 1, respectively, being the next bit of the sequence. This next bit is revealed and the martingale has $d(w0) = 2\pi(w,0)d(w)$ in the case of a 0 and $d(w1) = 2\pi(w,1)d(w)$ in the case of a 1.

Betting games are similar to martingales but have an additional capability of selecting which position in a sequence (or which string in a language) to bet upon next. Because of this added complexity, it is simpler to break the description of a betting game into pieces.

Definition 2.2. *A* betting game *is a system that bets nonmonotonically on an infinite sequence (or a language) and formally is a triple $G = (s, \pi, V)$ consisting of a scan rule s, a stake function π, and a capital function V.*

1. *A* finite assignment *is a sequence $x \in (\Sigma^* \times \Sigma)^*$. In essence, it is a list of strings examined thus far, each string coupled with an assignment, saying whether or not it is included in the language being bet upon. The set of all finite assignments is denoted FA.*
2. *The* scan rule *is a (partial) computable function $s : FA \to \Sigma^*$ from finite assignments to strings that looks at a finite assignment and determines the next string (or bit of a sequence) to bet on. The scan rule is limited in that it cannot select a string that already appears in the current finite assignment.*
3. *The* stake function *is a partial function $\pi : FA \times \Sigma \to [0,1]$. Its function is to examine the current finite assignment and determine what fraction of the capital to bet on either side. It carries a condition that $\pi(w,0) + \pi(w,1) = 1$.*
4. *The* capital function *is a partial function $V : FA \to [0,\infty)$ from finite assignments to nonnegative reals, and utilizes the stake function π. Its initial capital $V(\lambda)$ is finite. When $V(w)$ is defined, the scan rule $s(w)$ determines the next string to bet on, and $\pi(w,b)$ is the stake amount, the capital is updated according to the rule*

$$V(w \cdot (s(w), b)) = 2\pi(w,b)V(w). \tag{2.1}$$

A betting game's capital function also satisfies an averaging condition, in analogy with the definition of a martingale:

$$V(w) = \frac{V(w \cdot (s(w),0)) + V(w \cdot (s(w),1))}{2}.$$

Note that a betting game is a martingale when the scan rule always selects the next string in the lexicographic order.

Definition 2.3. *If a betting game G earns unbounded capital on a language A (in the sense that for every constant c there is a point at which the capital function exceeds c when betting on A), we say that G* succeeds *on A. The* success set *of a betting game G, denoted $S^\infty[G]$, is the set of all languages on which G succeeds.*

The ability of the betting game to examine a sequence nonmonotonically makes determining its running time complicated, since each language can induce a unique computation of the betting game. In other words, the betting game may choose to examine strings in different orders depending upon the language it is wagering against. Buhrman et al. looked at a betting game as an infinite process on a language, rather than a finite process on a string. They used the following definition:

Definition 2.4. *A betting game G runs in time $t(n)$ if for all languages A, every query of length n made by G occurs in the first $t(n)$ steps of the computation.*

Specifically, once a $t(n)$-time-bounded betting game uses $t(n)$ computational steps, its scan rule cannot go back and select any string of length n. We remark that in the results of this paper all betting games have the special form that they bet on all strings of each length before moving on to the next length, so the technical issue of measuring the run time is not important for this paper. In any case, the crucial result is that no exponential-time betting game is successful against the class $\mathrm{EXP} = \mathrm{DTIME}(2^{n^{O(1)}})$.

Theorem 2.5. (Buhrman et al. [4]) *No $2^{n^{O(1)}}$-time betting game succeeds on* EXP.

In our results, we often refer to $2^{O(n)}$-time bounds for sake of comparison to the results of Fortnow and Klivans [5]. We note that our results also hold for $2^{n^{O(1)}}$-time bounds, but for simplicity we prefer to not focus on such minor issues in this paper.

2.2 Exact Learning

In general, a learning algorithm seeks to identify an unknown concept from some known class of concepts. We now review the basic notation and definitions for the exact learning model.

A *concept* is a Boolean function $c_n : \Sigma^n \to \Sigma$. For any string $x \in \Sigma^n$, if $c_n(x) = 1$, then x is positively classified as belonging to the concept, while if $c_n(x) = 0$, then x is classified as not belonging to the concept. A string x paired with the classification $c_n(x)$ is called an *example*. A concept c_n is often identified with the set of positive examples $\{x \mid c_n(x) = 1\} \subseteq \Sigma^n$. A *concept class* $\mathcal{C}_n$ is a set of concepts over Σ^n. A *concept class family* is a sequence $\mathcal{C} = \{\mathcal{C}_n\}_{n \geq 0}$ of concept classes.

A learning algorithm tries to identify a *target concept* drawn from $\mathcal{C}_n$, and often does this by forming a *hypothesis*, which is typically some concept in $\mathcal{C}_n$ that is consistent with (i.e. classifies correctly) all the examples seen thus far. In the exact learning paradigm, a learner $\mathcal{A}$ may make various sorts of queries to a teacher, and then, depending on the answers, formulate a hypothesis. This process repeats until $\mathcal{A}$ has successfully discovered the target concept. We will focus on two types of queries: equivalence queries and membership queries.

Definition 2.6. *An* equivalence query *is a request to the teacher to know if the current hypothesis matches the target concept. If the answer is yes, the teacher responds accordingly. If the answer is no, then the teacher provides the learner with a counterexample (an example that is incorrectly classified by the current hypothesis).*

Definition 2.7. *A* membership query *is a request to the teacher to know the classification of a specific string x. The teacher responds with $c_n(x)$, where c_n is the target concept.*

3 Exact Learning and Betting Games

Theorem 3.1. *Let $\mathcal{C} = \{\mathcal{C}_n \mid n \in \mathbb{N}\}$ be a concept class family, and let*

$$X = \{A \mid (\exists^\infty n) A_{=n} \in \mathcal{C}_n\}.$$

If there is an exact learning algorithm for $\mathcal{C}$ that learns each $\mathcal{C}_n$ in time 2^{cn} and makes no more than 2^{n-2} equivalence and membership queries, then there exists a betting game G that runs in time $O(2^{(c+2)n})$, such that $X \subseteq S^\infty[G]$.

Proof. Let $\mathcal{A}$ be the learning algorithm that learns concepts in $\mathcal{C}$. In other words, for each $n \in \mathbb{N}$, and for any target concept $c_n \in \mathcal{C}_n$, $\mathcal{A}$ can learn c_n using no more than 2^{cn} time, and making at most 2^{n-2} equivalence and membership queries.

Let $G(s, \pi, V)$ be described as follows. G effectively runs in stages, examining strings by length, betting on all strings of length n before betting on any string of size $n+1$. This is proper, since $\mathcal{A}$ learns concepts taken from $\mathcal{C}_n$, whose concepts only classify strings of length n. Therefore, we will apply two subscripts to the stages of calculation, the first to indicate string length, and the second to indicate how many stages have been executed at the string length.

G starts with capital $V_{0,0} = 2$, but it treats its capital as divided up into an infinite number of amounts, 2^{-n} for each n. Thus at each stage $(n, 0)$, the capital effectively is $V_{n,0} = 2^{-n}$ (with all previous winnings "banked" and untouchable). To reflect this, we will divide π in a class of functions $\{\pi_n\}_{n\geq 0}$, so that π_n only touches the capital $V_{n,i}$.

At each stage (n, i), the scan rule s runs the learning algorithm $\mathcal{A}$, which updates its current hypothesis $h_{n,i}$. In the process of formulating $h_{n,i+1}$, one of two things will happen:

- $\mathcal{A}$ will make a membership query to some string $x \in \Sigma^n$
- $\mathcal{A}$ will make an equivalence query

If $\mathcal{A}$ makes a membership query to x, s then checks to see if x occurs in w, the current finite assignment. If so, then s answers the query according to the label. If not, then we set $s(w) = x$ and $\pi_n(w, 0) = \pi_n(w, 1) = 1/2$. In other words, the betting game selects x and makes no bet on the outcome. Once the label for x is revealed, the finite assignment is updated ($w = w \cdot (x, b)$, where b is the label for x). The scan rule then provides the correct classification of x to $\mathcal{A}$ as the

answer to the membership query. The betting game's capital is unchanged and the computation then proceeds onto stage $(n, i+1)$.

If $\mathcal{A}$ makes an equivalence query, then s proceeds in an online fashion. First, the scan rule selects $s(w) = x$, where x is the lexicographically least string in Σ^n that does not appear in w. The stake function computes the prediction $b = h_{n,i}(x)$ using the current hypothesis $h_{n,i}$ and sets $\pi_n(w, b) = 3/4$ and $\pi_n(w, 1-b) = 1/4$. The true classification of x is revealed and $V_{n,i}$ is updated according to (2.1). If $c_n(x) \neq h_{n,i}(x)$, then $(x, 1-b)$ is presented to $\mathcal{A}$ as a counterexample, and computation proceeds to stage $(n, i+1)$. If $c_n(x) = h_{n,i}(x)$, then computation proceeds to stage $(n, i+1)$, and $\mathcal{A}$ is treated as still making an equivalence query. This process repeats until either a counterexample is discovered or the strings of size n are exhausted.

Without loss of generality, we will assume that $\mathcal{A}$ always finishes with an equivalence query to ensure correctness of its hypothesis. Thus if $\mathcal{A}$ has formed the correct hypothesis, G will continue searching for a counterexample until all strings of length n are exhausted, and G moves onto $(n+1, 0)$, where it utilizes $\mathcal{A}$ to learn a new concept.

To examine the running time of G, we note first that $\mathcal{A}$ updates its hypothesis in 2^{cn} time. The remaining time is allotted to the scan rule, which takes only time linear in the size of the current finite assignment (which has size at most 2^{n+1}) to determine a string to select, and to updating the capital function, which can be done in time $O(2^n)$. Hence the aggregate time for G to make all queries of size n is $O(2^n \cdot 2^{cn} \cdot 2^n)$. Therefore G is an $O(2^{(c+2)n})$-time betting game.

To see that G succeeds on X, it suffices to show that for infinitely many n, $V_{n,2^n} \geq 1$. We know that for any $A \in X$, there exist infinitely many n such that $A_{=n} \in \mathcal{C}_n$, and hence for infinitely many n, $\mathcal{A}$ will either correctly learn $A_{=n}$, or at least will classify correctly a sufficiently large number of strings in $A_{=n}$. Thus it suffices to show that for any sufficiently large n, $\mathcal{A}$ learns sufficiently quickly enough to bring in large earnings.

The worst case occurs if all 2^{n-2} queries are equivalence queries, to which a counterexample is ultimately found for each query. (If we allow for membership queries, the bet for each query is 0, and so the capital does not change regardless of the true label. The counterexample to the equivalence query, though, will decrease capital.) Since, by definition, each string of length n will be queried, we have:

$$V_{n,2^n} = V_{n,0} \cdot \left(\frac{1}{2}\right)^{2^{n-2}} \cdot \left(\frac{3}{2}\right)^{3 \cdot 2^{n-2}} = 2^{-n} \left(\frac{27}{16}\right)^{2^{n-2}} \geq 1$$

Hence for infinitely many n, G will "bank" one dollar, and therefore its earnings will increase unbounded. Therefore, $X \subseteq S^\infty[G]$.

Our improvement to the result of Fortnow and Klivans now follows as an immediate corollary to Theorems 2.5 and 3.1:

Corollary 3.2. *Under the assumptions of Theorem 3.1,* $\mathrm{EXP} \not\subseteq X$.

Corollary 3.3. *Let C be any subset of* P/poly *that is exactly learnable in time 2^{cn}, making at most 2^{n-2} equivalence and membership queries. Then* EXP $\not\subseteq C$. *Specifically, if* P/poly *is learnable under these assumptions, then* EXP $\not\subseteq$ P/poly.

We remark that throughout this section, the stronger results where either EXP is replaced by $\mathrm{E} = \mathrm{DTIME}(2^{O(n)})$ or the learning algorithms are allowed to run in $2^{n^{O(1)}}$ time also hold via the same proofs.

References

1. Aizenstein, H., Hegedüs, T., Hellerstein, L., Pitt, L.: Complexity theoretic hardness results for query learning. Computational Complexity 7, 19–53 (1998)
2. Angluin, D.: Queries and concept learning. Machine Learning 2(4), 319–342 (1988)
3. Buhrman, H., Homer, S.: Superpolynomial circuits, almost sparse oracles and the exponential hierarchy. In: Shyamasundar, R.K. (ed.) FSTTCS 1992. LNCS, vol. 652, pp. 116–127. Springer, Heidelberg (1992)
4. Buhrman, H., van Melkebeek, D., Regan, K.W., Sivakumar, D., Strauss, M.: A generalization of resource-bounded measure, with application to the BPP vs. EXP problem. SIAM Journal on Computing 30(2), 576–601 (2001)
5. Fortnow, L., Klivans, A.R.: Efficient learning algorithms yield circuit lower bounds. Journal of Computer and System Sciences 75(1), 27–36 (2009)
6. Hitchcock, J.M.: Online learning and resource-bounded dimension: Winnow yields new lower bounds for hard sets. SIAM Journal on Computing 36(6), 1696–1708 (2007)
7. Karp, R.M., Lipton, R.J.: Some connections between nonuniform and uniform complexity classes. In: Proceedings of the 12th Annual ACM Symposium on Theory of Computing, pp. 302–309 (1980)
8. Kearns, M., Valiant, L.: Cryptographic limitations on learning Boolean formulae and finite automata. J. ACM 41(1), 67–95 (1994)
9. Kharitonov, M.: Cryptographic lower bounds for learnability of Boolean functions on the uniform distribution. J. of Comput. Syst. Sci. 50(3), 600–610 (1995)
10. Lindner, W., Schuler, R., Watanabe, O.: Resource-bounded measure and learnability. Theory of Computing Systems 33(2), 151–170 (2000)
11. Littlestone, N.: Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. Machine Learning 2(4), 285–318 (1988)
12. Lutz, J.H.: Almost everywhere high nonuniform complexity. Journal of Computer and System Sciences 44(2), 220–258 (1992)
13. Merkle, W., Miller, J., Nies, A., Reimann, J., Stephan, F.: Kolmogorov-Loveland Randomness and Stochasticity. Annals of Pure and Applied Logic 138(1-3), 183–210 (2006)
14. Muchnik, A.A., Semenov, A.L., Uspensky, V.A.: Mathematical metaphysics of randomness. Theoretical Computer Science 207(2), 263–317 (1998)
15. Toda, S.: On the computational power of PP and $\oplus$P. SIAM Journal on Computing 20(5), 865–877 (1991)
16. Valiant, L.G.: A theory of the learnable. Communications of the ACM 27(11), 1134–1142 (1984)

Constraint Satisfaction Parameterized by Solution Size

Andrei A. Bulatov[1,⋆] and Dániel Marx[2,⋆⋆]

[1] Simon Fraser University
abulatov@cs.sfu.ca
[2] Humboldt-Universität zu Berlin
dmarx@cs.bme.hu

Abstract. In the constraint satisfaction problem (CSP) corresponding to a constraint language (i.e., a set of relations) Γ, the goal is to find an assignment of values to variables so that a given set of constraints specified by relations from Γ is satisfied. In this paper we study the fixed-parameter tractability of constraint satisfaction problems parameterized by the size of the solution in the following sense: one of the possible values, say 0, is "free," and the number of variables allowed to take other, "expensive," values is restricted. A size constraint requires that exactly k variables take nonzero values. We also study a more refined version of this restriction: a global cardinality constraint prescribes how many variables have to be assigned each particular value. We study the parameterized complexity of these types of CSPs where the parameter is the required number k of nonzero variables. As special cases, we can obtain natural and well-studied parameterized problems such as INDEPENDENT SET, VERTEX COVER, d-HITTING SET, BICLIQUE, etc. In the case of constraint languages closed under substitution of constants, we give a complete characterization of the fixed-parameter tractable cases of CSPs with size constraints, and we show that all the remaining problems are W[1]-hard. For CSPs with cardinality constraints, we obtain a similar classification, but for some of the problems we are only able to show that they are BICLIQUE-hard. The exact parameterized complexity of the BICLIQUE problem is a notorious open problem, although it is believed to be W[1]-hard.

1 Introduction

In a constraint satisfaction problem (CSP) we are given a set of variables, and the goal is to find an assignment of the variables subject to specified constraints. A constraint is usually expressed as a requirement that combinations of values of a certain (usually small) set of variables belong to a certain relation. In the theoretical study of CSPs, one of the key research direction has been the complexity of the CSP when there are restrictions on the type of allowed relations [9,3,2]. This research direction has been started by the seminal Schaefer's Dichotomy Theorem [17], which showed that every Boolean CSP (i.e., CSP with 0-1 variables) restricted in this way is either solvable in polynomial time or is NP-complete. An outstanding open question is the so called *Dichotomy conjecture* of Feder and Vardi [7] which suggests that the dichotomy remains true for

⋆ Research supported by an NSERC Discovery grant.
⋆⋆ Research supported by the Alexander von Humboldt Foundation and OTKA grant 67651.

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 424–436, 2011.

CSPs over any fixed finite domain. The significance of a dichotomy result is that it is very likely to provide a comprehensive understanding of the algorithmic nature of the problem. Indeed, in order to obtain the tractability part of such a conjecture one needs to identify all the algorithmic ideas relevant for the problem.

Parameterized complexity [6,8] investigates the complexity of problems in finer details than classical complexity. Instead of expressing the running time of an algorithm as a function of the input size n only, the running time is expressed as a function of n and a well-defined parameter k of the input instance (such as the size of the solution k we are looking for). For many problems and parameters, there is a polynomial-time algorithm for every fixed value of k, i.e., the problem can be solved in time $n^{f(k)}$. In this case, it makes sense to ask if the combinatorial explosion can be limited to the parameter k by improving the running time to $f(k) \cdot n^{O(1)}$. Problems having algorithms with running time of this form are called *fixed-parameter tractable (FPT)*; it turns out that many well-known NP-hard problems, such as k-VERTEX COVER, k-PATH, and k-DISJOINT TRIANGLES are FPT. On the other hand, the theory of W[1]-hardness suggests that certain problems (e.g., k-CLIQUE, k-DOMINATING SET) are unlikely to be FPT.

The canonical complete problems of the W-hierarchy are (circuit) satisfiability problems where the solution is required to contain exactly k ones. This leads us to the study of Boolean CSP problems with the goal of finding a solution with exactly k ones. The first attempt to study the parameterized complexity of Boolean CSP was made in [14]. If we consider 0 as a "cheap" value available in abundance, while 1 is "costly" and of limited supply then the natural parameter is the number of 1's in a solution. Boolean CSP asking for a solution that assigns exactly k ones is known as the k-ONES problem [5,11]. Clearly, the problem is polynomial-time solvable for every fixed k (by brute force), but it is not at all obvious whether it is FPT. For example, it is possible to express k-VERTEX COVER (which is FPT) and k-INDEPENDENT SET (which is W[1]-hard) as a Boolean CSP. Therefore, characterizing the parameterized complexity of k-ONES requires understanding a class of problems that includes, among many other things, the most basic graph problems. It turned out that the parameterized complexity of the k-ONES problem depends on a new combinatorial property called *weak separability* [14]. Assuming that the constraints are restricted to a finite set Γ of Boolean relations, if every relation in Γ is weakly separable, then the problem is FPT; if Γ contains even one relation violating weak separability, then the problem is W[1]-hard.

There have been further parameterized complexity studies of Boolean CSP [12,18,13], but CSP's with larger domains were not studied. In most cases, we expect that results for larger domains are much more complex than for the Boolean case, and usually require significant new ideas. The goal of the present paper is to generalize the results of [14] to non-Boolean domains. First, we have to define what the proper generalization of k-ONES is if the variables are not Boolean. One natural generalization assumes that there is a distinguished "cheap" value 0 and requires that in a solution there are exactly k nonzero variables. We will call this version of the CSP a *constraint satisfaction problem with size constraints* and denote by OCSP. Another generalization of k-ONES specifies the number $\pi(d)$ of variables assigned each nonzero value d: A mapping $\pi : D \setminus \{0\} \to \mathbb{N}$ is given, and it is required that for each nonzero value d, exactly $\pi(d)$ variables are assigned value d. In the CSP and AI literature, requirements of this

form are called *global cardinality constraints* [1,15]. We will call this problem the *constraint satisfaction problem with cardinality constraints* and denote it by CCSP. In both versions, the parameter is the number of nonzero values required, that is, k for OCSP, and $\sum_{d \in D \setminus \{0\}} \pi(d)$ for CCSP. The usual (non-parametrized) complexity of CCSP over arbitrary domain was characterized in [4]. We investigate both versions; as we shall see, there are unexpected differences between the complexity of the two variants.

A natural minor generalization of CSPs is allowing the use of constants in the input, that is, some variables in the input can be fixed to constant values, or equivalently the constant unary relation $\{(d)\}$ is allowed for every element d of the domain. It is known that the complexity of the decision CSP (corresponding to a 'core' structure) does not change with this generalization [3]. While there is no similar result for the versions of CSPs we study here (and thus this assumption may diminish the generality of our results), this setting is still quite general and at the same time more robust. Many technicalities can be avoided with this formulation. For example, the availability of constants ensures that the decision and search problems are equivalent: by repeatedly substituting constants and solving the decision problem, we can actually find a solution.

Is weak separability the right tractability criterion in the non-Boolean case? It is not difficult to observe that the algorithm of [14] using weak separability generalizes for non-Boolean problems[1]. However, it is not true that only weakly separable relations are tractable. It turns out that there are certain degeneracies and symmetries that allow us to solve the problem even for some relations that are not weakly separable. To understand these degenerate situations, the notion of multivalued morphisms (a generalization of homomorphisms) turns out to be crucial.

Results. For CSP with size constraints, we prove a dichotomy result:

Theorem 1.1. *For every finite Γ closed under substitution of constants,* OCSP(Γ) *is either* FPT *or* W[1]*-hard.*

The precise tractability criterion (which is quite technical) is stated in Section 4. The algorithmic part of Theorem 1.1 consists of a preprocessing to eliminate degeneracies and trivial cases, followed by the use of weak separability. In the hardness part, we take a relation R having a counterexample to weak separability, and use it to show that a known W[1]-hard problem can be simulated with this relation. In the Boolean case [14], this is fairly simple: by identifying coordinates and substituting 0's, we can assume that the relation R is binary, and we need to prove hardness only for two concrete binary relations. For larger domains, however, this approach does not work. We cannot reduce the counterexample to a binary relation by identifying coordinates, thus a complex case analysis would be needed. Fortunately, our hardness proof is more uniform than that. We introduce gadgets that control the values that can appear on the variables. There are certain degenerate cases when these gadgets do not work. However, these degenerate cases can be conveniently described using multivalued morphisms, and these cases turn out to be exactly the cases that we can use in the algorithmic part of the proof.

In the case of cardinality constraints, we face an interesting obstacle. Consider the binary relation R containing only tuples $(0,0)$, $(1,0)$, and $(0,2)$. Given a CSP instance with constraints of this form, finding a solution where the number of 1's is exactly k

[1] In fact, we give an algorithm for non-Boolean domains that is simpler then the one in [14].

and the number of 2's is exactly k is essentially equivalent to finding an independent set of a bipartite graph with k vertices in both classes, or equivalently, a complete bipartite graph (biclique) with $k+k$ vertices. The parameterized complexity of the k-BICLIQUE problem is a longstanding open question (it is conjectured to be W[1]-hard). Thus at this point, it is not possible to give a dichotomy result similar to Theorem 1.1 in the case of cardinality constraints, unless we prove that BICLIQUE is hard:

Theorem 1.2. *For every finite Γ closed under substitution of constants,* CCSP(Γ) *is either* FPT*, or* BICLIQUE*-hard.*

2 Preliminaries

Constraint satisfaction problem with size and cardinality constraints. Let D be a set. We assume that D contains a distinguished element 0. Let D^n denote the set of all n-tuples of elements from D. An n-ary *relation* on D is a subset of D^n, and a *constraint language* Γ is a set of relations on D. In this paper constraint languages are assumed to be finite. We denote by dom(Γ) the set of all values that appear in tuples of the relations in Γ. Given a constraint language Γ, an instance of the *constraint satisfaction problem* (CSP) is a pair $I = (V, \mathcal{C})$, where V is a set of *variables*, and $\mathcal{C}$ is a set of *constraints*. A constraint is a pair $\langle \mathbf{s}, R\rangle$, where R is a (say, n-ary) relation from Γ, and $\mathbf{s}$ is an n-tuple of variables. A *satisfying assignment* of I is a mapping $\tau : V \to D$ such that for every $\langle \mathbf{s}, R\rangle \in \mathcal{C}$ with $\mathbf{s} = (s_1 \dots, s_n)$ the image $\tau(\mathbf{s}) = (\tau(s_1), \dots, \tau(s_n))$ belongs to R. The question in the CSP is whether there is a satisfying assignment for a given instance. The CSP over constraint language Γ is denoted by CSP(Γ).

The *size* of an assignment is the number of variables receiving nonzero values. A *size constraint* is a prescription on the size of the assignment. A *cardinality constraint* for a CSP instance I is a mapping $\pi : D \to \mathbb{N}$ with $\sum_{a\in D} \pi(a) = |V|$. A satisfying assignment τ of I satisfies the cardinality constraint π if the number of variables mapped to each $a \in D$ equals $\pi(a)$. We denote by CCSP(Γ) the variant of CSP(Γ) where the input contains both a cardinality constraint π and the size constraint $k = \sum_{a\in D\setminus\{0\}} \pi(a)$ (this constraint is used a parameter); the question is, given an instance I, an integer k, and a cardinality constraint π, whether there is a satisfying assignment of I of size k and satisfying π. So, instances of OCSP (resp., CCSP) are triples $(V, \mathcal{C}, k)$ (resp., quadruples $(V, \mathcal{C}, k, \pi)$). A *solution* of an instance is a satisfying assignment satisfying the size/cardinality constraints. For both OCSP(Γ) and CCSP(Γ), we are interested in FPT with respect to the parameter k. The INDEPENDENT SET problem is representable as OCSP(R_{IS}), where $R_{IS} = \{(0,0), (0,1), (1,0)\}$. Similarly, the BICLIQUE problem in which given a bipartite graph $G(A, B)$, find two $A' \subseteq A$ and $B' \subseteq B$ with $|A'| = |B'| = t$ and such that every vertex of A' is adjacent with every vertex of B'. This problem is equivalent to CCSP($\{R_{BC}\}$), where R_{BC} is a relation on $\{0, 1, 2\}$ given by $\{(0,0), (1,0), (0,2)\}$.

Closures and 0-validity. A constraint language is called *constant closed* (cc-, for short) if along with every (say, n-ary) relation R, any i, $1 \le i \le n$, and any $d \in D$ the relation obtained by *substitution of constants* $R^{|i;d} = \{(a_1, \dots, a_{i-1}, a_{i+1}, \dots, a_n) \mid (a_1, \dots, a_{i-1}, d, a_{i+1}, \dots, a_n) \in R\}$, also belongs to R. Substitution of constants

$d_1, \dots, d_q$ for coordinate positions $i_1, \dots, i_q$ is defined in a similar way; the resulting relation is denoted by $R^{|i_1,\dots,i_q;d_1,\dots,d_q}$. We call the smallest cc-language containing a constraint language Γ the *cc-closure* of Γ. It is easy to see that the cc-closure of Γ is the set of relations obtained from relations of Γ by substituting constants.

Let f be a satisfying assignment for an instance $I = (V, \mathcal{C}, k)$ of OCSP(Γ) and $S = \{v \mid f(v) \neq 0\}$. We say that the instance $I' = (V', \mathcal{C}', k')$ is *obtained by substituting the nonzero values of f as constants* if I' is constructed as follows: $V' = V \setminus S$, and for each constraint $\langle \mathbf{s}, R\rangle \in \mathcal{C}$ such that $v_{i_1}, \dots v_{i_r}$ are the variables from $\mathbf{s}$ contained in S and $\{v_{j_1}, \dots, v_{j_q}\} = \mathbf{s} \setminus S$, we include in $\mathcal{C}'$ the constraint $\langle (v_{j_1}, \dots, v_{j_q}), R^{|i_1,\dots i_r;f(v_{i_1}),\dots f(v_{i_r})}\rangle$. The size constraint k' is set to k minus the size of f. This operation is defined similarly for a CCSP(Γ) instance $I = (V, \mathcal{C}, k, \pi)$, but in this case the new cardinality constraint π' is given by $\pi'(d) = \pi(d) - |\{v \in V \mid f(v) = d\}|$.

A relation is said to be *0-valid* if the all-zero tuple belongs to the relation. A constraint language Γ is a *cc0-language* if every $R \in \Gamma$ is 0-valid, and if R' is a 0-valid relation obtained from R by substitution of constants, then $R' \in \Gamma$. Observe that if Γ is a cc-language and Γ_0 is the set of 0-valid relations in Γ, then Γ_0 is a cc0-language (but not necessarily a cc-language).

We say that tuple $\mathbf{t}_1 = (a_1, \dots, a_r)$ is an *extension* of tuple $\mathbf{t}_2 = (b_1, \dots, b_r)$ if they are of the same length and for every $1 \leq i \leq r$, $a_i = b_i$ if $b_i \neq 0$. Tuple $\mathbf{t}_2$ is then called a *subset* of $\mathbf{t}_1$. A *minimal satisfying extension* of an assignment f is an extension f' of f (where f, f' are viewed as tuples) such that f' is satisfying, and f has no satisfying extension $f'' \neq f'$ such that f' is an extension of f''.

By repeatedly branching on the unsatisfied constraints, a simple bounded search tree algorithm can enumerate all the minimal satisfying extensions of an assignment.

Lemma 2.1. *Let Γ be a finite constraint language over D. There are functions $d'_\Gamma(k)$ and $e'_\Gamma(k)$ such that for every instance of* CSP(Γ) *with n variables, every assignment f has at most $d'_\Gamma(k)$ minimal satisfying extensions of size at most k and all these minimal extensions can be enumerated in time $e'_\Gamma(k)n^{O(1)}$.*

A consequence of Lemma 2.1 is that, as in [14], CCSP(Γ) and OCSP(Γ) can be reduced to a set of 0-valid instances. We enumerate all the minimal satisfying extensions of size at most k of the all zero assignment (where k is the size constraint) and obtain the 0-valid instances by substituting the nonzero values as constants.

Corollary 2.2. *Let Γ be a cc-language and let $\Gamma_0 \subseteq \Gamma$ be the set of all 0-valued relations from Γ. Then* CCSP(Γ) *is* FPT/W[1]*-hard/*BICLIQUE*-hard if and only if* CCSP(Γ_0) *is. The same holds for* OCSP(Γ) *and* OCSP(Γ_0).

A nonzero satisfying assignment f is said to be a *minimal (nonzero) satisfying assignment* if it is not a proper extension of any nonzero satisfying assignment.

Lemma 2.3. *Let Γ be a finite constraint language. There are functions $d_\Gamma(k)$ and $e_\Gamma(k)$ such that for any instance of* CSP(Γ) *with n variables every variable v is nonzero in at most $d_{\Gamma(k)}$ minimal satisfying assignments of size at most k and all these minimal satisfying assignments can be enumerated in time $e_\Gamma(k)n^{O(1)}$.*

3 Properties of Constraints

By Corollary 2.2, for proving Theorems 1.1 and 1.2 it is sufficient to consider only cc0-languages. Thus in the rest of the paper, we consider only cc0-languages.

3.1 Weak Separability

In the Boolean case, the tractability of 0-valid constraints depends only on weak separability [14]. This is not true exactly this way for larger domains: as we shall see (Theorems 4.1 and 5.1), the complexity characterizations have further conditions.

Tuples $\mathbf{t}_1 = (a_1, \ldots, a_r)$ and $\mathbf{t}_2 = (b_1, \ldots, b_r)$ are *disjoint* if $a_i = 0$ or $b_i = 0$ for every i. The *union* of disjoint tuples $\mathbf{t}_1$ and $\mathbf{t}_2$ is $\mathbf{t}_1 + \mathbf{t}_2 = (c_1, \ldots, c_r)$ where $c_i = a_i$ if $a_i \neq 0$ and $c_i = b_i$ otherwise. If $(a_1, \ldots, a_r)$ is an extension of $(b_1, \ldots, b_r)$, then their *difference* is the tuple $(c_1, \ldots, c_r)$ where $c_i = a_i$ if $b_i = 0$ and $c_i = 0$ otherwise. A tuple $\mathbf{t}$ is *contained* in a set $C \subseteq D$ if every nonzero entry of $\mathbf{t}$ is in C.

A 0-valid relation R is said to be *weakly separable* if the following two conditions hold: (1) For every disjoint tuples $\mathbf{t}_1, \mathbf{t}_2 \in R$, we have $\mathbf{t}_1 + \mathbf{t}_2 \in R$. (2) For every disjoint tuples $\mathbf{t}_1, \mathbf{t}_2$ with $\mathbf{t}_2, \mathbf{t}_1 + \mathbf{t}_2 \in R$, we have $\mathbf{t}_1 \in R$. A constraint language Γ is said to be weakly separable if every relation from Γ is weakly separable. If constraint language Γ is not weakly separable, then we call a triple $(R, \mathbf{t}_1, \mathbf{t}_2)$, $R \in \Gamma$, witnessing that a *union counterexample* if $\mathbf{t}_1, \mathbf{t}_2$ violate condition (1), while if $\mathbf{t}_1, \mathbf{t}_2$ violate condition (2) it is called a *difference counterexample.*

The following combinatorial property is the key for solving weakly separable instances (this property does not necessarily hold for arbitrary relations):

Lemma 3.1. *Let Γ be a weakly separable finite cc0-language over D and I an instance of* CCSP(Γ) *or* OCSP(Γ).

(1) Any satisfying assignment of I is a union of pairwise disjoint minimal ones.
(2) If there is a satisfying assignment f with $f(v) = d$ for some variable v and $d \in D$, then there is a minimal satisfying assignment f' with $f'(v) = d$.

In light of Lemma 3.1(1), it is sufficient to enumerate every minimal assignment of size at most k (using Lemma 2.3) and then to find a disjoint minimal assignments that together satisfy the size/cardinality constraints. As the total size of the assignments we select is at most k and furthermore Lemma 2.3 implies that each variable is nonzero in at most a bounded number of these minimal assignments, the fixed-parameter tractability of finding such disjoint assignments can be shown by standard arguments.

Theorem 3.2. *Let Γ be a finite weakly separable cc0-language over D.*

1. A solution to an instance $(V, \mathcal{C}, k, \pi)$ of CCSP(Γ) *can be found in time $e_\Gamma(k)|V|^{O(1)}$.*
2. A solution to an instance $(V, \mathcal{C}, k)$ of OCSP(Γ) *can be found in time $k^{|D|-1}e_\Gamma(k)|V|^{O(1)}$.*

3.2 Morphisms

Homomorphisms and polymorphisms are standard tools for understanding the complexity of constraints [3,10]. We make use of the notion of multivalued morphisms, a

generalization of homomorphisms, that in a different context has appeared in the literature (see, e.g. [16]) under the guise of hyperoperation. We classify the values into 4 types according to the existence of such morphisms (Definition 3.3). This classification and the observation that these types play an essential role in the way the MVM gadgets (Section 3.4) work are the main technical ideas behind the hardness proofs.

For a subset $0 \in D' \subseteq D$ and an n-ary relation R on D, by $R_{|D'}$ we denote the relation $R \cap (D')^n$. For a language Γ, $\Gamma_{|D'}$ contains every relation $R_{|D'}$ for $R \in \Gamma$.

For a tuple $\mathbf{t} = (a_1, \ldots, a_r) \in \mathrm{dom}(\Gamma)^r$, we denote by $h(\mathbf{t})$ the tuple $(h(a_1), \ldots, h(a_r))$. An *endomorphism* of Γ is a mapping $h : \mathrm{dom}(\Gamma) \to \mathrm{dom}(\Gamma)$ such that $h(0) = 0$ and for every $R \in \Gamma$ and $\mathbf{t} \in R$, the tuple $h(\mathbf{t})$ is also in R. Observe that the requirement $h(0) = 0$ is nonstandard, but it is natural in our setting. The mapping sending all elements of $\mathrm{dom}(\Gamma)$ to 0 is an endomorphism of any 0-valid language. An *inner homomorphism* of Γ from D_1 to D_2 with $0 \in D_1, D_2 \subseteq \mathrm{dom}(\Gamma)$ is a mapping $h : D_1 \to D_2$ such that $h(0) = 0$ and $h(\mathbf{t}) \in R$ holds for any r-ary relation $R \in \Gamma$ and $\mathbf{t} \in D_1^r \cap R$.

A *multivalued morphism* of Γ is a mapping $\phi : \mathrm{dom}(\Gamma) \to 2^{\mathrm{dom}(\Gamma)}$ such that $\phi(0) = \{0\}$ and for every $R \in \Gamma$ and $(a_1, \ldots, a_r) \in R$, we have $\phi(a_1) \times \cdots \times \phi(a_r) \subseteq R$. An *inner multivalued morphism* ϕ from D_1 to D_2 where $0 \in D_1, D_2 \subseteq \mathrm{dom}(\Gamma)$ is defined to be a mapping $\phi : D_1 \to 2^{D_2}$ such that $\phi(0) = \{0\}$ and for every $R \in \Gamma$ and $(a_1, \ldots, a_r) \in R_{|D_1}$, we have $\phi(a_1) \times \cdots \times \phi(a_r) \subseteq R_{|D_2}$.

Observe that if $\phi : \mathrm{dom}(\Gamma) \to 2^{\mathrm{dom}(\Gamma)}$ is a multivalued morphism of a constraint language Γ, and $\phi' : \mathrm{dom}(\Gamma) \to 2^{\mathrm{dom}(\Gamma)}$ is a mapping such that $\phi'(d) \subseteq \phi(d)$ for $d \in \mathrm{dom}(\Gamma)$, then ϕ' is a multivalued morphism. Similar statement holds for inner multivalued morphisms $\psi, \psi' : D_1 \to 2^{D_2}$.

The *product* $g \circ h$ of two endomorphisms or inner homomorphisms is defined by $(g \circ h)(x) = h(g(x))$ for every $x \in D$. If ϕ and ψ are (inner) multivalued morphisms then their product $\phi \circ \psi$ is given by $(\phi \circ \psi)(x) = \bigcup_{y \in \phi(x)} \psi(y)$.

For $x, y \in \mathrm{dom}(\Gamma)$, we say that x *produces* y *in* Γ if Γ has a multivalued morphism ϕ with $\phi(x) = \{0, y\}$ and $\phi(z) = \{0\}$ for every $z \neq x$. Observe that the relation "x produces y" is transitive.

Definition 3.3. *A value* $y \in \mathrm{dom}(\Gamma)$ *is*

1. regular *if there is no multivalued morphism* ϕ *where* $0, y \in \phi(x)$ *for some* $x \in \mathrm{dom}(\Gamma)$*,*
2. semi-regular *if there is a multivalued morphism* ϕ *where* $0, y \in \phi(x)$ *for some* $x \in \mathrm{dom}(\Gamma)$*, but there is no* $x \in \mathrm{dom}(\Gamma)$ *that produces* y*,*
3. self-producing *if* y *produces* y*, and for every* x *that produces* y*,* y *also produces* x*.*
4. degenerate *otherwise.*

It will sometimes be convenient to say that a value y has *type* 1, 2, 3, or 4. We need the following simple properties:

Proposition 3.4. *If there is an endomorphism* h *with* $h(x) = y$*, then the type of* x *cannot be larger than that of* y*.*

Proposition 3.5. *Every degenerate value* y *is produced by a nondegenerate value* x*.*

3.3 Components

The structure of endomorphisms and inner homomorphisms plays an important role in our study. Let Γ be a cc0-language. A *retraction* to $X \subseteq D \setminus \{0\}$ is a mapping ret_X such that $\mathrm{ret}_X(x) = x$ for $x \in X$ and $\mathrm{ret}_X(x) = 0$ otherwise. A nonempty subset $C \subseteq D \setminus \{0\}$ is a *component* of Γ if ret_C is an endomorphism of Γ. A component C is *minimal* if there is no component that is a proper subset of C. If a set C is not a component, then there is a relation $R \in \Gamma$ and $\mathbf{t} \in R$ such that $\mathbf{t}' = \mathrm{ret}_C\mathbf{t} \notin R$. Observe that the intersection of two components is also a component (if it is nonempty). Hence for every nonempty $X \subseteq D \setminus \{0\}$, there is a unique inclusion-wise minimal component that contains X; this component is called the component *generated by* X (or simply the component of X). The importance of components comes from the following result:

Lemma 3.6. *If Γ is not weakly separable, then either*

– there is a union counterexample $(R, \mathbf{t}_1, \mathbf{t}_2)$ such that $\mathbf{t}_1$ (resp., $\mathbf{t}_2$) is contained in a component generated by a value a_1 (resp., a_2), or
– there is a difference counterexample $(R, \mathbf{t}_1, \mathbf{t}_2)$ such that both $\mathbf{t}_1$ and $\mathbf{t}_2$ are contained in a component generated by a value a_1.

3.4 Multivalued Morphism Gadgets

For a relation R and a tuple $\mathbf{t} \in R$, we denote by $\mathsf{supp}(\mathbf{t})$ the set of coordinate positions of $\mathbf{t}$ occupied by nonzero elements. Let $\mathsf{supp}_{\mathbf{t}}(R)$ denote the relation obtained by substituting 0 into all coordinates of R except for $\mathsf{supp}(\mathbf{t})$, i.e. if R is r-ary and $\mathsf{supp}(\mathbf{t}) = \{1, \ldots, r\} \setminus \{i_1, \ldots, i_r\}$ then $\mathsf{supp}_{\mathbf{t}}(R) = R^{|_{i_1,\ldots,i_r;0,\ldots,0}}$.

For a cc0-language Γ and some $0 \in D' \subseteq \mathrm{dom}(\Gamma)$, a *multivalued morphism gadget* $\mathrm{MVM}(\Gamma, D')$ consists of $|D'| - 1$ bags of vertices B_d, $d \in D' \setminus \{0\}$. The number of variables in each bag will be specified every time it is used. The gadget is equipped with the following set of constraints. For every $R \in \Gamma$ and every tuple $\mathbf{t} = (a_1, \ldots, a_r) \in R_{|D'}$ (with, say, $\mathsf{supp}(\mathbf{t}) = \{i_1, \ldots, i_q\}$), we add all possible constraints $\langle \mathbf{s}, \mathsf{supp}_{\mathbf{t}}(R)\rangle$ where $\mathbf{s} = (v_{i_1}, \ldots, v_{i_q})$ such that $v_j \in B_{a_j}$ for every $j \in \{i_1, \ldots, i_q\}$. The *standard assignment* of a gadget assigns a to every variable in bag B_a; observe that it assignment satisfies every constraint of the gadget. We say that bag B_a and the variables in bag B_a *represent* a.

Proposition 3.7. *Let $0 \in D' \subseteq \mathrm{dom}(\Gamma)$. Consider a satisfying assignment f of an $\mathrm{MVM}(\Gamma, D')$ gadget. If $h_f : D' \to 2^{\mathrm{dom}(\Gamma)}$ is a mapping such that $h_f(a)$ is the set of values appearing in bag B_a of the gadget and $h_f(0) = \{0\}$, then h_f is an inner multivalued morphism of Γ from D' to $\mathrm{dom}(\Gamma)$.*

We define gadgets connecting MVM gadgets. The gadget $\mathrm{NAND}(G_1, G_2)$ on $\mathrm{MVM}(\Gamma, D')$ gadgets G_1, G_2 has constraints as follows. For every $R \in \Gamma$ and disjoint tuples $\mathbf{t}_1 = (a_1, \ldots, a_r)$, $\mathbf{t}_2 = (b_1, \ldots, b_r)$ in $R_{|D'}$, we add a constraint $\langle \mathbf{s}, \mathsf{supp}_{\mathbf{t}_1+\mathbf{t}_2} R\rangle$, where $\mathbf{s} = (v_{i_1}, \ldots, v_{i_q})$ with $\{i_1, \ldots i_q\} = \mathsf{supp}(\mathbf{t}_1 + \mathbf{t}_2)$, such that v_j for $j \in \{i_1, \ldots i_q\}$ is in bag B_{a_j} of G_1 if $a_j \neq 0$ and v_j is in bag B_{b_j} of G_2 if $b_j \neq 0$.

If one of G_1, G_2 has the standard assignment and the other is fully zero, then all the constraints in NAND(G_1, G_2) are satisfied. On the other hand, if both G_1 and G_2 have the standard assignment and there is a union counterexample, then NAND(G_1, G_2) is not satisfied. For the reductions, we need this second conclusion not only if both G_1 and G_2 have the standard assignment, but also assignments that "behave well" in some sense. The right notion for our purposes is the following: An inner homomorphism $h : D' \to \mathrm{dom}(\Gamma)$ is $\mathbf{t}$-*recoverable* if Γ has an endomorphism h' such that $(h \circ h')(\mathbf{t}) = \mathbf{t}$.

Lemma 3.8. *Let $0 \in D' \subseteq \mathrm{dom}(\Gamma)$ and let there be a* NAND(G_1, G_2) *gadget on* MVM(Γ, D') *gadgets G_1, G_2.*
(1) If one of G_1 and G_2 has the standard assignment and the other gadget is fully zero, then all constraints of NAND(G_1, G_2) *are satisfied.*
(2) If $\Gamma_{|D'}$ has a union counterexample $(R, \mathbf{t}_1, \mathbf{t}_2)$ and an assignment τ is such that for $i = 1, 2$, τ on G_i is a $\mathbf{t}_i$-recoverable inner homomorphism h_i, then NAND(G_1, G_2) *is not satisfied.*

The IMP(G_1, G_2) gadget is defined similarly, but instead of $\mathbf{t}_1, \mathbf{t}_2 \in R_{|D'}$, we require $\mathbf{t}_2, \mathbf{t}_1 + \mathbf{t}_2 \in R_{|D'}$. An analog of Lemma 3.8 holds for such gadgets.

When the multivalued morphism gadgets are used in the reductions, it will be essential that the bags of the gadgets have very specific sizes. We will ensure somehow that in a solution each bag is either fully zero or fully nonzero. Our aim is to choose the sizes of the bags in such a way that if the sum of the sizes of certain bags add up to a certain integer, then this is only possible if there is exactly one bag of each size.

Fix an integer t and a set $0 \in D' \subseteq D$. It will be convenient to assume that $D' = \{0, 1, \ldots, d\}$. By $\mathcal{Z}^{t,D'}$ we denote the set of integers $Z_{i,j}^{t,D'}$ for $1 \leq i \leq t$ and $1 \leq j \leq d$, given by $Z_{i,j}^{t,D'} := (4td)^{2td+(id+j)} + (4td)^{5td-(id+j)}$.

Lemma 3.9. *Let us fix t and $D' = \{0, 1, \ldots, d\}$. If $A \subseteq \mathcal{Z}^{t,D'}$ and $\mathcal{B}$ is a multiset of values from $\mathcal{Z}^{t,D'}$ with $|\sum_{S \in A} S - \sum_{S \in \mathcal{B}} S| < (4td)^{2td}$, then $\mathcal{B}$ is a set and $\mathcal{B} = A$.*

3.5 Frequent Instances

The following property plays an important role in our algorithms. We say that an instance of CCSP(Γ) or OCSP(Γ), with parameter k is c-*frequent* (for some integer c) if for every $d \in \mathrm{dom}(\Gamma) \setminus \{0\}$ there are at least c variables that take value d in satisfying assignments of size at most k. The algorithm of Lemma 2.1 can be used to decide in fpt-time whether an instance is c-frequent. Lemma 3.10 shows that if an instance is not c-frequent, it can be reduced to c-frequent instances satisfying an additional technical requirement. This is done by eliminating values that appear on less than c variables one by one. A subset $0 \in D' \subseteq \mathrm{dom}(\Gamma)$ is *closed* (with respect to Γ) if Γ has no inner homomorphism from D' that maps some element of D' to an element in $\mathrm{dom}(\Gamma) \setminus D'$.

Lemma 3.10. *Let Γ be a finite cc0-language. Given an instance I of* CCSP(Γ) *or* OCSP(Γ) *with parameter k and an integer c, we can construct in time $f_\Gamma(k, c)n^{O(1)}$ a set of c-frequent instances such that*

1. instance I has a solution iff at least one of the constructed instances has a solution, 2. each instance I_i is an instance of CCSP($\Gamma_{|D_i}$)*, respectively,* OCSP($\Gamma_{|D_i}$)*, for some $D_i \subseteq \mathrm{dom}(\Gamma)$ closed in Γ, and 3. the parameter k_i of I_i is at most k.*

4 Classification for Size Constraints

Unlike in the Boolean case, weak separability of Γ is not equivalent to the tractability of OCSP(Γ): it is possible that Γ is not weakly separable, but OCSP(Γ) is FPT. However, if there is a subset $D' \subseteq \mathrm{dom}(\Gamma)$ of the domain such that $\Gamma_{|D'}$ is not weakly separable and D' has "no special problems" in a certain technical sense, then OCSP(Γ) is W[1]-hard. We need the following definitions. A value $d \in \mathrm{dom}(\Gamma)$ is *weakly separable* if $\Gamma_{|\{0,d\}}$ is weakly separable. A *contraction* of Γ to D' with $0 \in D' \subseteq \mathrm{dom}(\Gamma)$ is an endomorphism $h : \mathrm{dom}(\Gamma) \to D'$ such that $h(d) \neq 0$ for any $d \in \mathrm{dom}(\Gamma) \setminus \{0\}$. Contraction h is *proper* if $D' \neq \mathrm{dom}(\Gamma)$.

The main result for the size constraints CSP is the following dichotomy theorem.

Theorem 4.1. *Let Γ be a finite cc0-language. If there are two sets $\{0\} \subseteq D_2 \subseteq D_1 \subseteq \mathrm{dom}(\Gamma)$ such that (1) D_1 is closed in Γ, (2) $\Gamma_{|D_1}$ has a contraction h to D_2, (3) $\Gamma_{|D_2}$ has no proper contraction, (4) $\Gamma_{|D_1}$ has no weakly separable value that is either degenerate or self-producing, and (5) $\Gamma_{|D_2}$ is not weakly separable, then* OCSP(Γ) *is* W[1]*-hard. If there are no such D_1, D_2, then* OCSP(Γ) *is FPT.*

We present an algorithm solving the FPT cases of the problem and then an important case of the hardness proof, demonstrating the concepts introduced in Section 3.

The algorithm. Let $I = (V, \mathcal{C}, k)$ be an instance of OCSP(Γ). Let us use Lemma 3.10 to obtain instances $I_1, \ldots, I_\ell$ such that I_i is a k-frequent instance of OCSP($\Gamma_{|D^i}$) for some closed set $D^i \subseteq \mathrm{dom}(\Gamma)$. Fix some i and let h be a contraction of $\Gamma_{|D^i}$ such that $|h(D^i)|$ is minimum possible. Set $D_1 := D^i$ and $D_2 := h(D^i)$.

The pair D_1, D_2 violates one of properties (1)–(5) in Theorem 4.1. By the way D_1 and D_2 defined, it is clear that (1) and (2) hold. If the pair violates (3), then let g be a proper contraction of $\Gamma_{|D_2}$. Then $h \circ g$ is a contraction of $\Gamma_{|D_1}$ such that $|g(h(D_1))|$ is strictly less than $|h(D_1)|$, a contradiction. If D_1, D_2 violate (4), then instance I_i always has a solution. Indeed, suppose that $d \in D_1$ is weakly separable and d is produced by $d' \in D_1$ (possibly $d = d'$). Let k_i be the parameter of I_i; then $k_i \leq k$ by Lemma 3.10(4). Since I_i is k-frequent, the set S of variables of I_i where d' can appear in a satisfying assignment of size at most k contains at least k elements. As d' produces d, $\Gamma_{|D_1}$ has a multivalued morphism ϕ such that $\phi(d') = \{0, d\}$ and $\phi(a) = \{0\}$ for $a \in D_1 \setminus \{d'\}$. Therefore, for every $v \in S$, the assignment $\delta_{v,d}$ with $\delta_{v,d}(v) = d$ and 0 everywhere else is a satisfying assignment of I_i. As d is weakly separable in $\Gamma_{|D_1}$, the disjoint union of k_i such assignments $\delta_{v,d}$ is a solution to I_i. Finally, if (5) is violated, then instance I_i of OCSP($\Gamma_{|D_1}$) has a solution if and only if it has a solution restricted to D_2, and the latter can be decided using Lemma 3.2 (as $\Gamma_{|D_2}$ is weakly separable).

Hardness. We say that a set $p_1, \ldots, p_\ell$ of endomorphisms of Γ is a *partition set* if, for every $d \in D' \setminus \{0\}$, $p_i(d) \neq 0$ for exactly one i. The *sum* of the partition set is the mapping h defined such that $h(d)$ is the unique nonzero value in $p_1(d), \ldots, p_\ell(d)$.

The partition set is *good* if the sum of these pairwise disjoint endomorphisms is also an endomorphism; otherwise, the partition set is *bad*.

Lemma 4.2. *If every value is regular in $\Gamma_{|D_2}$, there is no bad partition set in $\Gamma_{|D_2}$, and there is a union counterexample in $\Gamma_{|D_2}$, then* OCSP(Γ) *is* W[1]*-hard.*

Proof. Assume $D_2 = \{0, 1, \ldots, p\}$. The reduction is from MULTICOLORED INDEPENDENT SET, the following W[1]-hard problem: Given a graph G with vertices $v_{i,j}$ ($1 \leq i \leq t$, $1 \leq j \leq n$), find an independent set of size t of the form $\{v_{1,y_1}, \ldots, v_{t,y_t}\}$. For each $v_{i,j}$, we introduce a gadget MVM(Γ, D_2) denoted by $G_{i,j}$. The bag of $G_{i,j}$ corresponding to value $d \in D_2 \setminus \{0\}$ has size $Z_{i,d}^{t,D_2}$. The size constraint is $k := \sum_{i=1}^{t} \sum_{d \in D_2 \setminus \{0\}} Z_{i,d}^{t,D_2}$. If $v_{i,j}$ and $v_{i',j'}$ are adjacent, then we add the gadget NAND($G_{i,j}, G_{i',j'}$). Also, for every $1 \leq i \leq t$, $1 \leq j < j' \leq n$, we add the NAND($G_{i,j}, G_{i,j'}$) gadget.

Suppose that there is a solution C of size exactly t for the MULTICOLORED INDEPENDENT SET instance. If vertex $v_{i,j}$ is in C, then set the standard assignment on gadget $G_{i,j}$, otherwise set the zero assignment. It is clear that this results in an assignment satisfying the size constraint. By Lemma 3.8(1), the constraints of the MVM(Γ, D_2) gadgets as well as the NAND($G_{i,j}, G_{i,j'}$) gadgets are satisfied.

For the other direction, suppose that there is a solution τ satisfying the size constraint. First we observe that τ contains values only from D_1. Indeed, if $c \notin D_1$ appears in bag B_d of a gadget $G_{i,j}$, then τ on $G_{i,j}$ is an inner homomorphism g of Γ from D_2 with $g(d) = c$. Now $h \circ g$ maps a value of D_1 to c, contradicting the assumption that D_1 is a closed set. By applying h on a solution, it can be assumed that only values from D_2 are used. Thus τ on the MVM(Γ, D_2) gadgets provides multivalued morphisms of $\Gamma_{|D_2}$. Since every value is regular in $\Gamma_{|D_2}$, each bag is either fully zero or fully nonzero. The sizes of the nonzero bags add up exactly to the size constraint k. Thus by Lemma 3.9, there is exactly one nonzero bag with size $Z_{i,d}^{t,D_2}$ for every $1 \leq i \leq t$ and $d \in D_2 \setminus \{0\}$.

Take a union counterexample $(R, \mathbf{t}_1, \mathbf{t}_2)$ in $\Gamma_{|D_2}$; by Lemma 3.6, we can assume that $\mathbf{t}_1$, $\mathbf{t}_2$ are in the components of $\Gamma_{|D_2}$ generated by some $a_1, a_2 \in D_2$, respectively. We show that for every $1 \leq i \leq t$, there are values y_i^1 and y_i^2 such that every endomorphism of $\Gamma_{|D_2}$ given by G_{i,y_i^1} (resp., G_{i,y_i^2}) is $\mathbf{t}_1$-recoverable (resp., $\mathbf{t}_2$-recoverable). For a fixed i, let $g_1, \ldots, g_n$ be arbitrary endomorphisms of $\Gamma_{|D_2}$ given by $G_{i,1}, \ldots, G_{i,n}$, respectively. Since the sizes of nonzero bags are all different, these endomorphisms are pairwise disjoint and they form a partition set. As there is no bad partition set in $\Gamma_{|D_2}$, their sum g is an endomorphism of $\Gamma_{|D_2}$. Since $\Gamma_{|D_2}$ has no proper contractions, g has to be a permutation and hence g^s is the identity for some $s \geq 1$. There is a unique $1 \leq y_i^1 \leq n$ such that $g_{y_i^1}(a_1) \neq 0$. The homomorphism $g_{y_i^1} \circ g^{s-1}$ maps every $a \in D_2$ either to 0 or a; i.e., $g_{y_i^1} \circ g^{s-1} = \mathrm{ret}_S$ for some set $S \subseteq D_2$ containing a_1. Hence S is a component containing a_1 and S contains every value of $\mathbf{t}_1$. It follows that $g_{y_i^1}$ given by $G_{y_i^1}$ is $\mathbf{t}_1$-recoverable. A similar argument works for y_j^2, thus the required values y_i^1, y_i^2 exist. Let us observe that it is not possible that $y_i^1 \neq y_i^2$: by Lemma 3.8(2) the constraints of NAND(G_{i,y_i^1}, G_{i,y_i^2}) are not satisfied in this case. Let C contain vertex $v_{i,j}$ if $j = y_i^1 = y_i^2$. It follows that C is a multicolored independent set: if vertices $v_{i,j}$, $v_{i',j'}$ are adjacent, then some constraint of NAND($G_{i,j}, G_{i',j'}$)=NAND($G_{i,y_j^1}, G_{i',y_j^2}$) is not satisfied. □

5 Classification for Cardinality Constraints

The characterization of the complexity of CCSP(Γ) requires a new definition, which was not relevant for OCSP(Γ). The *core* of Γ is the component generated by the set of all nondegenerate values in dom(Γ). We say that Γ is a core if the core of Γ is dom(Γ).

Theorem 5.1. *Let Γ be a cc0-language. If there is a $0 \in D' \subseteq \mathrm{dom}(\Gamma)$ s.t. $\Gamma_{|D'}$ is a core and not weakly separable, then* CCSP(Γ) *is* BICLIQUE*-hard, and FPT otherwise.*

A significant difference between the hardness proofs of OCSP(Γ) and CCSP(Γ) is that in OCSP(Γ), we can assume that no proper contraction exists and this can be used to show that certain endomorphisms have to be permutations (see Lemma 4.2). For CCSP(Γ), we cannot make this assumption, thus we need a delicate argument, making use of the cardinality constraint, to achieve a similar effect.

References

1. Bessière, C., Hebrard, E., Hnich, B., Walsh, T.: The complexity of global constraints. In: Wallace, M. (ed.) AAAI. LNCS, vol. 3258, pp. 112–117. Springer, Heidelberg (2004)
2. Bulatov, A.: Tractable conservative constraint satisfaction problems. In: LICS, pp. 321–330. IEEE Computer Society, Los Alamitos (2003)
3. Bulatov, A.A., Jeavons, P., Krokhin, A.A.: Classifying the complexity of constraints using finite algebras. SIAM J. Comput. 34(3), 720–742 (2005)
4. Bulatov, A.A., Marx, D.: The complexity of global cardinality constraints. In: LICS, pp. 419–428. IEEE Computer Society, Los Alamitos (2009)
5. Creignou, N., Schnoor, H., Schnoor, I.: Non-uniform boolean constraint satisfaction problems with cardinality constraint. In: Kaminski, M., Martini, S. (eds.) CSL 2008. LNCS, vol. 5213, pp. 109–123. Springer, Heidelberg (2008)
6. Downey, R.G., Fellows, M.R.: Parameterized Complexity (1999)
7. Feder, T., Vardi, M.Y.: Monotone monadic snp and constraint satisfaction. In: STOC, pp. 612–622 (1993)
8. Flum, J., Grohe, M.: Parameterized Complexity Theory. Springer, Berlin (2006)
9. Jeavons, P., Cohen, D., Gyssens, M.: Closure properties of constraints. J. ACM 44, 527–548 (1997)
10. Jeavons, P., Cohen, D., Gyssens, M.: How to determine the expressive power of constraints. Constraints 4, 113–131 (1999)
11. Khanna, S., Sudan, M., Trevisan, L., Williamson, D.P.: The approximability of constraint satisfaction problems. SIAM J. Comput. 30(6), 1863–1920 (2001)
12. Kratsch, S., Wahlström, M.: Preprocessing of min ones problems: A dichotomy. In: Abramsky, S., Gavoille, C., Kirchner, C., Meyer auf der Heide, F., Spirakis, P.G. (eds.) ICALP 2010. LNCS, vol. 6198, pp. 653–665. Springer, Heidelberg (2010)
13. Krokhin, A.A., Marx, D.: On the hardness of losing weight. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part I. LNCS, vol. 5125, pp. 662–673. Springer, Heidelberg (2008)
14. Marx, D.: Parameterized complexity of constraint satisfaction problems. Computational Complexity 14

15. Régin, J.C., Gomes, C.P.: The cardinality matrix constraint. In: Wallace, M. (ed.) CP 2004. LNCS, vol. 3258, pp. 572–587. Springer, Heidelberg (2004)
16. Rosenberg, I.: Multiple-valued hyperstructures. In: ISMVL, pp. 326–333 (1998)
17. Schaefer, T.J.: The complexity of satisfiability problems. In: STOC, pp. 216–226 (1978)
18. Szeider, S.: The parameterized complexity of k-flip local search for sat and max sat. In: Kullmann, O. (ed.) SAT 2009. LNCS, vol. 5584, pp. 276–283. Springer, Heidelberg (2009)

Preprocessing for Treewidth: A Combinatorial Analysis through Kernelization*

Hans L. Bodlaender, Bart M.P. Jansen, and Stefan Kratsch

Utrecht University, The Netherlands
{hansb,bart,kratsch}@cs.uu.nl

Abstract. Using the framework of kernelization we study whether efficient preprocessing schemes for the Treewidth problem can give provable bounds on the size of the processed instances. Assuming the AND-distillation conjecture to hold, the standard parameterization of Treewidth does not have a kernel of polynomial size and thus instances (G, k) of the decision problem of Treewidth cannot be efficiently reduced to equivalent instances of size polynomial in k. In this paper, we consider different parameterizations of Treewidth. We show that Treewidth has a kernel with $\mathcal{O}(\ell^3)$ vertices, where ℓ denotes the size of a vertex cover, and a kernel with $\mathcal{O}(\ell^4)$ vertices, where ℓ denotes the size of a feedback vertex set. This implies that given an instance (G, k) of Treewidth we can efficiently reduce its size to $\mathcal{O}((\ell^*)^4)$ vertices, where ℓ^* is the size of a minimum feedback vertex set in G. In contrast, we show that Treewidth parameterized by the vertex-deletion distance to a co-cluster graph and Weighted Treewidth parameterized by the size of a vertex cover do not have polynomial kernels unless NP $\subseteq$ coNP/poly. Treewidth parameterized by the target value plus the deletion distance to a cluster graph has no polynomial kernel unless the AND-distillation conjecture does not hold.

1 Introduction

Treewidth is a well-studied graph parameter, with many theoretical and practical applications. A related parameter is *Weighted Treewidth*, where vertices have weights, and the width of a tree decomposition is the maximum over all bags of the sum of the weights of the vertices in a bag minus one. In this work we study the decision problems related to these width parameters, which given a graph G and integer k ask whether the (weighted) treewidth of G is at most k. For precise definitions, see Section 2.

Preprocessing heuristics for Treewidth and Weighted Treewidth have been studied in a practical setting [8,9,15]. The experimental results reported in these papers show that there are preprocessing heuristics that give significant reductions in size for many practical instances, making it more feasible to compute, exactly or approximately, the treewidth of those graphs. However, these

* This work was supported by the Netherlands Organization for Scientific Research (N.W.O.), project "KERNELS: Combinatorial Analysis of Data Reduction".

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 437–448, 2011.

heuristics do not give any guarantees on the effectiveness of the preprocessing: there is no provable bound on the size of the processed instances. The purpose of this work is to give a theoretical analysis of the potential of preprocessing for Treewidth, studying whether there are efficient preprocessing procedures whose effectiveness can be proven, and what the resulting size bounds look like. Such investigations are made possible using the concept of *kernelization* [16], which is a relatively young subfield of algorithm design and analysis. A *kernelization algorithm* (or *kernel*) is a polynomial-time algorithm which given an instance $(x, k) \in \Sigma^* \times \mathbb{N}$ of some parameterized problem, computes an equivalent instance (x', k') whose size is bounded by a function $f(k)$ depending only on the chosen parameter, i.e., $|x'|, k' \leq f(k)$. The function f is the *size* of the kernel, and *polynomial kernels* ($f \in k^{\mathcal{O}(1)}$) are of particular interest.

From a theoretical point of view, the fact that Treewidth belongs to FPT (see for instance [3,17]), implies that there is a kernel for the problem. However, the size of such a kernel depends on the function of the parameter in the running time of the FPT algorithm; with the current state of FPT algorithms for Treewidth this size would be exponential in k^3 (where k is the target treewidth). Bodlaender et al. [5] have shown that Treewidth with standard parameterization (i.e., parameterized by k) has no polynomial kernel unless all coNP-complete problems have distillation algorithms; hence it is unlikely that there is a polynomial-time algorithm that reduces the size of an instance (G, k) of Treewidth to a polynomial in the desired treewidth k. We therefore turn to other parameters (e.g., the vertex cover number of the input graph), and determine whether we can efficiently shrink an input of Treewidth to a size which is polynomial in such a parameter. We consider different structural parameters of the input graph: these parameters measure the number of vertex deletions needed to transform the input into a graph of some very simple graph class. All parameterized problems we consider fit the following template, where $\mathcal{F}$ is a class of graphs:

> Treewidth parameterized by a modulator to $\mathcal{F}$
> **Instance:** A graph $G = (V, E)$, a positive integer k, and a set $S \subseteq V$ such that $G - S \in \mathcal{F}$.
> **Parameter:** $\ell := |S|$.
> **Question:** $\mathbf{tw}(G) \leq k$?

The set S is a *modulator* to the class $\mathcal{F}$.

Our work. In this paper, we add positive theoretical results to the positive experimental work. Our theoretical results can possibly be of practical value, but an experimental evaluation has not yet been undertaken. We first take as parameter the size of a vertex cover of G, resulting in the problem Treewidth parameterized by a vertex cover (which fits into the given template when using $\mathcal{F}$ as the class of edgeless graphs). We prove that this problem admits a polynomial kernel with $\mathcal{O}(\ell^3)$ vertices. Since we can first compute a 2-approximation for the minimum vertex cover and then feed this to our kernelization algorithm, this implies that an instance (G, k) of Treewidth on a graph with a minimum vertex

cover of size ℓ^* can be shrunk in polynomial-time into an instance with $\mathcal{O}((\ell^*)^3)$ vertices, even if we are not given a minimum vertex cover in the input.

We then turn to the parameter "feedback vertex number", which is easily seen to be at most the value of the vertex cover number. We extend our positive results by showing that Treewidth parameterized by a feedback vertex set (which fits the template when $\mathcal{F}$ is the class of forests) admits a kernel with $\mathcal{O}(\ell^4)$ vertices. By using a polynomial-time 2-approximation algorithm for Feedback Vertex Set [2], we can again drop the assumption that such a set is supplied in the input.

After these two examples it becomes an interesting question whether there is a parameter even smaller than the feedback vertex number in which the size of an instance can be bounded efficiently. Since Treewidth is trivially solvable on chordal graphs, and the deletion distance to a chordal graph is at most the feedback vertex number, one might hope that Treewidth parameterized by a modulator to chordal graphs admits a polynomial kernel. This appears to be very unlikely. Assuming the AND-distillation conjecture [5] we prove the stronger statement that even when using the compound parameter "target treewidth plus the size of a given modulator to cluster graphs", Treewidth does not admit a polynomial kernel - recall that a cluster graph is a disjoint union of cliques. We also prove that Treewidth parameterized by a modulator to co-cluster graphs (the edge-complements of cluster graphs) does not admit a polynomial kernel unless NP $\subseteq$ coNP/poly, and use this result to show that, under the same assumption, the Weighted Treewidth problem does not even admit a polynomial kernel when parameterized by the size of a vertex cover. It is interesting to note the difference between Treewidth and Weighted Treewidth when parameterized by vertex cover.

Organization of the paper. After this introduction, we give preliminary definitions and results in Section 2. In Section 3, we show that Treewidth parameterized by a vertex cover has a kernel with $\mathcal{O}(\ell^3)$ vertices. To do so, we introduce a number of 'safe' reduction rules, that are variants of rules from existing treewidth algorithms and preprocessing methods, including rules that remove simplicial vertices. In Section 4, we turn to Treewidth parameterized by a feedback vertex set (of size ℓ) and show a kernel with $\mathcal{O}(\ell^4)$ vertices. A key role, in addition to variants of the rules for vertex cover, will be played by *almost simplicial vertices* and we will give a set of safe reduction rules that remove all those vertices. In Section 5 we present our lower bound results for Treewidth parameterized by distance from cluster respectively co-cluster graphs as well as for Weighted Treewidth parameterized by a vertex cover; we build upon the recent framework of Bodlaender et al. [5] as well as the notion of cross-composition [6]. Some final remarks are made in Section 6.

2 Preliminaries

In this work all graphs are finite, simple, and undirected. The open neighborhood of a vertex $v \in V$ in a graph G is denoted by $N_G(v)$, and its closed neighborhood

is $N_G[v]$. If $S \subseteq V$ is a vertex set then $G - S$ denotes the graph obtained from G by deleting all vertices of S and their incident edges. A vertex v is *simplicial* in a graph G if $N_G(v)$ is a clique. A vertex $v \in V$ is *almost simplicial* in a graph G if v has a neighbor w such that $N_G(v) - \{w\}$ is a clique. In such a case, we call w the *special neighbor* of v.

A *tree decomposition* of a graph $G = (V, E)$ is a pair $(\{X_i \mid i \in I\}, T = (I, F))$ with $\{X_i \mid i \in I\}$ a family of subsets of V, and T a tree on edge set F, such that

- $\bigcup_{i \in I} X_i = V$.
- For all $\{v, w\} \in E$, there is an $i \in I$ with $v, w \in X_i$.
- For all $v \in V$, the set $I_v = \{i \in I \mid v \in X_i\}$ induces a subtree of T.

The sets X_i are called the *bags* of the tree decomposition. The *width* of a tree decomposition $(\{X_i \mid i \in I\}, T = (I, F))$ is $\max_{i \in I} |X_i| - 1$, and the *treewidth* of G is the minimum width of a tree decomposition of G.

Suppose we have a graph $G = (V, E)$ with a weight function $w : V \to \mathbb{N}$. The *weighted width* of a tree decomposition $(\{X_i \mid i \in I\}, T = (I, F))$ of G equals $\max_{i \in I} \sum_{v \in X_i} w(i) - 1$, and the *weighted treewidth* of G is the minimum weighted width of a tree decomposition of G.

A graph $H = (W, F)$ is a *minor* of a graph $G = (V, E)$, if H can be obtained from G by a series of zero or more vertex deletions, edge deletions, and/or edge contractions. An edge contraction is the operation where two adjacent vertices v, w are replaced by one vertex with neighborhood $(N(v) \cup N(w)) \setminus \{v, w\}$. The following proposition is well known.

Proposition 1. *Let H be a minor of G. Then the treewidth of H is at most the treewidth of G.*

The kernelization algorithms we present consist of a number of reduction rules. In each case, the input to the rule is a graph $G = (V, E)$, an integer k, and a deletion set $S \subseteq V$ such that $G - S$ is a member of the relevant graph class $\mathcal{F}$, and the output is an instance $(G' = (V', E'), k', S')$. A rule is said to be *safe* if for all inputs (G, k, S) which satisfy $G - S \in \mathcal{F}$ we have $\mathbf{tw}(G) \leq k \Leftrightarrow \mathbf{tw}(G') \leq k'$ and $G' - S' \in \mathcal{F}$. We will sometimes say that the algorithm answers **yes** or **no**; this should be interpreted as outputting a constant-size **yes** or **no** instance of the problem at hand, i.e., a clique on three vertices with $k = 2$, respectively the same clique with $k = 1$.

Several proofs had to be omitted from this extended abstract due to space restrictions, and can be found in the full version of this work [7].

3 Kernelization with Respect to Vertex Cover Number: Eliminating Simplicial Vertices

In this section we show our kernelization for Treewidth parameterized by a vertex cover (i.e., parameterized by a modulator to an independent set). The kernelization focuses mostly on simplicial vertices; removing them (and possibly

updating a bound for the treewidth) is a well known and often used preprocessing rule for TREEWIDTH; see the discussion in [9]. Another rule, first used in the linear time algorithm for bounded treewidth in [3] adds edges between vertices with many common neighbors. The rule was also used in lower bound heuristics for treewidth, see [13,14], see also [10,4].

Rule 1 (Low degree simplicial vertex). *If v is a simplicial vertex of degree at most k then remove v.*

Rule 2 (High degree simplicial vertex). *If v is a simplicial vertex of degree greater than k then answer* **no**.

Standard theory on treewidth shows that Rules 1 and 2 are safe. It is well known [3] that if non-adjacent vertices v, w have at least $k+1$ common neighbors, then adding the edge $\{v, w\}$ does not affect[1] whether the treewidth of the graph is at most k. We use this rule in a restricted setting, to ensure that S remains a vertex cover of the graph.

Rule 3 (Common neighbors improvement). *Suppose that $\{v, w\} \notin E$ and that $v \in S$ or $w \in S$. If v and w have at least $k+1$ common neighbors, then add the edge $\{v, w\}$.*

Yet another simple rule is the following, using that S is a vertex cover of G.

Rule 4 (Trivial decision). *If $k \geq |S|$, then answer* **yes**.

Safeness can be argued as follows. The treewidth of G is at most $|S|$: for each $v \in V - S$, take a bag with vertex set $S \cup \{v\}$, and connect these bags in any way. This gives a tree decomposition of G of width at most $|S|$.

It is not hard to argue the, possibly surprising, fact that the exhaustive application of Rules 1–4 (i.e., until we answer **no** or **yes** or no application of one of these rules is possible) already gives a polynomial kernel for TREEWIDTH PARAMETERIZED BY A VERTEX COVER. It is clear that this reduction can be performed in polynomial time (it is easy to do it in time $\mathcal{O}(|V| \cdot |E|)$).

Theorem 1. TREEWIDTH PARAMETERIZED BY A VERTEX COVER *has a kernel with $\mathcal{O}(\ell^3)$ vertices.*

Proof. Let (G, k, S) be an instance of TREEWIDTH PARAMETERIZED BY A VERTEX COVER. Let (G', k', S') be the instance obtained from exhaustive application of Rules 1–4. By safety of the reduction rules (G', k', S') is **yes** if and only if (G, k, S) is **yes**.

The reduction rules guarantee that $S' \subseteq S$ is a vertex cover in G', with $|S'| \leq \ell$. Each vertex $v \in V' - S'$ has at least one pair of distinct neighbors in S' that are not adjacent, otherwise v is simplicial and would have been handled by Rule 1 or

[1] In any triangulation, either $\{v, w\}$ is an edge, or all common neighbors of v and w form a clique. In the latter case, this clique plus v is a clique with at least $k+2$ vertices, implying a tree decomposition of width at least $k+1$.

Rule 2. Assign v to this pair. If we assign v to the pair $\{w, x\}$, then v is a common neighbor of w and x. Hence a pair cannot have more than k vertices assigned to it, otherwise Rule 3 applies. As there are at most $\ell \cdot (\ell - 1)/2$ pairs of non-adjacent neighbors in S', we have $|V' - S'| \leq k \cdot \ell \cdot (\ell - 1)/2 \leq \ell^2 \cdot (\ell - 1)/2 \in \mathcal{O}(\ell^3)$. □

By combining Theorem 1 with a polynomial-time 2-approximation algorithm for vertex cover, we obtain the following corollary.

Corollary 1. *There is a polynomial-time algorithm that given an instance* $(G = (V, E), k)$ *of* Treewidth *computes an equivalent instance* $(G' = (V', E'), k)$ *such that* $V' \subseteq V$ *and* $|V'| \in \mathcal{O}((\ell^*)^3)$*, where* ℓ^* *is the size of a minimum vertex cover of* G*.*

4 A Polynomial Kernel for Treewidth Parameterized by Feedback Vertex Set

In this section, we give the proof that Treewidth parameterized by a feedback vertex set (of size ℓ) has a kernel with $\mathcal{O}(\ell^4)$ vertices. The kernelization algorithm is again given by a set of safe reduction rules, that are applied while possible: two simple rules, three rules that remove all almost simplicial vertices (Section 4.1), and four rules that reduce the graph when there is a clique-seeing path (defined in Section 4.2). In Section 4.3, we show that graphs to which no rule applies have $\mathcal{O}(\ell^4)$ vertices, and thus arrive at our kernel bound.

The first new rule generalizes Rule 3; it was used in experiments [13] and its correctness is proven in [4]. The rule can be implemented in polynomial time by using a maximum flow algorithm to find the disjoint paths.

Rule 5 (Disjoint paths improvement). *Suppose* $\{v, w\} \notin E$ *and that* $v \in S$ *or* $w \in S$*. If there are at least* $k + 1$ *internally vertex-disjoint paths between* v *and* w*, then add the edge* $\{v, w\}$*.*

The second new rule is straight forward, and is correct because reasoning similar to that of Rule 4 shows that each graph with a feedback vertex set of size ℓ has treewidth at most $\ell + 1$.

Rule 6 (Trivial decision). *If* $k \geq |S| + 1$*, then answer* **yes**.

4.1 Almost Simplicial Vertices

In [9], the notion of *almost simplicial vertex* was introduced, and a reduction rule was given that removed almost simplicial vertices whose degree was at most a known lower bound for the treewidth of the input graph. In this section, we give a set of rules that also remove almost simplicial vertices of higher degree. Rule 7 is a reformulation the *Low Degree Almost Simplicial Vertex Rule* from [9]. Rule 8 gives a simple way to deal with almost simplicial vertices of degree larger than $k + 1$. Its correctness is obvious: v with its neighbors except its special neighbor forms a clique with at least $k + 2$ vertices, so the treewidth is larger than k.

Rule 7 (Low Degree Almost Simplicial Vertex). *Let v be an almost simplicial vertex with special neighbor w. If the degree of v is at most k, then contract the edge $\{v, w\}$ into w obtaining G'. If $v \in S$, then let $S' := S \setminus \{v\} \cup \{w\}$, else let $S' := S$.*

Lemma 1. *Rule 7 is safe.*

Proof. It is clear that S' is a feedback vertex set of G'.

Let G' be the graph resulting after the operation. If the treewidth of G is at most k, then the treewidth of G' is at most k as the treewidth cannot increase by contraction (Proposition 1).

Suppose the treewidth of G' is at most k. Take a tree decomposition of G' of width at most k. It is well-known that because $N_G(v)$ is a clique in G', there must be a bag that contains all vertices of $N_G(v)$, say $N_G(v) \subseteq X_i$. Add a new bag with vertex set $N_G[v]$ and make it adjacent in the tree decomposition to node i; we obtain a tree decomposition of G of width at most k. □

Rule 8 (High Degree Almost Simplicial Vertex). *Let v be an almost simplicial vertex. If the degree of v is at least $k + 2$, then answer* **no**.

We introduce a new, more complex rule that deals with almost simplicial vertices of degree exactly $k + 1$. The correctness proof had to be omitted [7].

Rule 9 (Degree $k+1$ Almost Simplicial Vertex). *Let v be an almost simplicial vertex with special neighbor w, and let the degree of v be exactly $k + 1$.*

- *If for each vertex $x \in N_G(v) - \{w\}$, there is an edge $\{x, w\} \in E$ or a path in G from x to w that avoids $N_G[v] - \{x, w\}$, answer* **no**.
- *Otherwise, contract the edge $\{v, w\}$ to a new vertex x, obtaining G'. If $v \in S$ or $w \in S$, then let $S' := S \setminus \{v, w\} \cup \{x\}$, else let $S' := S$.*

Note that the rules for almost simplicial vertices (Rules 7, 8, and 9) can be easily seen to subsume the rules for simplicial vertices (Rules 1 and 2) used in the previous section (the first case of Rule 9 covers simplicial vertices of degree $k + 1$).

The following proposition follows from counting arguments similar to those in the proof of Theorem 1, by observing that after exhaustive application of the rules no leaf in the forest $G - S$ is almost simplicial, and hence every such leaf must have a pair of neighbors in S which are non-adjacent.

Proposition 2. *Suppose an instance (G, k) of* Treewidth *is given together with a feedback vertex set S of size ℓ. If we exhaustively apply Rules 5 – 9, then we obtain in polynomial time an equivalent instance (G', k) with a feedback vertex set S' of size at most ℓ, such that the forest $G' - S'$ has $\mathcal{O}(\ell^3)$ leaves.*

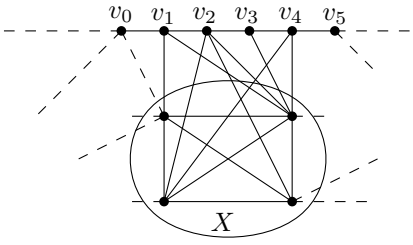


Fig. 1. An example of a clique-seeing path

Preprocessing heuristics. At this point, we want to make a sidestep. In [9], the *Almost Simplicial Vertex Rule* was given as a *preprocessing heuristic* for the optimization version of TREEWIDTH. There, a variable LOW was invariantly kept as lower bound on the treewidth of the original input graph; the rule could be applied to almost simplicial vertices of degree d whenever d was at most this value LOW. Thus, in the work of [9], almost simplicial vertices of degree at least LOW $+1$ could not be dealt with. Using a variant of Rule 9, we can now extend the *Almost Simplicial Vertex Rule* and preprocess a graph such that we remove all almost simplicial vertices, as follows. Suppose v is an almost simplicial vertex of degree d with special neighbor w. If $d - 1 >$ LOW, then LOW is set to $d - 1$. Then, if $d - 1 =$ LOW, we check if for each vertex $x \in N_G(v) - \{w\}$, there is an edge $\{x, w\} \in E$ or a path from x to w that does not use any vertex in $N_G[v] - \{x, w\}$. If so, LOW is increased by one. Now, we contract the edge $\{x, w\}$. This rule is *safe* in the sense that the treewidth of the original graph G equals the maximum of LOW and the treewidth of the reduced graph.

An experimental evaluation of this 'extended simplicial vertex rule', similar as the work in [9] has not yet been undertaken.

4.2 Clique-Seeing Paths

We call a path $(v_0, v_1, \ldots, v_r, v_{r+1})$ a *clique-seeing path* that *sees* clique X, if

- $X = \bigcup_{i=1}^{r} N_G(v_i) \setminus \{v_0, v_1, \ldots, v_{r+1}\}$ is a clique.
- For each each i, $1 \leq i \leq r$, $N(v_i) \subseteq \{v_{i-1}, v_{i+1}\} \cup X$.

An example is given in Fig. 1. Note that v_0 and v_{r+1} play a special role. Each v_i, $1 \leq i \leq r$, has exactly two neighbors outside X, namely the previous and next vertex on the path (v_{i-1} and v_{i+1}). In our analysis it is sufficient when we look at clique-seeing paths with all vertices on the path in the forest $V - S$, and all vertices in the seen clique in the feedback vertex set S, but the rules are also safe in other cases.

We present four reduction rules. The first rule deals with clique-seeing paths that see a clique of size at most $k - 2$. The second and third consider the case that $\{v_1, \ldots, v_r\} \cup X$ separate v_0 and v_{r+1} in the graph. The fourth decides **no** if the path has at least $6k + 6$ inner vertices and no other rule applies. All proofs are deferred to the full version [7].

Rule 10. *Suppose we have a clique-seeing path* $(v_0, v_1, \ldots, v_r, v_{r+1})$*, that sees a clique* X *with* $|X| \leq k-2$*. If*

$$N(v_r) \cap X \subseteq \bigcup_{1 \leq i \leq r-1} N(v_i) \cap X$$

then contract the edge $\{v_r, v_{r+1}\}$ *into the vertex* v_{r+1}*, obtaining* G'*. If* $v_r \in S$*, then let* $S' := S \setminus \{v_r\} \cup \{v_{r+1}\}$*, else let* $S' := S$*.*

The next rule is based upon the notion of *minimal almost clique separators* from [8]. A set of vertices Q *separates* vertices v and w if each path from v to w uses at least one vertex in Q. A set of vertices is a *separator* if there exist vertices v and w such that Q separates v from w. Q *minimally separates* v and w if it separates v and w but there is no proper subset of Q that separates v and w. Q is a *minimal separator* if there is a pair of vertices v and w that are minimally separated by Q. A set of vertices Q is a *minimal almost clique separator*, if it is a minimal separator and there is a vertex $v \in Q$ such that $Q - \{v\}$ is a clique. Bodlaender and Koster [8] have shown that the treewidth is not changed when edges are added to make a minimal almost clique separator into a clique. We use a version of this rule that ensures that S still is a feedback vertex set. Safeness of the following rule thus follows directly from the analysis in [8]. The discussion in [8] also shows that it can be tested in polynomial time.

Rule 11. *Let* Q *be a minimal almost clique separator in* G*, and suppose there is at most one vertex* $v \in Q$ *with* $v \notin S$*. Then, add an edge between each pair of non-adjacent vertices in* Q*.*

Exhaustive application of the following rule will provide us with a useful connectivity property that allows a rejection of instances with clique-seeing paths which remain long after application of Rules 5 to 12.

Rule 12. *Suppose* $(v_0, v_1, v_2, v_3, v_4)$ *is a clique-seeing path in* $V \setminus S$ *that sees clique* $X \subseteq S$*. Suppose* $\{v_1, v_2, v_3\} \cup X$ *separates* v_0 *from* v_4 *and suppose that Rule 11 cannot be applied. Compute the treewidth of* $G[\{v_1, v_2, v_3\} \cup X]$*. If it is larger than* k*, then answer* **no***, otherwise remove* v_2 *from* G*.*

Polynomial-time computability of the treewidth of $G[\{v_1, v_2, v_3\} \cup X]$ follows from a result in [12]. A detailed counting argument shows safeness of our last rule.

Rule 13. *Suppose we have a clique-seeing path* $(v_0, \ldots, v_{r+1})$ *with* $r \geq 6k+6$*, and suppose Rules 5–12 are not applicable. Then answer* **no***.*

4.3 The Kernelization

We show that exhaustive application of Rules 5 through 13 gives the following kernelization result.

Theorem 2. Treewidth parameterized by a feedback vertex set *has a kernel with* $\mathcal{O}(\ell^4)$ *vertices.*

Proof. Let (G, k, S) be an instance of Treewidth parameterized by a feedback vertex set and let (G', k, S') be obtained from exhaustive application of Rules 5 through 13. Rules 12 and 13 are only tested for paths in $G - S$; as that is a forest, we need to test at most $\mathcal{O}(n^2)$ paths, and thus the test has to be done a polynomial number of times. It was showed that all rules are safe and that they can be performed exhaustively in polynomial time, so the instances are equivalent, S' is a feedback vertex set of G', and $|S'| \leq |S|$.

Let us analyze the size of G'. The forest $G' - S'$ has $\mathcal{O}(\ell^3)$ leaves (Proposition 2), and hence $\mathcal{O}(\ell^3)$ vertices of degree at least three. There are $\mathcal{O}(\ell^3)$ paths in the forest connecting leaves and vertices of degree at least three. Each path of length at least $6k + 8$, i.e., with at least $6k + 6$ internal vertices is not clique-seeing by Rule 13. Thus, for the analysis, we split the paths into parts of size $6k + 8$ which are therefore not clique-seeing. At most $6k + 7$ vertices per path will not belong to such a part, but these are at most $\mathcal{O}(\ell^3 \cdot k)$ in total. We assign each part to a pair of non-adjacent vertices in S which are adjacent to internal vertices of the part. We can assign at most k parts to a pair $\{u, v\}$: indeed, since the parts are disjoint they would otherwise give rise to more than k disjoint $u - v$ paths, contradicting (G', k, S') being reduced under Rule 5. Thus, we have $\mathcal{O}(k(6k + 8)\ell^2) + \mathcal{O}(\ell^3 \cdot k) = \mathcal{O}(\ell^4)$ vertices in the forest $G' - S'$, and thus $\mathcal{O}(\ell^4)$ vertices in G'. □

Using a 2-approximation algorithm for feedback vertex set we obtain a corollary similar to the one given in the previous section.

Corollary 2. *There is a polynomial-time algorithm that given an instance* $(G = (V, E), k)$ *of* Treewidth *computes an equivalent instance* $(G' = (V', E'), k)$ *such that* $V' \subseteq V$ *and* $|V'| \in \mathcal{O}((\ell^*)^4)$*, where* ℓ^* *is the size of a minimum feedback vertex set of* G*.*

5 Kernelization Lower Bounds

In this section we present our lower bound results for Treewidth and Weighted Treewidth; due to space restrictions the proofs are deferred to the full version [7]. We show the following results.

Theorem 3. *Unless the AND-distillation conjecture fails and all coNP-complete problems have OR-distillation algorithms,* Treewidth parameterized by target treewidth plus the size of a given modulator to cluster graphs *does not admit a polynomial kernel.*

For this result we use that Treewidth on co-bipartite graphs is NP-complete [1, Theorem 3.3]. The key idea is that by identifying one bipartition from each of t co-bipartite graphs into one clique one obtains a graph that has treewidth at most some integer k if and only if all t graphs have treewidth at most k; this constitutes an AND-composition [5].

Theorem 4. Treewidth parameterized by a modulator to co-cluster graphs *does not admit a polynomial kernelization unless NP* $\subseteq$ *coNP/poly (which would imply a collapse of the polynomial hierarchy to its third level).*

The proof goes by giving a cross-composition from Treewidth to the target problem, i.e., a reduction of some t instances of Treewidth into one instance of the parameterized problem with a *small* parameter value, which is **yes** iff one of the t instances is **yes** (for details on cross-composition see [6]). We use that the join of t graphs each on n vertices has a treewidth equaling $(t-1)\cdot n$ plus the minimum treewidth of any of the graphs (a corollary of work by Bodlaender and Möhring [11]). To turn the join of the graphs into an almost co-cluster, their edges are replaced by a set of m cliques (each connected to the endpoints of one edge per graph), which are then added to the modulator.

Theorem 5. Weighted Treewidth parameterized by a vertex cover *does not admit a polynomial kernelization unless NP* $\subseteq$ *coNP/poly.*

This final result is obtained by an extension of the proof for Theorem 4. In addition to that construction, we also replace the join edges by a set of $2n^2 \log t$ vertices of high weight. The effect on the treewidth is the same, but adding the vertices to the modulator we obtain an independent set instead of a co-cluster.

6 Conclusions

We considered different parameterizations for the Treewidth problem and obtained both positive and negative results for the existence of polynomial kernels. Our first positive result, a cubic kernel for Treewidth parameterized by a vertex cover, is interesting as its algorithm largely consists of elements of existing preprocessing heuristics for Treewidth, and thus also sheds some light on the experimentally observed success of these heuristics. Our second positive result, a polynomial kernel for Treewidth parameterized by a feedback vertex set, is not only interesting from a theoretical point of view, but we expect that some of the reduction rules are also of practical value. In that regard it would be interesting to carry out an algorithmic engineering study, and implement (part of) the algorithms. E.g., does the *Degree* $k+1$ *Almost Simplicial Vertex Rule* (Rule 9) give significant reductions of instance sizes for practical instances? This could be compared to the experiments reported in [9,15].

Our lower bounds, for distance from cluster respectively co-cluster graphs, rule out polynomial kernels for Treewidth for a number of possibly interesting parameters like distance from cographs or from chordal, interval, or split graphs. We recall also that Treewidth is NP-hard on bipartite graphs (easily seen by subdividing all edges), implying that parameterization by distance ℓ from bipartite or perfect graphs does not even permit $\mathcal{O}(n^{f(\ell)})$ time algorithms unless P = NP; hence also no polynomial kernels.

Apart from improving the kernel sizes, e.g., for parameterization by a feedback vertex set, or giving polynomial lower bounds (e.g., $\Omega(\ell^2)$), it seems interesting whether parameter-wise one can obtain stronger results. For example, is

there a polynomial kernel with respect to the size of a modulator to outerplanar graphs or even to planar graphs (note that membership in P or NP-hardness of Treewidth on planar graphs is open). For the lower bounds, further research may consider whether they can be extended to distance from a single clique (which is both cluster and co-cluster); complementary to a vertex cover. If true, then the method needs to generalize proofs using ANDs and ORs of sets of input instances (cf. [5]). If instead there is a polynomial kernel, then the reduction rules have to handle large cliques, the main building block of our lower bounds.

References

1. Arnborg, S., Corneil, D.G., Proskurowski, A.: Complexity of finding embeddings in a k-tree. SIAM Journal on Algebraic and Discrete Methods 8, 277–284 (1987)
2. Becker, A., Geiger, D.: Optimization of Pearl's method of conditioning and greedy-like approximation algorithms for the vertex feedback set problem. Artificial Intelligence 83, 167–188 (1996)
3. Bodlaender, H.L.: A linear time algorithm for finding tree-decompositions of small treewidth. SIAM Journal on Computing 25, 1305–1317 (1996)
4. Bodlaender, H.L.: Necessary edges in k-chordalizations of graphs. Journal of Combinatorial Optimization 7, 283–290 (2003)
5. Bodlaender, H.L., Downey, R.G., Fellows, M.R., Hermelin, D.: On problems without polynomial kernels. Journal of Computer and System Sciences 75, 423–434 (2009)
6. Bodlaender, H.L., Jansen, B.M.P., Kratsch, S.: Cross-composition: A new technique for kernelization lower bounds. In: STACS 2011, pp. 165–176 (2011)
7. Bodlaender, H.L., Jansen, B.M.P., Kratsch, S.: Preprocessing for treewidth: A combinatorial analysis through kernelization. CoRR, abs/1104.4217 (2011)
8. Bodlaender, H.L., Koster, A.M.C.A.: Safe separators for treewidth. Discrete Mathematics 306, 337–350 (2006)
9. Bodlaender, H.L., Koster, A.M.C.A., van den Eijkhof, F.: Pre-processing rules for triangulation of probabilistic networks. Computational Intelligence 21(3), 286–305 (2005)
10. Bodlaender, H.L., Koster, A.M.C.A., Wolle, T.: Contraction and treewidth lower bounds. Journal of Graph Algorithms and Applications 10, 5–49 (2006)
11. Bodlaender, H.L., Möhring, R.H.: The pathwidth and treewidth of cographs. SIAM Journal on Discrete Mathematics 6, 181–188 (1993)
12. Bodlaender, H.L., Rotics, U.: Computing the treewidth and the minimum fill-in with the modular decomposition. Algorithmica 36, 375–408 (2003)
13. Clautiaux, F., Carlier, J., Moukrim, A., Négre, S.: New lower and upper bounds for graph treewidth. In: Jansen, K., Margraf, M., Mastrolli, M., Rolim, J.D.P. (eds.) WEA 2003. LNCS, vol. 2647, pp. 70–80. Springer, Heidelberg (2003)
14. Clautiaux, F., Moukrim, A., Négre, S., Carlier, J.: Heuristic and meta-heuristic methods for computing graph treewidth. RAIRO Operations Research 38, 13–26 (2004)
15. van den Eijkhof, F., Bodlaender, H.L., Koster, A.M.C.A.: Safe reduction rules for weighted treewidth. Algorithmica 47, 138–158 (2007)
16. Guo, J., Niedermeier, R.: Invitation to data reduction and problem kernelization. ACM SIGACT News 38, 31–45 (2007)
17. Lagergren, J., Arnborg, S.: Finding minimal forbidden minors using a finite congruence. In: Leach Albert, J., Monien, B., Rodríguez-Artalejo, M. (eds.) ICALP 1991. LNCS, vol. 510, pp. 532–543. Springer, Heidelberg (1991)

Subset Feedback Vertex Set Is Fixed-Parameter Tractable

Marek Cygan[1], Marcin Pilipczuk[1],
Michał Pilipczuk[1], and Jakub Onufry Wojtaszczyk[2]

[1] Institute of Informatics, University of Warsaw, Poland
{cygan@,malcin@,mp248287@students.}mimuw.edu.pl
[2] Institute of Mathematics, University of Warsaw, Poland
(currently Google Inc., Cracow, Poland)
onufry@google.com

Abstract. The classical FEEDBACK VERTEX SET problem asks, for a given undirected graph G and an integer k, to find a set of at most k vertices that hits all the cycles in the graph G. FEEDBACK VERTEX SET has attracted a large amount of research in the parameterized setting, and subsequent kernelization and fixed-parameter algorithms have been a rich source of ideas in the field.

In this paper we consider a more general and difficult version of the problem, named SUBSET FEEDBACK VERTEX SET (SUBSET-FVS in short) where an instance comes additionally with a set $S \subseteq V$ of vertices, and we ask for a set of at most k vertices that hits all simple cycles passing through S. Because of its applications in circuit testing and genetic linkage analysis SUBSET-FVS was studied from the approximation algorithms perspective by Even et al. [SICOMP'00, SIDMA'00].

The question whether the SUBSET-FVS problem is fixed-parameter tractable was posed independently by Kawarabayashi and Saurabh in 2009. We answer this question affirmatively. We begin by showing that this problem is fixed-parameter tractable when parametrized by $|S|$. Next we present an algorithm which reduces the given instance to $2^k n^{O(1)}$ instances with the size of S bounded by $O(k^3)$, using kernelization techniques such as the 2-Expansion Lemma, Menger's theorem and Gallai's theorem. These two facts allow us to give a $2^{O(k \log k)} n^{O(1)}$ time algorithm solving the SUBSET FEEDBACK VERTEX SET problem, proving that it is indeed fixed-parameter tractable.

1 Introduction

FEEDBACK VERTEX SET (FVS) is one of the long–studied problems in the algorithms area. It can be stated as follows: given an undirected graph G on n vertices and a parameter k decide if one can remove at most k vertices from G so that the remaining graph does not contain a cycle, i.e., is a forest. The problem of finding feedback sets in undirected graphs arises in a variety of applications in genetics, circuit testing, artificial intelligence, deadlock resolution, and analysis of manufacturing processes [15].

Because of its importance the feedback vertex set problem was studied from the approximation algorithms perspective in different variants and generalisations including DIRECTED FEEDBACK VERTEX SET and SUBSET FEEDBACK VERTEX SET (see [14]

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 449–461, 2011.

and [16] for further references). In this paper we will study the SUBSET FEEDBACK VERTEX SET problem from the parametrized complexity perspective.

In the parameterized complexity setting, an instance comes with an integer parameter k — formally, a parameterized problem Q is a subset of $\Sigma^* \times \mathbb{N}$ for some finite alphabet Σ. We say that a problem is *fixed-parameter tractable* (*FPT*) if there exists an algorithm solving any instance (x, k) in time $f(k)\text{poly}(|x|)$ for some (usually exponential) computable function f. Intuitively, the parameter k measures the hardness of the instance. Fixed-parameter tractability has received much notice as a method of effectively solving NP-hard problems for instances with a small parameter value.

The long line of research concerning FVS in the parameterized complexity setting contains [1,2,6,7,10,12,13,18,19,23,24]. Currently the fastest known algorithm works in $3^k n^{O(1)}$ time [9]. Thomassé [26] has shown a quadratic kernel for this problem improving previous results [3,5]. The directed version has been proved to be FPT in 2008 by Chen et al. [8], closing a long-standing open problem in the parameterized complexity community. The natural question concerning the parameterized complexity of the SUBSET FEEDBACK VERTEX SET problem was posed independently by Kawarabayashi at the 4th workshop on Graph Classes, Optimization, and Width Parameters (GROW 2009) and by Saurabh at the Dagstuhl seminar 09511 [11].

Notation. Let us now introduce some notation. Let $G = (V, E)$ be a simple undirected graph with n vertices. A *cycle* in G is a sequence of vertices $v_1 v_2 \dots v_m \in V$ such that $v_i v_{i+1} \in E$ and $v_m v_1 \in E$. We say a cycle is *simple* if $m > 2$ and the vertices v_i are pairwise different. We will also consider multigraphs (i.e., graphs with multiple edges and loops), in which a simple cycle can have two vertices if there is a multiple edge between them. We call an edge $vw \in E$ a *bridge* if in $(V, E \setminus \{vw\})$ the vertices v and w are in different connected components. Note that no simple cycle can contain a bridge as one of its edges. Given subsets $X, Y \subseteq V$, by $E(X, Y)$ we denote the set of edges with one endpoint in X and the other in Y. By $G[X]$ we denote the subgraph induced by X with the edge set $E(X, X)$. By $N(X)$ we denote the neighbourhood of X, i.e. $\{u \in V \setminus X : \exists_{v \in X} uv \in E\}$. For a subset of edges $E' \subseteq E$ by $V(E')$ we denote the set of all endpoints of edges from the set E'.

Problem definitions. In this paper we study the SUBSET FEEDBACK VERTEX SET problem (SUBSET-FVS), where an instance comes with a subset of vertices S, and we ask for a set of at most k vertices that hits all simple cycles passing through S. It is easy to see that SUBSET-FVS is a generalisation of FVS by putting $S = V$. The weighted version of SUBSET-FVS was introduced by Even et al. [15] as a generalization of two problems: FEEDBACK VERTEX SET and NODE MULTIWAY CUT. Even et al. motivate SUBSET-FVS problem by explaining its applicability to genetic linkage.

SUBSET FEEDBACK VERTEX SET (SUBSET-FVS) **Parameter:** k
Input: An undirected graph $G = (V, E)$, a set $S \subseteq V$ and a positive integer k
Question: Does there exist a set $T \subseteq V$ such that $\lvert T\rvert \leq k$ and no simple cycle in $G[V \setminus T]$ contains any vertex of S?

We also define a variant of SUBSET-FVS, where the set S is a subset of edges of G.

EDGE SUBSET FEEDBACK VERTEX SET (EDGE-SUBSET-FVS) **Parameter:** k
Input: An undirected graph $G = (V, E)$, a set $S \subseteq E$ and a positive integer k
Question: Does there exist a set $T \subseteq V$ with $|T| \leq k$, such that no simple cycle in $G[V \setminus T]$ contains an edge from S?

The two problems stated above are equivalent. To see this, note that if (G, S, k) is an instance of SUBSET-FVS, we create an instance (G, S', k) of EDGE-SUBSET-FVS by selecting as S' all the edges incident to any vertex of S. Then any simple cycle passing through a vertex of S has to pass through an edge of S', and conversely, any cycle passing through an edge of S' contains a vertex from S. In the other direction, if (G, S', k) is an instance of EDGE-SUBSET-FVS, obtain G' by replacing each edge $uv \in S'$ by a path $u - x_{uv} - v$ of length 2, and solve the SUBSET-FVS instance (G', S, k) where $S = \{x_e : e \in S'\}$. Clearly both reductions work in polynomial time and do not change the parameter. Thus, in the rest of this paper we focus on solving EDGE SUBSET FEEDBACK VERTEX SET. A simple cycle containing an edge from S is called an *S–cycle*.

Let us recall here the definitions of two other problems related to SUBSET-FVS.

NODE MULTIWAY CUT **Parameter:** k
Input: An undirected graph $G = (V, E)$, a set of vertices $\mathcal{T} \subseteq V$, called *terminals*, and a positive integer k
Question: Does there exist a set $T \subseteq V$ of at most k non-terminals, such that no two terminals are in the same connected component of $G[V \setminus T]$?

NODE MULTICUT **Parameter:** k
Input: An undirected graph $G = (V, E)$, a set of pairs of vertices $\mathcal{T} \subseteq V \times V$, called *terminal pairs*, and a positive integer k
Question: Does there exist a set $T \subseteq V$ of at most k non-terminals, such that no terminal pair is contained in one connected component of $G[V \setminus T]$?

Our contributions. We develop a $2^{O(k \log k)} n^{O(1)}$ algorithm for EDGE-SUBSET-FVS (which implies an algorithm of the same time complexity for SUBSET-FVS). This result resolves an open problem posted in 2009 independently by Kawarabayashi and by Saurabh. To achieve this result we use several tools such as iterative compression, the 2-Expansion Lemma, Menger's theorem, Gallai's theorem and the algorithm for the MULTIWAY CUT problem. Some of our ideas were inspired by previous FPT results: the algorithm for MULTICUT parameterized by $(|\mathcal{T}|, k)$ by Guillemot [17], the $37.7^k n^{O(1)}$–time algorithm for FVS by Guo et al. [18] and the quadratic kernel for FVS by Thomassé [26].

Related work. As observed by Even et al. [15] the weighted version of SUBSET-FVS is a generalisation of NODE MULTIWAY CUT. It is straightforward to adjust their reduction to the unweighted parameterized case.

Recently a lot of effort was put into developing kernelization and FPT algorithms for terminal separation problems, including the quadratic kernel [26] and the fast FPT algorithm [9] for FVS and the results resolving the parametrized complexity status of MULTICUT independently obtained by Bousquet et al. [4] and by Marx and Razgon [22]. To the best of our knowledge, though, none of those results implies an FPT algorithm for SUBSET-FVS.

SUBSET-FVS was studied from the approximation perspective and the best known approximation algorithm by Even at al. [16] gives approximation ratio equal to 8.

We were recently informed that an FPT algorithm for SUBSET-FVS was independently discovered by Kawarabayashi and Kobayashi [20]. Their algorithm uses significantly different techniques (minor theory) and its dependency on k in the running time is worse than $2^{O(k \log k)}$.

Outline of the paper. In Section 2 we present an FPT algorithm for EDGE-SUBSET-FVS when parameterized by $|S|$, where for the sake of presentation we give an easy-to-describe $f(|S|)n^{O(1)}$ algorithm at the cost of a fast growing function f. This algorithm can be enhanced using techniques of Guillemot [17] so that the function f is replaced by $2^{O(k \log |S|)}$. Later in Section 3 we develop a set of branchings and reductions which work in $2^k n^{O(1)}$ time and reduce the size of the set S to $O(k^3)$.

The proofs of all results marked with the spade symbol (♠), including the enhanced algorithm from Section 2, the detailed proof of a crucial lemma from Section 3, as well as some details of the final reductions in Section 3, do not appear in this extended abstract due to space limitations.

2 EDGE-SUBSET-FVS Parameterized by $|S|$

In this section we concentrate on solving the EDGE-SUBSET-FVS problem parameterized by $|S|$, which means that our complexity function can be exponentially dependent on the number of edges in the set S. This is the first step towards obtaining an FPT algorithm when parameterized by k. Observe that we may assume $k < |S|$ since otherwise we may delete one vertex from each edge from the set S thus removing all edges from the set S from our graph. We begin by showing an FPT algorithm which is easy to understand and later we present methods to improve the time complexity. We use the fact that NODE MULTICUT is FPT when parameterized by $(k, |\mathcal{T}|)$ which was shown by Marx [21].

Theorem 1. *There exists an algorithm solving the* EDGE SUBSET FEEDBACK VERTEX SET *problem in $f(|S|)n^{O(1)}$ time, for some function f.*

Proof. Let T be some solution of EDGE-SUBSET-FVS. Our new parametrization, by $|S|$, allows us to guess, by checking all possibilities, the subset $T_S = T \cap V(S)$ that is removed by the solution T. Moreover, our algorithm guesses how the set $V(S) \setminus T_S$ is partitioned into connected components in the graph $G[V \setminus T]$ with the edges from S removed. Clearly both the number of subsets and of possible partitions is a function of $|S|$. For a partition $\mathcal{P} = \{P_1, \ldots, P_m\}$ of $V(S) \setminus T_S$ we form a multigraph $G_{\mathcal{P}}$ on the set $\{P_1, \ldots, P_m\}$ by adding an edge $P_i P_j$ for every edge $uv \in S$, where

$u \in P_i, v \in P_j$. Now we check whether there exists an edge in $G_{\mathcal{P}}$ which is not a bridge. If that is the case we know that the partition $\mathcal{P}$ does not correspond to any solution of EDGE-SUBSET-FVS, as any simple cycle in $G_{\mathcal{P}}$ can be converted into a simple cycle in G — hence we skip this partition. Otherwise we create a set of pairs $\mathcal{T}$, containing all pairs of vertices from the set $V(S) \setminus T_S$ that belong to different sets in the partition $\mathcal{P}$. Formally $\mathcal{T} = \{(v_i, v_j) : v_i \in P_{i'}, v_j \in P_{j'}, i' \neq j'\}$. Because of the properties of the multigraph $G_{\mathcal{P}}$ it is sufficient to ensure that no pair from the set $\mathcal{T}$ is contained in one connected component, hence the last step is calling an algorithm for the NODE MULTICUT problem with parameter $k - |T_S|$. If the call returns a positive answer and a solution X, the set $T_S \cup X$ is a solution to EDGE-SUBSET-FVS: the connected components of $G[V \setminus (T_S \cup X)]$ induce a partition of $V(S) \setminus (T_S \cup X)$ that is a subpartition of $\mathcal{P}$ and thus all remaining edges of S are bridges in $G[V \setminus (T_S \cup X)]$. Note that we do not require here that $X \cap V(S) = \emptyset$ nor that the induced partition of $V(S) \setminus (T_S \cup X)$ is exactly the partition $\mathcal{P}$ (being a subpartition is sufficient). On the other hand, if the answer to EDGE-SUBSET-FVS is positive, the NODE MULTICUT call returns a solution for at least one choice of T_S and $\mathcal{P}$, the one implied by the EDGE-SUBSET-FVS solution. Observe that $|\mathcal{T}| = O(|S|^2)$ so we obtain an FPT algorithm for the EDGE SUBSET FEEDBACK VERTEX SET problem parameterized by $|S|$.

Function EdgeSubsetFeedbackVertexSet(G, S, k) {parameterized by $(|S|, k)$}
1: **for all** subsets $T_S \subseteq V(S), |T_S| \leq k$ **do**
2: **for all** partitions $\mathcal{P} = \{P_1, \ldots, P_m\}$ of $V(S) \setminus T_S$ **do**
3: form a multigraph $G_{\mathcal{P}}$ on the set $\{P_1, \ldots, P_m\}$ by adding an edge P_iP_j for every edge $uv \in S, u \in P_i, v \in P_j$.
4: **if** all edges in $G_{\mathcal{P}}$ are bridges **then**
5: let $\mathcal{T} = \{(v_i, v_j) : v_i \in P_{i'}, v_j \in P_{j'}, i' \neq j'\}$
6: **if** MultiCut$(G[V \setminus T_S], \mathcal{T}, k - |T_S|)$ returns (YES, X) **then**
7: **return** $T_S \cup X$
8: **return** NO

Using techniques of Guillemot [17] we can improve the time complexity.

Theorem 2 (♠). *There exists an algorithm solving the* EDGE SUBSET FEEDBACK VERTEX SET *problem in* $2^{O(k \log |S|)} n^{O(1)}$ *time.*

3 EDGE-SUBSET-FVS Parameterized by k

In this section we show an FPT algorithm for EDGE SUBSET FEEDBACK VERTEX SET.

Begin by noting that, using standard arguments, one can show that EDGE-SUBSET-FVS is *self-reducible* — i.e., if we have an algorithm that solves EDGE-SUBSET-FVS, we can also find a witness: a set T that intersects all cycles passing through S.

We now follow the idea of *iterative compression* proposed by Reed et al. [25]. First, note that if $V' \subseteq V$ and T is a feasible solution to an EDGE-SUBSET-FVS instance (G, S, k), then $V' \cap T$ is a feasible solution to the instance $(G[V'], S', k)$, where $S' = S \cap E(G[V'])$. Thus, if the answer for $(G[V'], S', k)$ is negative, so is the answer for (G, S, k). Let $V = \{v_1, v_2, \ldots, v_n\}$ be an arbitrary ordering of the set of vertices of G. We consecutively construct solutions to EDGE-SUBSET-FVS for instances $\mathcal{I}_i =$

$(G[V_i], S_i, k)$, where $V_i = \{v_1, v_2, \ldots, v_i\}$ and $S_i = S \cap E(G[V_i])$. When looking for a solution for graph $G[V_{i+1}]$, we use the fact that if T_i is a solution for $\mathcal{I}_i$, then $Z_{i+1} = T_i \cup \{v_{i+1}\}$ is a solution for $(G[V_{i+1}], S_{i+1}, k+1)$ — a solution for our problem with the parameter increased by one.

We start with a standard branching into $2^{|Z_{i+1}|}$ subcases, guessing which vertices from Z_{i+1} are taken into a solution to the instance $\mathcal{I}_{i+1}$. Let us focus on a fixed branch, where we decided to take $T_Z \subseteq Z_{i+1}$ into a solution and denote $Z = Z_{i+1} \setminus T_Z$. We delete T_Z from the graph G, reduce S to $S \cap E(G \setminus T_Z)$, and decrease k by $|T_Z|$, arriving at the following subproblem.

DISJOINT EDGE-SUBSET-FVS **Parameter:** k and $|Z|$
Input: A EDGE-SUBSET-FVS instance (G, S, k) together with a set $Z \subseteq V(G)$ that is a solution to the EDGE-SUBSET-FVS instance $(G, S, |Z|)$
Question: Does there exist a solution to (G, S, k) that is disjoint with Z?

Definition 1. *We say that a* DISJOINT EDGE-SUBSET-FVS *instance* (G, S, k, Z) *is a* maximal *YES-instance if every feasible solution to* EDGE-SUBSET-FVS *instance* (G, S, k) *is disjoint with* Z. *We say that a* DISJOINT EDGE-SUBSET-FVS *instance* (G', S', k', Z') *is a* properly reduced *instance* (G, S, k, Z) *if the following holds:*

1. $|V(G')| \leq |V(G)|$ *and* $k' \leq k$;
2. (G', S', k') *is an equivalent* EDGE-SUBSET-FVS *instance to* (G, S, k);
3. *if* (G, S, k, Z) *is a maximal* DISJOINT EDGE-SUBSET-FVS *YES-instance then* (G', S', k', Z') *is a maximal YES-instance.*

Note that we do not insist that if (G, S, k, Z) is a DISJOINT EDGE-SUBSET-FVS YES-instance, (G', S', k', Z') is a YES-instance.

Theorem 3. *There exists a polynomial-time algorithm* $\mathcal{R}$ *that, given a* DISJOINT EDGE-SUBSET-FVS *instance* (G, S, k, Z), *either:*

1. *returns a properly reduced instance* (G', S', k', Z') *with* $|S'| = O(k|Z|^2)$; *or*
2. *returns IGNORE; in this case* (G, S, k, Z) *is not a* maximal *YES-instance.*

We first show that Theorem 3 leads to the desired FPT algorithm for EDGE-SUBSET-FVS. In each step of the iterative compression, in each of $2^{|Z_{i+1}|}$ branches, we run the algorithm $\mathcal{R}$. If it gives the first answer, we invoke the algorithm from Theorem 2 on EDGE-SUBSET-FVS instance (G', S', k'), leading to running time $2^{O(k \log k)} n^{O(1)}$. Note that (G', S', k') is a YES-instance iff (G, S, k) is a YES-instance, and any solution (even not disjoint with Z) to (G, S, k) can be extended to a solution of $\mathcal{I}_{i+1}$ by taking its union with T_Z. In the case of the second answer, we ignore this branch. Obviously, if $\mathcal{I}_{i+1}$ is a NO-instance, the algorithm cannot find a solution. Otherwise, let T be a solution to $\mathcal{I}_{i+1}$ with maximum possible intersection with Z_{i+1}. We claim that the algorithm finds a solution in the branch $T_Z = T \cap Z_{i+1}$. Indeed, then (G, S, k, Z) is a maximal YES-instance to DISJOINT EDGE-SUBSET-FVS and the algorithm $\mathcal{R}$ cannot return IGNORE. Thus we obtain a EDGE-SUBSET-FVS instance (G', S', k'), equivalent to the YES-instance (G, S, k), and the algorithm from Theorem 2 finds a solution.

The proof of Theorem 3 consists of a set of polynomial-time *proper reductions* (in the sense of Definition 1), each either decreasing $|V(G)|$ or decreasing $|E(G)|$ while not changing $|V(G)|$.

In all our reductions the last property of Definition 1 is satisfied, as there is a trivial way to extend a solution in the reduced instance a solution in the original instance. We will not provide a description of this extension. Some reductions may result with an IGNORE answer, in which case the answer is immediately returned from this branch. Note that the last property of Definition 1 implies that all DISJOINT EDGE-SUBSET-FVS instances in the current sequence of reductions are not maximal YES-instances. We assume that at each step, the lowest–numbered applicable reduction is used. If no reduction is applicable, we clam that $|S| = O(k|Z|^2)$.

We start with an obvious reduction. Note that if it is not applicable, every edge in S is contained in some simple cycle.

Reduction 1. Remove all bridges and all connected components not containing any edge from S.

3.1 The Outer–Abundant Lemma

In this section we consider an instance of DISJOINT EDGE-SUBSET-FVS (G, S, k, Z), where $G = (V, E)$. We assume that Reduction 1 is not applicable, i.e., every edge in S belongs to some simple cycle. The approach here is based on ideas from the quadratic kernel for the classical FEEDBACK VERTEX SET problem [26], however, a few aspects need to be adjusted to better fit our needs.

Definition 2. *A set $F \subseteq V$ is called* outer–abundant *iff:*

(a) $G[F]$ is connected,
(b) there are no edges from S in $G[F]$,
(c) there at least $10k$ edges from S incident with F.

Lemma 1 (♠, The outer–abundant lemma). *Let F be an outer–abundant set. If Reduction 1 is not applicable, then in polynomial time one can find a nonempty set $X \subseteq V \setminus F$ such that the following condition is satisfied: if there exists a solution A for* EDGE-SUBSET-FVS *on (G, S, k) such that $A \cap F = \emptyset$, then there exists a solution A' such that $A' \cap F = \emptyset$ and $X \subseteq A'$.*

The Lemma allows us to greedily assume X is in the solution we are looking for (after we ensure that it is disjoint with F), and either take it into the solution (if $X \cap Z = \emptyset$) or return IGNORE (if $X \cap Z \neq \emptyset$). The proof of Lemma 1 involves kernelization techniques such as the 2-Expansion Lemma, Menger's theorem and Gallai's theorem. As a direct application of Lemma 1, we obtain the following reduction rule. Note that if it is not applicable, there are at most $10k|Z|$ edges from S incident with Z.

Reduction 2. Let $v \in Z$ be a vertex that is incident to at least $10k$ edges from S. Apply Lemma 1 to the outer–abundant set $F = \{v\}$ obtaining a set X. Then, as we seek for a solution disjoint with Z, we may greedily take X into the solution. If $X \cap Z = \emptyset$, we remove X and decrease k by $|X|$, otherwise we return IGNORE.

3.2 Bubbles

Recall that our goal is to reduce the size of S. After Reduction 2, there are $O(k|Z|)$ edges from S incident with Z. Thus, we need to care only about $S \cap E(G[V \setminus Z])$.

As Z is a feasible solution to EDGE-SUBSET-FVS on $(G, S, |Z|)$, every edge from $S \cap E(G[V \setminus Z])$ has to be a bridge in $G[V \setminus Z]$. After removing those bridges $G[V \setminus Z]$ becomes an union of connected components not having any edge from S. We call each such a component a *bubble*. Denote the set of bubbles by $\mathcal{D}$. On $\mathcal{D}$ we have a natural structure of a graph $H = (\mathcal{D}, E_{\mathcal{D}})$, where $IJ \in E_{\mathcal{D}}$ iff components I and J are connected by an edge from S. As Z is a solution, H is a forest and each I, J connected in H are connected in G by a single edge from S.

Consider $I \in \mathcal{D}$. Denote the set of vertices of I by V_I. Note that if at most a single edge leaves I (that is, $|E(V_I, V \setminus V_I)| \leq 1$) then V_I would be removed while processing Reduction 1. The following reduction sorts out bubbles with exactly two outgoing edges and later we assume that for every $I \in \mathcal{D}$ we have $|E(V_I, V \setminus V_I)| \geq 3$.

Reduction 3. Let us assume that $|E(V_I, V \setminus V_I)| = 2$ and let $\{u, v\} = N(V_I)$ (possibly $u = v$). Each cycle passing through a vertex from V_I is either fully contained in I and thus non–S–cycle, or exits V_I through u and v. We remove V_I from the graph and replace it with a single edge uv, belonging to S iff any one of the two edges in $E(V_I, V \setminus V_I)$ is in S. If the addition of the edge uv lead to a multiple edge or a loop, we immediately resolve it: If uv is a loop and $uv \notin S$, we delete it. If uv is a loop and $uv \in S$, we return IGNORE, as the fact that Z is a solution to $(G, S, |Z|)$ implies that $u \in Z$. If uv is a multiple edge and no edge between u and v is from S, we delete the new edge uv. If uv is a multiple edge and one of the edges between u and v is S, we first note that, since Z is a solution to $(G, S, |Z|)$, u or v is in Z. If both are in Z, we return IGNORE, otherwise we delete $\{u, v\} \setminus Z$ from the graph and decrease k by one.

We are now left with bubbles that have at least three outgoing edges. We classify those bubbles according to the number of edges that connect them to other bubbles, that is $\deg_H(I)$.

Definition 3. *We say that a bubble $I \in \mathcal{D}$ is*

(a) a solitary bubble *if* $\deg_H(I) = 0$,
(b) a leaf bubble *if* $\deg_H(I) = 1$,
(c) an edge bubble *if* $\deg_H(I) = 2$,
(d) an inner bubble *if* $\deg_H(I) \geq 3$.

Denote by $\mathcal{D}_s, \mathcal{D}_l, \mathcal{D}_e, \mathcal{D}_i$ the sets of appropriate types of bubbles.

We show that we can do some reductions to make following inequalities hold:

$$|\mathcal{D}_l| = O(k|Z|^2), \qquad |\mathcal{D}_i| < |\mathcal{D}_l|, \qquad |\mathcal{D}_e| < 3(|Z| + k) + |\mathcal{D}_i| + |\mathcal{D}_l|.$$

Note that these conditions imply that $|\mathcal{D} \setminus \mathcal{D}_s| = O(k|Z|^2)$. As edges of H create a forest over $\mathcal{D} \setminus \mathcal{D}_s$, this bounds the number of edges from S not incident with Z by $O(k|Z|^2)$, as desired.

Lemma 2. $|\mathcal{D}_i| < |\mathcal{D}_l|$.

Proof. As H is a forest, then $|E_{\mathcal{D}}| < |\mathcal{D}| - |\mathcal{D}_s| = |\mathcal{D}_i| + |\mathcal{D}_e| + |\mathcal{D}_l|$. Moreover, $2|E_{\mathcal{D}}| = \sum_{I \in \mathcal{D}} \deg_H(I) \geq 3|\mathcal{D}_i| + 2|\mathcal{D}_e| + |\mathcal{D}_l|$. Therefore $|\mathcal{D}_l| > |\mathcal{D}_i|$.

Reduction 4. If $|\mathcal{D}_e| \geq 3(|Z| + k) + |\mathcal{D}_i| + |\mathcal{D}_l|$, then return IGNORE.

Proof (of correctness). We first show that the number of edge bubbles not adjacent to any other edge bubble in H is at most $|\mathcal{D}_i|+|\mathcal{D}_l|$. Let us root each connected component of H in an arbitrary leaf and for any bubble $I \in \mathcal{D}_e$ let $\varphi(I)$ be the only child of I. Observe that this mapping is injective and maps the set of edge bubbles isolated in $H[\mathcal{D}_e]$ into $\mathcal{D}_i \cup \mathcal{D}_l$.

We now prove by contradiction if we apply the reduction, every feasible solution of (G, S, k) contains a vertex from Z. As edge bubbles have degree 2 in H, $H[\mathcal{D}_e]$ is a set of paths of non-zero length and isolated vertices. There are at least $3(|Z| + k)$ vertices contained in the paths. Let M be a maximal matching in $H[\mathcal{D}_e]$. If a path contains l vertices (for $l \geq 2$), it has a matching of cardinality $\lfloor \frac{l}{2} \rfloor \geq \frac{l}{3}$. Therefore, in $H[\mathcal{D}_e]$ we have a matching of cardinality at least $|Z| + k$. Let us examine an arbitrary $IJ \in M$. Recall that at least three edges leave each bubble. As each of V_I, V_J is adjacent to two other bubbles by single edges, it has to be connected to Z as well. Choose u_I, u_J — vertices from Z such that $u_I \in N(V_I)$ and $u_J \in N(V_J)$. We see that there is a path from u_I to u_J passing through an edge from S: it goes from u_I to I, then to J through an edge from S, and then to u_J. As M is a matching, such paths are vertex–disjoint for all $IJ \in M$, except for endpoints u_I and u_J.

If we have a solution disjoint with Z of cardinality at most k, there are at least $|Z|$ pairs $IJ \in M$, where neither I nor J contains a vertex from the solution. Now we construct a graph $P = (Z, E_P)$ such that $u_I u_J \in E_P$ if u_I, u_J have been chosen for some $IJ \in M$, where I and J are solution–free. We prove that P has to be a forest. Indeed, otherwise there would be a cycle in P — and by replacing each edge from it by associated path, we construct an S-cycle in G (as paths in which edges from E_P originated are vertex–disjoint). This cycle does not contain any vertex from the solution, as it passes only through Z and solution–free bubbles. However, as $|E_P| \geq |Z|$, P cannot be a forest; the contradiction ends the proof.

3.3 The Leaf Bubble Reduction

We are left with the leaf bubbles and we need to show reductions that lead to $|\mathcal{D}_l| = O(k|Z|^2)$. We do this by a single large reduction described in this subsection. It proceeds in a number of steps. Each step either returns IGNORE (thus ending the reduction) or — after, possibly, modifying G — passes to the next step. Each step is not a standalone reduction, as it may increase $|E(G)|$. However, if the reduction below is fully applied, it either returns IGNORE or reduces $|V(G)|$.

Let I be a leaf bubble. As there are at least three edges leaving I, each leaf bubble is connected to Z by at least two edges. We begin with a bit of preprocessing:

Step 1. As long as there are two vertices v, v' in Z with $vv' \notin E$, and at least $k + 1$ bubbles, each connected to both v and v' by edges not in S, we add an edge vv' to E, with $vv' \notin S$.

Lemma 3 (♠). *The output (G', S, k) of Step 1 and the input (G, S, k) have equal sets of feasible solutions.*

Now for each bubble I with vertex set V_I we choose arbitrarily two of the edges connecting it to Z: e_I and e'_I. Additionally assume that if $S \cap E(V_I, Z) \neq \emptyset$ then $e_I \in S$. Let v_I and v'_I be the endpoints of e_I and e'_I in Z (possibly $v_I = v'_I$). We say that a bubble I is *associated* with vertices v_I, v'_I. If two leaf bubbles I_1, I_2 are connected in H (form a K_2 in H), by an edge $e_{I_1 I_2} \in S$, we call them a *bubble–bar*. The proofs of the following lemmata proceed along lines similar to the proof of correctness for Reduction 4:

Lemma 4 (♠). *If there are at least $|Z|^2(k+2)$ leaf bubbles I such that $e_I \in S$ then every feasible solution of (G, S, k) contains a vertex from Z.*

Lemma 5 (♠). *If there are at least $|Z|^2(k+1)$ leaf bubbles I such that $v_I v'_I \in S$, then every feasible solution of (G, S, k) contains a vertex from Z.*

Lemma 6 (♠). *If there are at least $|Z|^2(k+2)$ bubble–bars, then any feasible solution to (G, S, k) contains a vertex from Z.*

Step 2. If any of the situations from Lemmata 4, 5 and 6 occur, return IGNORE.

Summing all the obtained bounds, we can count almost all the leaf bubbles (possibly more than once) and bound their number by $O(k|Z|^2)$. The ones that are left satisfy the following definition:

Definition 4. *A leaf bubble I satisfying the following three conditions is called a* clique bubble*:*

(a) $G[N(V_I) \cap Z]$ is a clique not containing any edge from S,
(b) I is connected to Z by edges not belonging to S,
(c) I is connected to a non–leaf bubble.

Denote the only edge from S connecting a given clique bubble I with its neighbour bubble by $w_I w'_I$, with $w_I \in V_I$.

Lemma 7. *If there exists a feasible solution T for* Edge-Subset-FVS *on (G, S, k), then there exists a feasible solution T', which is disjoint from all clique bubbles in G.*

Proof. Let I be a clique bubble. Assume we have a feasible solution T, with $T \cap V_I \neq \emptyset$. We show $T' = (T \setminus V_I) \cup \{w'_I\}$ is also a feasible solution. Consider any S–cycle C in $G[V \setminus T']$. This cycle has to pass through V_I (possibly multiple times), or it would be an S–cycle in $G[V \setminus T]$, contrary to the assumption T was a feasible solution. Note that C has to enter and exit V_I through $N(V_I) \cap Z$, as the only vertex in $N(V_I) \setminus Z$ is w'_I, which is removed by T'. But then C can be shortened to C' by replacing every part contained in V_I by a single edge in Z (as $N(V_I) \cap Z$ is a clique). Now C' is disjoint from V_I and is an S–cycle due to the definition of the clique bubble. So C' is an S–cycle in $G[V \setminus T]$, a contradiction.

Note that the only vertex we added to T was w'_I, which does not belong to a clique bubble (it does not belong even to a leaf bubble, from property (c) in the definition of clique bubbles). Thus we can apply this procedure inductively, at each step reducing the number of vertices in T contained in clique bubbles, until none are left.

Assume there is a vertex $v \in Z$ such that $v \in N(V_{I_j})$ for some distinct clique–bubbles $I_1, I_2, \ldots, I_{10k}$. We show that the set $F = \{v\} \cup \bigcup_{j=1}^{10k} V_{I_j}$ is outer–abundant in G. Indeed, it is connected and due to the definition of bubbles and properties of the clique bubble definition, the subgraph $G[F]$ does not contain edges from S. Moreover, there are at least $10k$ edges from S incident with $G[F]$ — these are the edges connecting bubbles I_j with other bubbles, not contained in F as they are non–leaf ones due to property (c). This enables us to formulate the key step:

Step 3. If there is a vertex $v \in Z$ which is adjacent to at least $10k$ clique bubbles, we apply Lemma 1 to the set $F = \{v\} \cup \bigcup_{j=1}^{10k} V_{I_j}$ to obtain a set X. If $X \cap Z = \emptyset$, we remove X from the graph and decrease k by $|X|$, otherwise we return IGNORE.

Suppose there is a feasible solution T to (G, S, k). Due to Lemma 7 we may assume T to be disjoint with $F \setminus \{v\}$. Thus either T contains v, or it is disjoint with F, and by Lemma 1 there exists a solution containing X. This justifies the correctness of Step 3.

Thus we managed to reach the state when the number of leaf bubbles is bounded by $O(k|Z|^2)$. As we modified only the subgraph $G[Z]$, the sets $\mathcal{D}_i$, $\mathcal{D}_e$, $\mathcal{D}_l$ remain the same after modifications and we obtain a graph with $|S| = O(k|Z|^2)$. This completes the description of the $2^{O(k \log k)} n^{O(1)}$ algorithm for EDGE-SUBSET-FVS.

Now we summarize the steps made in this section to show clearly that the number of leaf bubbles is bounded by $O(k|Z|^2)$.

Assume no reduction is applicable. Note that in the last run, the last reduction may add some edges in Step 1. Let G' denote the modified graph. Let us check that the graph G' indeed has $O(k|Z|^2)$ edges from S:

1. The decomposition of $V(G) \setminus Z$ into bubbles is the same as the decomposition of $V(G') \setminus Z$ and bubbles that were inner or edge bubbles in G are, respectively, inner or edge bubbles in G';
2. If Step 2 is not applicable, there are at most $|Z|^2(k+2) - 1$ leaf bubbles connected to Z by an edge from S, at most $|Z|^2(k+1) - 1$ leaf bubbles associated with a pair of vertices connected with an edge from S, and at most $2|Z|^2(k+2)$ leaf bubbles connected to other leaf bubbles.
3. If Step 1 is not applicable, for any pair v, v' of vertices in Z with $vv' \notin E$ there are at most k leaf bubbles adjacent to both vertices of that pair through edges not in S.
4. If a leaf bubble is not a clique bubble, it either is connected to a leaf bubble (forming a bubble–bar), is connected to Z by an edge in S, has an edge from S between some two of its neighbours in Z, or has some two neighbours in Z not connected by an edge. The number of such bubbles in all four cases was estimated above. Thus, in total, there are at most $O(k|Z|^2)$ bubbles which are not clique bubbles.
5. Finally, if Step 3 is not applicable, there are at most $(10k-1)|Z|$ clique bubbles.
6. Thus $|\mathcal{D}_l| = O(k|Z|^2)$, moreover $|\mathcal{D}_i| \leq |\mathcal{D}_l|$ by Lemma 2 and $|\mathcal{D}_e| \leq 3(|Z| + k) + |\mathcal{D}_i| + |\mathcal{D}_l|$ by Reduction 4 — thus the number of edges in S not incident with Z is bounded by $O(k|Z|^2)$. We added no new edges to S, and the number of edges in S incident to Z was bounded by $O(k|Z|)$ in the input graph, thus in the output graph there are $O(k|Z|^2)$ edges from S, as desired.

References

1. Becker, A., Bar-Yehuda, R., Geiger, D.: Randomized algorithms for the loop cutset problem. J. Artif. Intell. Res (JAIR) 12, 219–234 (2000)
2. Bodlaender, H.L.: On disjoint cycles. Int. J. Found. Comput. Sci. 5(1), 59–68 (1994)
3. Bodlaender, H.L., van Dijk, T.C.: A cubic kernel for feedback vertex set and loop cutset. Theory Comput. Syst. 46(3), 566–597 (2010)
4. Bousquet, N., Daligault, J., Thomassé, S.: Multicut is FPT. In: Proc. of STOC 2011 (to appear, 2011)
5. Burrage, K., Estivill-Castro, V., Fellows, M.R., Langston, M.A., Mac, S., Rosamond, F.A.: The undirected feedback vertex set problem has a poly(k) kernel. In: Bodlaender, H.L., Langston, M.A. (eds.) IWPEC 2006. LNCS, vol. 4169, pp. 192–202. Springer, Heidelberg (2006)
6. Cao, Y., Chen, J., Liu, Y.: On feedback vertex set new measure and new structures. In: Kaplan, H. (ed.) SWAT 2010. LNCS, vol. 6139, pp. 93–104. Springer, Heidelberg (2010)
7. Chen, J., Fomin, F.V., Liu, Y., Lu, S., Villanger, Y.: Improved algorithms for feedback vertex set problems. J. Comput. Syst. Sci. 74(7), 1188–1198 (2008)
8. Chen, J., Liu, Y., Lu, S., O'Sullivan, B., Razgon, I.: A fixed-parameter algorithm for the directed feedback vertex set problem. In: Proc. of STOC 2008, pp. 177–186 (2008)
9. Cygan, M., Nederlof, J., Pilipczuk, M., Pilipczuk, M., van Rooij, J.M.M., Wojtaszczyk, J.O.: Solving connectivity problems parameterized by treewidth in single exponential time. CoRR abs/1103.0534 (2011)
10. Dehne, F.K.H.A., Fellows, M.R., Langston, M.A., Rosamond, F.A., Stevens, K.: An $O(2^{o(k)}n^3)$ FPT algorithm for the undirected feedback vertex set problem. In: Wang, L. (ed.) COCOON 2005. LNCS, vol. 3595, pp. 859–869. Springer, Heidelberg (2005)
11. Demaine, E.D., Hajiaghayi, M.T., Marx, D.: Open problems from dagstuhl seminar 09511 (2009)
12. Downey, R.G., Fellows, M.R.: Fixed parameter tractability and completeness. In: Complexity Theory: Current Research, pp. 191–225 (1992)
13. Downey, R.G., Fellows, M.R.: Parameterized Complexity. Springer, Heidelberg (1999)
14. Even, G., Naor, J., Schieber, B., Sudan, M.: Approximating minimum feedback sets and multicuts in directed graphs. Algorithmica 20(2), 151–174 (1998)
15. Even, G., Naor, J., Schieber, B., Zosin, L.: Approximating minimum subset feedback sets in undirected graphs with applications. SIAM J. Discrete Math. 13(2), 255–267 (2000)
16. Even, G., Naor, J., Zosin, L.: An 8-approximation algorithm for the subset feedback vertex set problem. SIAM J. Comput. 30(4), 1231–1252 (2000)
17. Guillemot, S.: FPT algorithms for path-transversals and cycle-transversals problems in graphs. In: Grohe, M., Niedermeier, R. (eds.) IWPEC 2008. LNCS, vol. 5018, pp. 129–140. Springer, Heidelberg (2008)
18. Guo, J., Gramm, J., Hüffner, F., Niedermeier, R., Wernicke, S.: Compression-based fixed-parameter algorithms for feedback vertex set and edge bipartization. J. Comput. Syst. Sci. 72(8), 1386–1396 (2006)
19. Kanj, I.A., Pelsmajer, M.J., Schaefer, M.: Parameterized algorithms for feedback vertex set. In: Downey, R.G., Fellows, M.R., Dehne, F. (eds.) IWPEC 2004. LNCS, vol. 3162, pp. 235–247. Springer, Heidelberg (2004)
20. Kawarabayashi, K., Kobayashi, Y.: Fixed-parameter tractability for the subset feedback set problem and the S-cycle packing problem (2010) (manuscript)
21. Marx, D.: Parameterized graph separation problems. Theor. Comput. Sci. 351(3), 394–406 (2006)

22. Marx, D., Razgon, I.: Fixed-parameter tractability of multicut parameterized by the size of the cutset. In: Proc. of STOC 2011 (to appear, 2011)
23. Raman, V., Saurabh, S., Subramanian, C.R.: Faster fixed parameter tractable algorithms for undirected feedback vertex set. In: Bose, P., Morin, P. (eds.) ISAAC 2002. LNCS, vol. 2518, pp. 241–248. Springer, Heidelberg (2002)
24. Raman, V., Saurabh, S., Subramanian, C.R.: Faster fixed parameter tractable algorithms for finding feedback vertex sets. ACM Transactions on Algorithms 2(3), 403–415 (2006)
25. Reed, B.A., Smith, K., Vetta, A.: Finding odd cycle transversals. Oper. Res. Lett. 32(4), 299–301 (2004)
26. Thomassé, S.: A quadratic kernel for feedback vertex set. In: Proc. of SODA 2009, pp. 115–119 (2009)

Domination When the Stars Are Out

Danny Hermelin[1], Matthias Mnich[2,⋆],
Erik Jan van Leeuwen[3], and Gerhard J. Woeginger[4]

[1] Max-Planck-Institut für Informatik, Saarbrücken, Germany
hermelin@mpi-inf.mpg.de
[2] International Computer Science Institute, Berkeley, USA
mmnich@icsi.berkeley.edu
[3] Department of Informatics, University of Bergen, Norway
E.J.van.Leeuwen@ii.uib.no
[4] Department of Mathematics and Computer Science, TU Eindhoven, The Netherlands
gwoegi@win.tue.nl

Abstract. We algorithmize the recent structural characterization for claw-free graphs by Chudnovsky and Seymour. Building on this result, we show that Dominating Set on claw-free graphs is (i) fixed-parameter tractable and (ii) even possesses a polynomial kernel. To complement these results, we establish that Dominating Set is not fixed-parameter tractable on the slightly larger class of graphs that exclude $K_{1,4}$ as an induced subgraph. Our results provide a dichotomy for Dominating Set in $K_{1,\ell}$-free graphs and show that the problem is fixed-parameter tractable if and only if $\ell \leq 3$. Finally, we show that our algorithmization can also be used to show that the related Connected Dominating Set problem is fixed-parameter tractable on claw-free graphs.

1 Introduction

The *dominating set problem* is the problem of determining whether a given graph G has a dominating set of size at most k. (A subset $D \subseteq V(G)$ is *dominating* if every vertex in G is either contained in D or adjacent to some vertex in D.) Dominating sets play a prominent role in both algorithmics and combinatorics (see e.g. [18,19]). Since the dominating set problem is hard in its decision [21], approximation [15], and parameterized versions [9], research has focused on finding special graph classes for which the problem becomes tractable. In this paper we consider the class of claw-free graphs. A graph is *claw-free* if no vertex has three pairwise nonadjacent neighbors, i.e. if it does not contain $K_{1,3}$ as an induced subgraph. The class of claw-free graphs contains several well-studied graph classes as a special case, including line graphs, unit interval graphs, the complements of triangle-free graphs, and graphs of several polyhedra and polytopes. Throughout

⋆ Matthias Mnich has been partly supported by the Netherlands Organisation for Scientific Research (NWO), grant 639.033.043, and by a DAAD Postdoctoral Fellowship.

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 462–473, 2011.

the years, this graph class attracted much interest, and is by now the subject of hundreds of mathematical research papers and surveys [13].

In the context of algorithms, initial study was directed towards extending algorithms developed for line graphs. The foremost examples are the two independent results by Shibi [30] and Minty [25] (the latter corrected by Nakamura and Tamura [27]), which extend Edmond's classical polynomial-time algorithm for Maximum Independent Set on line graphs [11], better known as Maximum Matching, to the class of claw-free graphs. In contrast to the independent set problem, Dominating Set on line graphs, also known as Edge Dominating Set, is NP-complete [32]. Nevertheless, Fernau recently showed that this problem has an $f(k) \cdot n^{O(1)}$ time algorithm, where k is the size of the solution [16]. Whether this result extends to claw-free graphs has been left an open question.

There has also been considerable study devoted to graphs excluding $K_{1,\ell}$ for $\ell > 3$, i.e. *ℓ-claw-free graphs*. These types of graphs appear frequently when considering geometric intersection graphs of various types. For instance, unit square graphs are $K_{1,5}$-free, and unit disk graphs are $K_{1,6}$-free. Marx showed that Dominating Set is W[1]-hard on unit square graphs, implying it is hard on $K_{1,5}$-free graphs [24]. Note that the problem becomes easy on $K_{1,2}$-free graphs, since these graphs are just disjoint unions of cliques. Thus, the remaining cases left open were $K_{1,3}$-free and $K_{1,4}$-free graphs. It is important that we exclude these graphs as an *induced* subgraph, and not as a subgraph. In the latter case, Dominating Set is known to be fixed-parameter tractable [28].

Our Results. We show that Dominating Set is fixed-parameter tractable on claw-free graphs, generalizing earlier results on line graphs. We use a recent, highly nontrivial structural characterization of claw-free graphs by Chudnovsky and Seymour. This characterization shows that every claw-free graph can be built by applying certain gluing operations to certain atomic structures. Its proof is contained in a sequence of seven papers (called *Claw-free graphs I-VII*). The survey [5] gives an accessible summary of the proof.

The original proof of the Chudnovsky and Seymour characterization theorem for claw-free graphs is essentially nonalgorithmic. Thus, the main challenge in our approach was to provide an algorithmic version of this theorem (see Section 2). Using our algorithmic claw-free decompostion, we establish the following:

- Dominating Set on claw-free graphs is fixed-parameter tractable. To be precise, we show that we can decide the existence of a dominating set of size at most k in $9^k \cdot n^{O(1)}$ time (Section 3).
- Connected Dominating Set on claw-free graphs is fixed-parameter tractable and solvable in $36^k \cdot n^{O(1)}$ time (Section 3), resolving an open question in [26].
- Dominating Set on claw-free graphs has a polynomial kernel with $O(k^4)$ vertices (Section 4).

To complement our results, we show that Dominating Set is W[1]-hard on $K_{1,4}$-free graphs (see the full paper). Thus, we completely determine the parameterized complexity status of Dominating Set in $K_{1,\ell}$-free graphs for all ℓ.

Related Work. Since the announcement of the Chudnovsky-Seymour decomposition theorem for claw-free graphs, several results appeared that use it to attack problems on the class of claw-free graphs, and particularly on a subclass called quasi-line graphs (see e.g. [4,3,17]). However, these results are structural and give no algorithms to find the decomposition.

Recently and independent of our work, two proposals appeared that find an algorithmic decomposition theorem for claw-free graphs. The decomposition theorem obtained by King [22] is based on the work of Chudnovsky and Seymour and has subtle differences when compared to ours, but his algorithmic methods are completely different. The decomposition theorem by Faenza et al. [14] is not based on the Chudnovsky-Seymour structure theorem and thus is substantially different from ours. They use it to obtain a faster polynomial-time algorithm for Weighted Independent Set on claw-free graphs. Although their decomposition can potentially be used to obtain a parameterized algorithm for Dominating Set on claw-free graphs [31], it is not clear whether a polynomial kernel would follow as well. Conversely, the ideas behind our work can be adapted to give a fast polynomial-time algorithm for Weighted Independent Set on claw-free graphs.

While this paper was under submission, another paper proving that Dominating Set is fixed-parameter tractable on claw-free graphs appeared [7]. Their algorithm does not use the Chudnovsky-Seymour decomposition theorem and runs in time $2^{O(k^2)} \cdot n^{O(1)}$, compared to our $9^k \cdot n^{O(1)}$ algorithm.

2 Algorithmic View of the Structure of Claw-Free Graphs

We give algorithms to find the decomposition of claw-free graphs implied by the structural characterization of claw-free graphs by Chudnovsky and Seymour. Moreover, we show that the characterization can be significantly simplified if we assume that the size of the maximum independent set of the graph is greater than three. Due to space limitations, we only state this theorem here and defer its proof to the full paper.

To understand the structure theorem, we need to introduce a significant amount of notation. All notions and definitions are essentially the same as in [6].

We work with a more general type of graph, a so-called trigraph. A *trigraph* is a graph with a distinguished subset of edges, that are called *semi-edges* and form a matching. Two vertices are called *semiadjacent* if there is a semi-edge between them, *strongly adjacent* if there is an edge between them that is not a semi-edge, and *strongly antiadjacent* if there is no edge between them. A graph is then simply a trigraph that has no semi-edges. We now say that u, v are *adjacent* if u, v are either strongly adjacent or semiadjacent, and u, v are *antiadjacent* if u, v are either strongly antiadjacent or semiadjacent. In a similar manner, we can distinguish (strong) neighborhoods, completeness, cliques, simplicial vertices, etc., as well as (strong) antineighborhoods, anticompleteness, stable sets, etc. We use $\alpha(G)$ to denote the maximum size of a stable set of G.

A trigraph G is a *thickening* of a trigraph G' if for every $v \in V(G')$ there is a nonempty set $X_v \subseteq V(G)$, such that X_v is a strong clique in G for each

$v \in V(G')$, $X_u \cap X_v = \emptyset$ for all distinct $u, v \in V(G')$, and $\bigcup_{v \in V(G')} X_v = V(G)$. Moreover, if u, v are strongly adjacent in G', then X_u is strongly X_v-complete in G; if u, v are strongly antiadjacent in G', then X_u is strongly X_v-anticomplete in G; and if u, v are semiadjacent in G', then X_u is neither strongly X_v-complete nor strongly X_v-anticomplete in G.

Note that if G is a thickening of G' and G' is a thickening of G'', then G is also a thickening of G''. Also note that if a graph G is a thickening of trigraph G' and u, v are semiadjacent in G', then $|X_u| + |X_v| \geq 3$.

A strong clique X of a trigraph G is *homogeneous* if every vertex in $G \backslash X$ is either strongly complete or strongly anticomplete to X. A trigraph G admits *twins* if G has a homogeneous strong clique of size 2. A pair of strong cliques (A, B) is *homogeneous* if every vertex $v \in V(G) \backslash (A \cup B)$ is either strongly complete or strongly anticomplete to A, and is either strongly complete or strongly anticomplete to B. A homogeneous pair of cliques (A, B) is a *W-join* if A is not strongly complete nor strongly anticomplete to B, and $|A| \geq 2$ or $|B| \geq 2$. A W-join is *proper* if no member of A is strongly complete or strongly anticomplete to B and no member of B is strongly complete or strongly anticomplete to A.

Note that by thickening a trigraph to a graph, one creates twins and W-joins. Hence given a graph G, we can find a trigraph G' such that G is a thickening of G' by contracting twins into a single vertex and replacing W-joins by semi-edges.

Strips, Stripes, and Base Classes. A *strip-graph* H consists of disjoint finite sets $V(H)$ and $E(H)$, and an incidence relation between $V(H)$ and $E(H)$ (i.e. a subset of $V(H) \times E(H)$). For any $F \in E(H)$, let $\overline{F}$ denote the set of $h \in V(H)$ incident with F. A strip-graph is essentially a hypergraph, except that we allow multiple edges and empty edges.

A *strip-structure* (H, η) of a trigraph G is a strip-graph H with $E(H) \neq \emptyset$, and a function η such that for each $F \in E(H)$, $\eta(F) \in 2^{V(G)}$, and for each $h \in \overline{F}$, $\eta(F, h) \subseteq \eta(F)$, such that:

- The sets $\eta(F)$ $(F \in E(H))$ are nonempty, pairwise disjoint, and have union $V(G)$.
- For each $h \in V(H)$, the union of the sets $\eta(F, h)$ for all $F \in E(H)$ with $h \in \overline{F}$ is a strong clique of G.
- For all distinct $F_1, F_2 \in E(H)$, if $v_1 \in \eta(F_1)$ and $v_2 \in \eta(F_2)$ are adjacent in G, then there exists $h \in \overline{F_1} \cap \overline{F_2}$ such that $v_1 \in \eta(F_1, h)$ and $v_2 \in \eta(F_2, h)$.
- For each $F \in E(H)$, the strip corresponding to F is claw-free.

Here the strip corresponding to $F \in E(H)$, where $\overline{F} = \{h_1, \ldots, h_k\}$, is defined as follows. Let $z_1, \ldots, z_k$ be new vertices and let J be the trigraph obtained from $G[\eta(F)]$ by adding $z_1, \ldots, z_k$, and for each i making z_i strongly complete to $\eta(F, h_i)$ and strongly anticomplete to $J \backslash \eta(F, h_i)$. Then $(J, \{z_1, \ldots, z_k\})$ is the *strip* corresponding to F. Observe that there is a direct correspondence between h_i and z_i, and between $\eta(F, h_i)$ and $N(z_i)$. Moreover, given just a strip-structure, it is easy to reconstruct the graph it is based on. We define $\eta(h) = \bigcup_{F \mid h \in \overline{F}} \eta(F, h)$ for all $h \in V(H)$.

We discern two special types of strips. A strip (J, Z) is a *spot* if J has three vertices, say v, z_1, z_2, and v is strongly adjacent to z_1, z_2, and z_1 is strongly

antiadjacent to z_2, and $Z = \{z_1, z_2\}$. A strip (J, Z) is a *stripe* if J is a claw-free trigraph and $Z \subseteq V(J)$ is a set of strongly simplicial vertices, such that Z is strongly stable and a vertex of $V(J) \backslash Z$ is adjacent to at most one vertex of Z.

We say that a stripe (J, Z) is a *thickening* of a stripe (J', Z') if J is a thickening of J' with sets X_v $(v \in V(J'))$, such that $|X_z| = 1$ for each $z \in Z'$ and $Z = \bigcup_{z \in Z'} X_z$.

The structural characterization shows that claw-free graphs can be decomposed into several base classes. Below we define these classes.

An arc is a connected subset of the sphere $\mathbb{S}_1$. A *circular-arc graph* is the intersection graph of a set $\mathcal{I}$ of arcs. If the arcs do not cover $\mathbb{S}_1$, this is an *interval graph*. A circular-arc graph is *proper* if no arc of $\mathcal{I}$ contains another arc of $\mathcal{I}$.

A *line trigraph* G of some graph H is a trigraph where $V(G) = E(H)$ and $e, f \in E(H)$ are adjacent in G if and only if e and f share an endpoint in H. Moreover, e, f are strongly adjacent if e and f share an endpoint of degree at least three.

Let G be the trigraph with $V(G) = \{v_1, \ldots, v_{13}\}$ such that $\{v_1, \ldots, v_6\}$ is an induced cycle; v_7 is strongly adjacent to v_1 and v_2; v_8 is strongly adjacent to v_4, v_5, and possibly adjacent to v_7; v_9 is strongly adjacent to v_1, v_2, v_3, and v_6; v_{10} is strongly adjacent to v_3, v_4, v_5, and v_6, and adjacent to v_9; v_{11} is strongly adjacent to v_1, v_3, v_4, v_6, v_9, and v_{10}; v_{12} is strongly adjacent to v_2, v_3, v_5, v_6, v_9, and v_{10}; v_{13} is strongly adjacent to v_1, v_2, v_4, v_5, v_7, and v_8. Then $G \backslash X$ for any $X \subseteq \{v_7, v_{11}, v_{12}, v_{13}\}$ is an *XX-trigraph*.

The following trigraphs are called *near-antiprismatic* trigraphs. Let G be a trigraph that is the disjoint union of three n-vertex strong cliques A, B, C for $n \geq 2$ and two vertices a_0, b_0. Let $X \subseteq A \cup B \cup C$ with $|C \backslash X| \geq 2$ and let the graph have the following adjacencies. For $1 \leq i, j \leq n$, a_i and b_j are adjacent if and only if $i = j$, and c_i is antiadjacent to a_j and b_j if and only if $i = j$. All (anti-)adjacencies are strong, except that possibly a_i is semiadjacent to b_i for at most one value of $i \in \{1, \ldots, n\}$, and if so then $c_i \in X$; a_i is semiadjacent to c_i for at most one value of $i \in \{1, \ldots, n\}$, and if so then $b_i \in X$; b_i is semiadjacent to c_i for at most one value of $i \in \{1, \ldots, n\}$, and if so then $a_i \in X$. Moreover, a_0 is strongly complete to A, b_0 is strongly complete to B, and a_0 is antiadjacent to b_0. Then $G \backslash X$ is near-antiprismatic.

We also define certain special stripes. Let J be near-antiprismatic, let a_0, b_0 be as in the above definition, with a_0, b_0 strongly antiadjacent, and let $Z = \{a_0, b_0\}$. The class of all such stripes (J, Z) is denoted by $\mathcal{Z}_2$.

Let H be a graph and let $h_1, \ldots, h_5$ be a path in H, such that h_1 and h_5 have degree one and every edge of H is incident with one of h_2, h_3, h_4. Let J be obtained from a line trigraph of H by making the vertices corresponding to edges h_2h_3 and h_3h_4 either semiadjacent or strongly antiadjacent, and let $Z = \{h_1h_2, h_4h_5\}$. The class of all such stripes (J, Z) is denoted by $\mathcal{Z}_3$.

Let J be the trigraph with the vertex set $\{a_0, a_1, a_2, b_0, b_1, b_2, b_3, c_1, c_2\}$ such that $\{a_0, a_1, a_2\}$, $\{b_0, b_1, b_2, b_3\}$, $\{a_2, c_1, c_2\}$, and $\{a_1, b_1, c_2\}$ are strong cliques,

b_2, c_1 are strongly adjacent, b_2, c_2 are semiadjacent, and b_3, c_1 are semiadjacent. Then let $Z = \{a_0, b_0\}$. The class of all such stripes (J, Z) is denoted by $\mathcal{Z}_4$.

Let J be an XX-trigraph, let $v_1, \ldots, v_{13}, X$ be as in the definition of XX-trigraphs, let v_7, v_8 be strongly antiadjacent in J, and let $Z = \{v_7, v_8\} \backslash X$. The class of all such stripes (J, Z) is denoted by $\mathcal{Z}_5$.

Algorithmic Structure Theorem. Our main result is the following algorithmic structure theorem of claw-free graphs. The proof is given in the full paper.

Theorem 1. *Let G be a connected claw-free graph, such that G does not admit twins or proper W-joins and $\alpha(G) > 3$. Then either*

- *G is a thickening of an XX-trigraph, or G is a proper circular-arc graph, or*
- *G admits a strip-structure such that each strip (J, Z) either is a spot or is a stripe with $1 \leq |Z| \leq 2$ for which either*
  - *J is a proper circular-arc graph and $|Z| = 1$,*
  - *J is a proper interval graph and $|Z| = 2$,*
  - *(J, Z) is a thickening of a member of $\mathcal{Z}_2 \cup \mathcal{Z}_3 \cup \mathcal{Z}_4 \cup \mathcal{Z}_5$, or*
  - *$|Z| = 1$, $\alpha(J) \leq 3$, and $J \backslash N[Z] \neq \emptyset$.*

We can distinguish the cases and find the strip-structure in polynomial time.

3 Application to Parameterized Algorithms

Using Theorem 1, we show that Dominating Set and Connected Dominating Set on claw-free graphs are fixed-parameter tractable. Due to space limitations, we defer the proof for Connected Dominating Set to the full paper.

Let $\gamma(G)$ denote the minimum size of a dominating set of G. More generally, let $\gamma(G, A)$, where $A \subseteq V(G)$, denote the size of a smallest subset of $V(G)$ dominating all vertices in $V(G) \backslash A$. We implicitly use the following result of Allan and Laskar [1]. Let $i(G)$ denote the minimum size of an independent dominating set of G, that is, of any subset of $V(G)$ that is both an independent set and a dominating set of G.

Theorem 2 ([1]). *If G is a claw-free graph, then $i(G) = \gamma(G)$.*

Allan and Laskar also give an algorithm to turn any dominating set into an independent dominating set of the same or smaller size. We can thus assume throughout w.l.o.g. that any (minimum) dominating set that we consider is also an independent set.

The idea of how to establish the fixed-parameter tractability of Dominating Set on claw-free graphs is as follows. We first show that we can remove twins and proper W-joins from G without changing the size of its minimum dominating set. If $\alpha(G) \leq 3$, then $\gamma(G) \leq 3$, and we can find a minimum dominating set by exhaustive enumeration. If $\alpha(G) > 3$, we can apply Theorem 1. Then G either belongs to a basic class, or it can be decomposed into strips. If G is in a basic class, we can easily find a minimum dominating set. If G can be decomposed into strips, we solve Minimum Dominating Set separately on each strip. We then use a parameterized algorithm to stitch the solutions of the strips together.

Easy Cases and Further Structure. We first show that we can remove twins and (proper) W-joins from a graph without changing the size of its minimum dominating set. Moreover, the reductions preserve claw-freeness.

Lemma 1. *Let a, b be twins of a graph G. Then $\gamma(G) = \gamma(G - a)$.*

Lemma 2. *Let (A, B) be a W-join of a graph G and let $X \subseteq V(G)$ be such that $A \cap X \neq \emptyset$ implies $A \subseteq X$ and the same for B. Then we can find in polynomial time sets $A' \subset A$ and $B' \subset B$ with $|A \backslash A'| + |B \backslash B'| \leq 3$, such that the graph G' obtained by removing A' and B' from G, and possibly removing all remaining edges between $A \backslash A'$ and $B \backslash B'$, satisfies $\gamma(G, X) = \gamma(G', X \cap V(G'))$.*

The lemma implies that we can remove proper W-joins. Hence we can use the structure theorem. We now show that if a graph G is in a 'basic class', a minimum dominating set can be computed in polynomial time.

Theorem 3 ([20]). *Let G be a circular-arc graph. Then $\gamma(G)$ can be computed in linear time.*

Lemma 3. *Let G be a graph that is a thickening of an XX-trigraph. Then $\gamma(G)$ can be computed in polynomial time.*

Given a strip-structure of a graph, we want to compute a minimum dominating set efficiently for all its strips and be able to stitch these solutions together in an optimal fashion. To this end, we need to parameterize the minimum dominating set of a strip (J, Z) by what a minimum dominating set D of G would look like relative to this strip. Let F be the edge of the strip structure corresponding to (J, Z). Then for each $h \in \overline{F}$ corresponding to some $z \in Z$, there are three possible cases, depending on whether or not $\eta(h)$ or $\eta(F, h)$ contains a vertex of D or not. We model this by considering two disjoint sets $X, Y \subseteq Z$. We put $z \in Z$ into X to model the situation where $\eta(h) \cap D = \emptyset$, and we put z into Y to model the situation where $(\eta(h) \backslash \eta(F, h)) \cap D \neq \emptyset$. Then, for any class of strips we have, we have to show how to compute $\gamma(J \backslash (X \cup Y), N[Y])$ efficiently for any disjoint $X, Y \subseteq Z$. Observe that the case when $z \in Z$ is neither in X nor in Y correctly models the case when $\eta(F, h) \cap D \neq \emptyset$, because, since z is simplicial, we can always assume that any dominating set of J contains a neighbor of z instead of z.

Lemma 4. *Let (J, Z) be a stripe of any of the types mentioned in Theorem 1. Then for any disjoint $X, Y \subseteq Z$, $\gamma(J \backslash (X \cup Y), N[Y])$ can be computed in polynomial time.*

We now investigate the strip-structure given by Theorem 1 in relation to the minimum dominating set problem. It follows from the strip-structure that we can see H as a multigraph with loops. The loops are precisely those $F \in E(G)$ for which $|\overline{F}| = 1$. We bicolor the edges of H as follows. Color an edge $F \in E(H)$ black if the strip (J, Z) corresponding to F satisfies that $V(J) \backslash N[Z] \neq \emptyset$, or that $V(J)$ is a union of two strong cliques, $|V(J)| \geq 4$, and $|Z| = 2$. All other edges are colored white. We can observe from Theorem 1 that strips corresponding to white edges of H are either spots or have exactly one edge and two vertices.

Lemma 5. *If $\gamma(G) \leq k$, then the subgraph H_B of H induced by the black edges has at most $2k$ vertices.*

Lemma 6. *Let D be a dominating set of G with $|D| \leq k$. Let H' denote the graph obtained from H by removing all black edges and removing the set of all k' vertices $h \in V(H)$ incident to a black edge $F \in E(H)$ for which $\eta(F,h) \cap D \neq \emptyset$. Then D induces a vertex cover of H' of size at most $2(k-k')$.*

FPT Algorithm. We show how to stitch the results of the stripes together. Our approach extends ideas of Fernau [16] for parameterized Edge Dominating Set.

Theorem 4. *Let G be a claw-free graph and $k \geq 0$ an integer. Then we can decide in $9^k \cdot n^{O(1)}$ time whether $\gamma(G) \leq k$.*

Proof. Following Lemma 1 and 2, we can assume that G does not admit twins or proper W-joins. Note that all twins in a graph can be found in linear time, while proper W-joins can be found in $O(n^2m)$ time [23]. If $\alpha(G) \leq 3$, then $\gamma(G) \leq 3$, which we can decide in polynomial time by exhaustive enumeration. Hence we can apply Theorem 1. Consider the various cases. If G is a proper circular-arc graph or if G is a thickening of an XX-trigraph, then $\gamma(G)$ can be computed in polynomial time by Theorem 3 and Lemma 3.

Otherwise, consider the strip-structure (H,η) found. Let D be a minimum dominating set of G. Then for any $F \in E(H)$, the way $\eta(F)$ is dominated is determined by $D \cap (\bigcup_{h \in \overline{F}} \eta(h))$. Lemmas 5 and 6 suggest the following approach to guessing this information. Let S_1 be any subset of the vertices of H incident to a black edge. Remove S_1 and all black edges from H, and call the remaining graph H'. Let S_2 be a minimal vertex cover of H' of size at most $2k - |S_1|$. Let $S = S_1 \cup S_2$. For each such set S, we will determine a dominating set D such that $D \cap \eta(h) \neq \emptyset$ for each $h \in S$. To this end, we construct an auxiliary multigraph G' with vertices v_h for each $h \in S$, a weight function w on the edges of G', and an integer k'. The idea is that k' is the number of vertices that any dominating set D of G must have if $D \cap \eta(h) \neq \emptyset$ for each $h \in S$. Then we use the multigraph and its associated edge weight function to decide which strips should be made responsible for ensuring that $D \cap \eta(h) \neq \emptyset$ for each $h \in S$, while minimizing $|D|$. Initially, G' consists of the vertices v_h and no edges, and $k' = 0$.

Consider some edge $F \in E(H)$ and let (J,Z) be the strip corresponding to F. Suppose that $\overline{F} = \{h\}$ for some $h \in V(H)$, i.e. that $|Z| = 1$. Note that (J,Z) must be a stripe, as spots have $|Z| = 2$. If $h \notin S$, then the strip is itself responsible for dominating all vertices in $\eta(F)$. So add $\gamma(J \backslash Z)$ to k'. Otherwise, i.e. if $h \in S$, then some vertex of $\eta(h)$ will be in the dominating set and it could potentially also be in $\eta(F)$. So add a vertex v_F to G', and add an edge e_F between v_F and v_h to G' with weight $\gamma(J) - \gamma(J \backslash Z, N[Z])$. This models the additional cost of having a vertex of $\eta(F,h)$ in the dominating set. Since $N[Z]$ is a strong clique, any dominating set for J can be assumed to have a vertex in $N(Z)$, i.e. in $\eta(F,h)$. Finally, add $\gamma(J \backslash Z, N[Z])$ to k'.

Suppose that $\overline{F} = \{h,h'\}$ for distinct $h,h' \in V(H)$, i.e. that $|Z| = 2$. If (J,Z) is a spot, then $\overline{F} \cap S \neq \emptyset$, or the vertex in $\eta(F)$ will never be dominated. Add

an edge e_F to G'. If $h, h' \in S$, then e_F runs between v_h and $v_{h'}$. If only one of h and h' belongs to S, say $h \in S$, add a new vertex v_F to G' and let e_F run between v_h and v_F. The weight of e_F is 1.

If (J, Z) is a stripe, let $Z = \{z, z'\}$, where z corresponds to h and z' to h'. If $\overline{F} \cap S = \emptyset$, add $\gamma(J \backslash Z)$ to k'. If $\overline{F} \cap S = \{h\}$, then add a vertex v_F to G' and an edge e_F with weight $\gamma(J \backslash \{z'\}) - \gamma(J \backslash Z, N[z])$ between v_F and v_h. Add $\gamma(J \backslash Z, N[z])$ to k'. The cases when $\overline{F} \cap S = \{h'\}$ or $\overline{F} \cap S = \{h, h'\}$ are similar.

Observe that each edge e added to G' for some $F \in E(H)$ in the above construction corresponds to a particular way to ensure that a dominating set of the strip F has a vertex in $\eta(F, h)$, where e is incident to v_h in G'. The weight on the edge corresponds to the number of extra vertices it would cost to ensure this, compared to the cost of having no vertex in $\eta(F, h)$. Therefore we want to find a subset of the edges of minimum total weight that covers all vertices v_h of G'. This clearly corresponds to a smallest dominating set D of G for which $D \cap \eta(h) \neq \emptyset$ for each $h \in S$. Finding this subset takes cubic time via a minimum generalized weighted edge cover [29,16].

The above algorithm yields a smallest dominating set D with $D \cap \eta(h) \neq \emptyset$ for each $h \in S$. Repeat this procedure for all $S = S_1 \cup S_2$. The number of possible choices for S_1 is at most $\sum_{i=0}^{2k} \binom{2k}{i}$. For each set S_1, we spend $2^{2k-|S_1|} \cdot n^{O(1)}$ time to enumerate all minimal vertex covers [8]. As $\sum_{i=0}^{2k} \binom{2k}{i} 2^{2k-i} = 9^k$ by the binomial theorem, the algorithm decides whether $\gamma(G) \leq k$ in $9^k \cdot n^{O(1)}$ time. □

4 Polynomial Kernel for Dominating Set

We show that Dominating Set has a polynomial kernel on claw-free graphs. A *kernelization algorithm* for a parameterized problem Π computes in polynomial time, given an instance (x, k) of Π, a new instance (x', k') of Π such that $(x', k') \in \Pi$ if and only if $(x, k) \in \Pi$, and $|x'| \leq f(k)$ for some computable function f. The instance (x', k') is called a *kernel* of Π, and it is called a *polynomial kernel* if f is a polynomial. Not every fixed-parameter tractable problem admits a polynomial kernel, or the polynomial hierarchy collapses to the third level [2].

The basic idea of our kernel is to replace each stripe of the strip-structure given in Theorem 1 by a stripe of size at most linear in k. We then reduce the strip-structure itself to have a polynomial number of vertices and edges.

We first consider stripes (J, Z) with $|Z| = 2$. We need to show that if a stripe (J, Z) is a thickening of a member of $\mathcal{Z}_2 \cup \mathcal{Z}_3 \cup \mathcal{Z}_4 \cup \mathcal{Z}_5$, we can distinguish of a member of which class (J, Z) is a thickening of. We provide recognition algorithms for each class in the full paper. Then we can reduce (J, Z).

Lemma 7. *Let (J, Z) be a stripe of any of the types mentioned in Theorem 1 with $|Z| = 2$ such that J is a graph, J does not admit twins, and $V(J)$ is not a union of two strong cliques. Then we can find a claw-free stripe (J', Z) with $|V(J')| \leq 18k + 33$, such that $\gamma(J \backslash (X \cup Y), N(Y)) = \gamma(J' \backslash (X \cup Y), N(Y))$ for any disjoint $X, Y \subseteq Z$.*

For stripes (J, Z) with $|Z| = 1$, we can follow a simpler approach.

Lemma 8. *Let (J, Z) be a thickening of an almost-unbreakable stripe such that $|Z| = 1$ and J is a graph. Then we can find in polynomial time a claw-free stripe (J', Z') and an integer $k' \geq 0$, such that $\gamma(J \backslash Z, N[Z]) = \gamma(J' \backslash Z', N[Z']) + k'$, $\gamma(J \backslash Z) = \gamma(J' \backslash Z') + k'$, $\gamma(J) = \gamma(J') + k'$, $|Z'| = 1$, and $|V(J')| \leq 4$.*

We now again consider the strip-structure as a multigraph with loops and color its edges black or white as before. Consider the subgraph H_W of H induced by the white edges and flatten it by removing all parallel edges and replacing loops by pendant vertices. Then H_W has a vertex cover of size at most $2k$ by Lemma 6 if $\gamma(G) \leq k$. Now kernelize H_W using the Buss rule. Repeatedly do the following until no longer possible: Remove a vertex of degree 0 and remove a vertex v of degree greater than $2k$ in H_W and add v to a set M. It is clear that the remaining graph H'_W must have at most $(2k)^2 + 2k$ vertices, or we can answer NO. Moreover, $|M| \leq 2k$, or we can answer NO.

Let H_B denote the subgraph of H induced by black edges. From Lemma 5 it follows that $|V(H_B)| \leq 2k$, or we can answer NO. Now consider the subgraph of H induced by $V(H'_W)$, M, and $V(H_B)$. To each vertex $h \in M$, we add a loop F with $\overline{F} = \{h\}$, and set $\eta(F) = \eta(F, h) = v_h^*$ for some new vertex v_h^*. Call the resulting graph H_k. It is clear that $|V(H_k)| \leq 6k + 4k^2$.

Lemma 9. *Let G_k be the subgraph of G induced by H_k. Then $\gamma(G) \leq k$ if and only if $\gamma(G_k) \leq k$.*

Theorem 5. *Dominating Set on claw-free graphs has an $O(k^4)$ kernel.*

Proof. By eliminating twins and proper W-joins (Lemma 1 and 2), and solving base cases as in Theorem 4, it follows from Theorem 1 that we find a nontrivial strip-structure (H, η). It follows from Lemma 9 that we can either return a trivial NO-instance, or reduce (H, η) to a strip-structure (H_k, η_k) with $|V(H_k)| = O(k^2)$. Let G_k be the graph induced by (H_k, η_k). Then $\gamma(G) \leq k$ if and only if $\gamma(G_k) \leq k$ by Lemma 9. Abusing notation, let $H = H_k$ and $\eta = \eta_k$. We now consider different types of strips (J, Z) and bound their number and size.

Suppose that $|Z| = 1$. Consider all strips for which additionally $|V(J)| = 2$. Note that each $h \in V(H)$ is incident to at most one $F \in E(H)$ that corresponds to such a strip, for any two would imply the existence of twins in G. Hence we can limit the number of such strips by $2k$. If $|V(J)| > 2$, then as G does not admit twins, $J \backslash N[Z] \neq \emptyset$. Hence any dominating set of G must have at least one vertex in $J \backslash Z$. Following Theorem 1, either J is a proper circular-arc graph, or $\alpha(J) \leq 3$, or (J, Z) is a thickening of a member of $\mathcal{Z}_5$. In all three cases, we can compute $\gamma(J \backslash (X \cup Y), N(Y))$ for any disjoint $X, Y \subseteq Z$ in polynomial time following Theorem 3 and Lemma 4. We then apply Lemma 8 to replace (J, Z) by a stripe (J', Z') of constant size. Since each dominating set of G must have at least one vertex in $J \backslash Z$, the number of these stripes is at most k.

So assume that $|Z| = 2$. Consider all stripes between h and h', for certain $h, h' \in V(H)$, that are a union of two strong cliques. Suppose that there are j such stripes and denote the two cliques of the i-th stripe by A_i and B_i. We can assume that $A_i \subseteq \eta(h)$ and $B_i \subseteq \eta(h')$. Hence $A = \bigcup_i A_i$ is a strong clique

and so is $B = \bigcup_i B_i$. But then we can actually view all these stripes as a single stripe that is the union of the two strong cliques A, B and assume that $j \leq 1$. (Alternatively, one can reduce (A, B) to form a proper W-join in G if $j \geq 2$.) If $j = 1$, we can show that this stripe is a thickening of a stripe with four vertices and one semiadjacent pair of vertices. Then we use Lemma 2 to reduce this stripe to a claw-free stripe (J', Z') with $|V(J')| \leq 5$ and $|Z'| = 2$. Since these stripes each correspond to a black edge of H, the total number of such stripes in H is at most $\binom{2k}{2}$ by Lemma 5.

Observe that any two spots incident to the same $h, h' \in V(H)$ form twins. But then H contains at most $\binom{6k+4k^2}{2}$ spots. They already have constant size.

We conclude from Theorem 1 that the strips not considered thus far have $J \backslash N[Z] \neq \emptyset$ and that $J \backslash Z$ thus must contain a vertex of any dominating set of G. Moreover, J is a proper interval graph or (J, Z) is a thickening of a member of $Z_2 \cup Z_3 \cup Z_4 \cup Z_5$. Then it follows that we can 'kernelize' each such strip to have size $O(k)$, following Lemma 7. Since $J \backslash Z$ must contain a vertex of any dominating set of G, there are at most k such strips.

Consider the strip structure (H', η') and strips as constructed above. Let G' be the graph induced by these strips. Then G' is claw-free, has $O(k^4)$ vertices, and has a dominating set of size k if and only if G has one. □

References

1. Allan, R.B., Laskar, R.: On Domination and Independent Domination Numbers of a Graph. Discrete Mathematics 23, 73–76 (1978)
2. Bodlaender, H.L., Downey, R.G., Fellows, M.R., Hermelin, D.: On problems without polynomial kernels. Journal of Computer and System Sciences 75(8), 423–434 (2009)
3. Chudnovsky, M., Fradkin, A.O.: An approximate version of Hadwiger's conjecture for claw-free graphs. Journal of Graph Theory 63(4), 259–278 (2010)
4. Chudnovsky, M., Ovetsky, A.: Coloring Quasi-Line Graphs. Journal of Graph Theory 54(1), 41–50 (2007)
5. Chudnovsky, M., Seymour, P.D.: The Structure of Claw-Free Graphs. London Mathematical Society Lecture Note Series: Surveys in Combinatorics, vol. 327, pp. 153–171 (2005)
6. Chudnovsky, M., Seymour, P.D.: Claw-free graphs. V. Global structure. Journal of Combinatorial Theory, Series B 98(6), 1373–1410 (2008)
7. Cygan, M., Philip, G., Pilipczuk, M., Pilipczuk, M., Wojtaszczyk, J.O.: Dominating Set is Fixed Parameter Tractable in Claw-free Graphs, arXiv:1011.6239v1 (cs.DS) (2010) (preprint)
8. Damaschke, P.: Parameterized Enumeration, Transversals, and Imperfect Phylogeny Reconstruction. In: Downey, R., Fellows, M., Dehne, P. (eds.) IWPEC 2004. LNCS, vol. 3162, pp. 1–12. Springer, Heidelberg (2004)
9. Downey, R.G., Fellows, M.R.: Fixed-parameter tractability and completeness. Congressus Numerantium 87, 161–178 (1992)
10. Downey, R.G., Fellows, M.R.: Parameterized Complexity. Springer, New York (1999)
11. Edmonds, J.: Paths, trees, and flowers. Canadian Journal of Mathematics 17, 449–467 (1965)

12. Eisenbrand, F., Oriolo, G., Stauffer, G., Ventura, P.: The Stable Set Polytope of Quasi-Line Graphs. Combinatorica 28(1), 45–67 (2008)
13. Faudree, R., Flandrin, E., Ryjáček, Z.: Claw-free graphs – A survey. Discrete Mathematics 164(1-3), 87–147 (1997)
14. Faenza, Y., Oriolo, G., Stauffer, G.: An algorithmic decomposition of claw-free graphs leading to an $O(n^3)$-algorithm for the weighted stable set problem. In: Proc. SODA 2011, pp. 630–646. ACM-SIAM (2011)
15. Feige, U.: A Threshold of $\ln n$ for Approximating Set Cover. Journal of the ACM 45, 634–652 (1998)
16. Fernau, H.: Edge Dominating Set: Efficient Enumeration-Based Exact Algorithms. In: Bodlaender, H.L., Langston, M.A. (eds.) IWPEC 2006. LNCS, vol. 4169, pp. 142–153. Springer, Heidelberg (2006)
17. Galluccio, A., Gentile, C., Ventura, P.: The stable set polytope of claw-free graphs with large stability number. Electronic Notes Discrete Mathematics 36, 1025–1032 (2010)
18. Haynes, T.W., Hedetniemi, S.T., Slater, P.J.: Fundamentals of domination in graphs. Marcel Dekker Inc., New York (1998)
19. Haynes, T.W., Hedetniemi, S.T., Slater, P.J.: Domination in graphs: Advanced Topics. Marcel Dekker Inc., New York (1998)
20. Hsu, W.-L., Tsai, K.-H.: Linear-time algorithms on circular arc graphs. Information Processing Letters 40, 123–129 (1991)
21. Karp, R.M.: Reducibility Among Combinatorial Problems. In: Miller, R.E., Thatcher, J.W. (eds.) Complexity of Computer Computations, pp. 85–103. Plenum, New York
22. King, A.D.: Claw-free graphs and two conjectures on ω, Δ, and χ, PhD Thesis, McGill University, Montreal, Canada (2009)
23. King, A.D., Reed, B.A.: Bounding χ in Terms of ω and Δ for Quasi-Line Graphs. Journal of Graph Theory 59(3), 215–228 (2008)
24. Marx, D.: Parameterized Complexity of Independence and Domination on Geometric Graphs. In: Bodlaender, H.L., Langston, M.A. (eds.) IWPEC 2006. LNCS, vol. 4169, pp. 154–165. Springer, Heidelberg (2006)
25. Minty, G.J.: On maximal independent sets of vertices in claw-free graphs. Journal of Combinatorial Theory, Series B 28(3), 284–304 (1980)
26. Misra, N., Philip, G., Raman, V., Saurabh, S.: The effect of girth on the kernelization complexity of Connected Dominating Set. In: Lodaya, K., Mahajan, M. (eds.) FSTTCS 2010, Schloss Dagstuhl, Daghstuhl, Germany. LIPIcs, vol. 8, pp. 96–107 (2010)
27. Nakamura, D., Tamura, A.: A revision of Minty's algorithm for finding a maximum weighted stable set of a claw-free graph. Journal of the Operations Research Society of Japan 44(2), 194–204 (2001)
28. Philip, G., Raman, V., Sikdar, S.: Solving Dominating Set in Larger Classes of Graphs: FPT Algorithms and Polynomial Kernels. In: Fiat, A., Sanders, P. (eds.) ESA 2009. LNCS, vol. 5757, pp. 694–705. Springer, Heidelberg (2009)
29. Plesník, J.: Constrained weighted matchings and edge coverings in graphs. Discrete Applied Mathematics 92(2-3), 229–241 (1999)
30. Sbihi, N.: Algorithme de recherche d'un stable de cardinalité maximum dans un graphe sans étoile. Discrete Mathematics 29(1), 53–76 (1980)
31. Stauffer, G.: Personal communication
32. Yannakakis, M., Gavril, F.: Edge dominating sets in graphs. SIAM Journal on Applied Mathematics 38(3), 364–372 (1980)

A Simple Deterministic Reduction for the Gap Minimum Distance of Code Problem*

Per Austrin[1] and Subhash Khot[2]

[1] University of Toronto
[2] New York University

Abstract. We present a simple deterministic gap-preserving reduction from SAT to the Minimum Distance of Code Problem over $\mathbb{F}_2$. We also show how to extend the reduction to work over any finite field (of constant size). Previously a randomized reduction was known due to Dumer, Micciancio, and Sudan [9], which was recently derandomized by Cheng and Wan [7, 8]. These reductions rely on highly non-trivial coding theoretic constructions whereas our reduction is *elementary*.

As an additional feature, our reduction gives a constant factor hardness even for asymptotically good codes, i.e., having constant rate and relative distance. Previously it was not known how to achieve deterministic reductions for such codes.

1 Introduction

The Minimum Distance of Code Problem over a finite field $\mathbb{F}_q$, which we denote Min Dist(q), asks for a non-zero codeword with minimum Hamming weight in a given linear code C (i.e., a linear subspace of $\mathbb{F}_q^n$). The problem was proved to be NP-hard by Vardy [16].

Dumer, Micciancio, and Sudan [9] proved that assuming RP $\neq$ NP the problem is hard to approximate within some factor $\gamma > 1$ using a *gap preserving* reduction from the Nearest Codeword Problem, denoted NCP(q) (which is known to be NP-hard even with a large gap). The latter problem asks, given a code $\tilde{C} \subseteq \mathbb{F}_q^m$ and a point $p \in \mathbb{F}_q^m$, for a codeword that is nearest to p in Hamming distance. However, Dumer et al.'s reduction is randomized: it maps an instance $(\tilde{C}, p)$ of NCP(q) to an instance C of Min Dist(q) in a randomized manner such that: in the YES Case, with high probability, the code C has a non-zero codeword with weight at most d, and in the NO Case, C has no non-zero codeword of weight less that γd, for some fixed constant $\gamma > 1$. We note that the minimum distance of code is multiplicative under the tensor product of codes; this enables one to *boost* the inapproximability result to any constant factor, or even to an *almost polynomial factor* (under a quasipolynomial time reduction), see [9].

* Research done while first author at New York University supported by NSF Expeditions grant CCF-0832795. Second author supported by NSF CAREER grant CCF-0833228, NSF Expeditions grant CCF-0832795, and BSF grant 2008059.

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 474–485, 2011.

The randomness in Dumer et al.'s reduction is used for constructing, as a gadget, a non-trivial coding theoretic construction with certain properties. In a remarkable pair of papers, Cheng and Wan [7, 8] recently constructed such a gadget deterministically, thereby giving a deterministic reduction to the Gap Min Dist(q) Problem. Cheng and Wan's construction is quite sophisticated. It is an interesting pursuit, in our opinion, to seek an *elementary* deterministic reduction for the Gap Min Dist(q) Problem.

In this paper, we indeed present such a reduction. For codes over $\mathbb{F}_2$, our reduction is (surprisingly) simple, and does not rely on any specialized gadget construction. The reduction can be extended to codes over any finite field $\mathbb{F}_q$; however, then the details of the reduction become more involved, and we need to use Viola's recent construction of a psedorandom generator for low degree polynomials [17]. Even in this case, the resulting reduction is conceptuelly quite simple.

We also observe that our reduction produces asymptotically good codes, i.e., having constant rate and relative distance. While Dumer et al. [9] are able to prove randomized hardness for such codes, this was not obtained by the deterministic reduction by Cheng and Wan. In [8], proving a constant factor hardness of approximation for asymptotically good codes is mentioned as an open problem.

Our main theorem is thus:

Theorem 1. *For any finite field $\mathbb{F}_q$, there exists a constant $\gamma > 0$ such that it is NP-hard (via a deterministic reduction) to approximate the* Min Dist(q) *problem to within a factor $1+\gamma$, even on codes with rate $\geq \gamma$ and relative distance $\geq \gamma$ (i.e., asymptotically good codes).*

Another motivation to seek a new deterministic reduction for Min Dist(q) is that it might lead to a deterministic reduction for the analogous problem for integer lattices, namely the Shortest Vector Problem (SVP). For SVP, we do not know of a deterministic reduction that proves even the basic NP-hardness, let alone a hardness of approximation result, except in the case of the ℓ_∞ norm. All known reductions are randomized [1, 6, 15, 12, 13, 11]. In fact, the reduction of Dumer et al. [9] giving hardness of approximation for Min Dist(q) assuming NP $\neq$ RP is inspired by a reduction by Micciancio [15] for SVP.

Our hope is that our new reduction for Min Dist(q) can be used to shed new light on the hardness of SVP. For instance, it might be possible to combine our reductions for Min Dist(q) for different primes q so as to give a reduction over integers, i.e., a reduction to SVP.

1.1 Organization

We present a proof of this theorem for the binary field in Section 3 and for a general finite field in Section 5. Even for the binary case, it is instructive to first see a reduction to NCP(2) in Section 3.1 which is then extended to the Min Dist(2) problem in Section 3.2.

2 Preliminaries

2.1 Codes

Let q be a prime power.

Definition 1. *A linear code C over a field $\mathbb{F}_q$ is a linear subspace of $\mathbb{F}_q^n$, where n is the block-length of the code and dimension of the subspace C is the dimension of the code. The distance of the code $d(C)$ is the minimum Hamming weight of any non-zero vector in C.*

The two problems Min Dist(q) and NCP(q) are defined as follows.

Definition 2. Min Dist(q) *is the problem of determining the distance $d(C)$ of a linear code $C \subseteq \mathbb{F}_q^n$. The code may be given by the basis vectors for the subspace C or by the linear forms defining the subspace.*

Definition 3. NCP(q) *is the problem of determining the minimum distance from a given point $p \in \mathbb{F}_q^n$ to any codeword in a given code $C \subseteq \mathbb{F}_q^n$. Equivalently, it is the problem of determining the minimum Hamming weight of any point z in a given affine subspace of $\mathbb{F}_q^n$ (which would be $C - p$).*

Our reduction uses tensor products of codes, which are defined as follows.

Definition 4. *Let $C_1, C_2 \subseteq \mathbb{F}_q^n$ be linear codes. Then the linear code $C_1 \otimes C_2 \subseteq \mathbb{F}_q^{n^2}$ is defined as the set of all $n \times n$ matrices over $\mathbb{F}_q$ such that each of its columns is a codeword in C_1 and each of its rows is a codeword in C_2.*

A well-known fact is that the distance of a code is multiplicative under the tensor product of codes.

Fact 2 *Let $C_1, C_2 \subseteq \mathbb{F}_q^n$ be linear codes. Then the linear code $C_1 \otimes C_2 \subseteq \mathbb{F}_q^{n^2}$ has distance $d(C_1 \otimes C_2) = d(C_1)d(C_2)$.*

We shall need the following Lemma which shows that for many codewords of $C \otimes C$ one can obtain a stronger bound on the distance than the bound $d(C)^2$ given by Fact 2. A proof can be found in the full version [4].

Lemma 1. *Let $C \subseteq \mathbb{F}_q^n$ be a linear code of distance $d = d(C)$, and let $Y \in C \otimes C$ be a non-zero codeword with the additional properties that*

1. *The diagonal of Y is zero.*
2. *Y is symmetric.*

Then Y has at least $d^2(1 + 1/q)$ non-zero entries.

2.2 Hardness of Constraint Satisfaction

The starting point in our reduction is a constraint satisfaction problem that we refer to as the Max NAND problem, defined as follows.

Definition 5. *An instance Ψ of the* Max NAND *problem consists of a set of quadratic equations over $\mathbb{F}_2$, each of the form $x_k = NAND(x_i, x_j) = 1 + x_i \cdot x_j$ for some variables x_i, x_j, x_k. The objective is to find an assignment to the variables such that as many equations as possible are satisfied. We denote by $\mathsf{Opt}(\Psi) \in [0,1]$ the maximum fraction of satisfied equations over all possible assignments to the variables.*

The following is an easy consequence of the PCP Theorem [10, 3, 2] and the fact that NAND gates form a basis for the space of boolean functions.

Theorem 3. *There is a universal constant $\delta > 0$ such that given a* Max NAND *instance Ψ it is NP-hard to determine whether $\mathsf{Opt}(\Psi) = 1$ or $\mathsf{Opt}(\Psi) \leq 1 - \delta$.*

3 The Binary Case

In this section we give a simple reduction from Max NAND showing that it is NP-hard to approximate Min Dist(2) to within some constant factor.

3.1 Reduction to Nearest Codeword

It is instructive to start with a reduction for the Nearest Codeword Problem, NCP(2), for which it is significantly easier to prove hardness. There are even simpler reductions known than the one we give here, but as we shall see in the next section this reduction can be modified to give hardness for the Min Dist(2) problem.

Given a Max NAND instance Ψ with n variables and m constraints, we shall construct an affine subspace $\mathcal{S}$ of $\mathbb{F}_2^{4m}$ such that:

(i) If Ψ is satisfiable then $\mathcal{S}$ has a vector of weight at most m.
(ii) If $\mathsf{Opt}(\Psi) \leq 1 - 2\delta$ then $\mathcal{S}$ has no vector of weight less than $(1 + 2\delta)m$.

This proves, according to Definition 3, that NCP(2) is NP-hard to approximate within a factor $1 + 2\delta$.

Every constraint $x_k = 1 + x_i x_j$ in Ψ gives rise to four new variables, as follows. We think of the four variables as a function $S_{ijk} : \mathbb{F}_2^2 \to \mathbb{F}_2$. The intent is that this function should be the indicator function of the values of x_i and x_j, in other words, that

$$S_{ijk}(a,b) = \begin{cases} 1 \text{ if } x_i = a \text{ and } x_j = b \\ 0 \text{ otherwise} \end{cases}.$$

With this interpretation in mind, each function S_{ijk} has to satisfy the following linear constraints over $\mathbb{F}_2$:

$$S_{ijk}(0,0) + S_{ijk}(0,1) + S_{ijk}(1,0) + S_{ijk}(1,1) = 1 \quad (1)$$
$$S_{ijk}(1,0) + S_{ijk}(1,1) = x_i \quad (2)$$
$$S_{ijk}(0,1) + S_{ijk}(1,1) = x_j \quad (3)$$
$$S_{ijk}(0,0) + S_{ijk}(0,1) + S_{ijk}(1,0) = x_k. \quad (4)$$

Thus, we have a set of $n + 4m$ variables $z_1, \ldots, z_{n+4m}$ (recall that n and m are the number of variables and constraints of Ψ, respectively) and $4m$ linear constraints of the form $\sum l_{ij} z_j = b_i$ where $l_i \in \mathbb{F}_2^{n+4m}$ and $b_i \in \mathbb{F}_2$.

Let $\mathcal{S} \subseteq \mathbb{F}_2^{4m}$ be the affine subspace of $\mathbb{F}_2^{4m}$ defined by the set of solutions to the system of equations, projected to the $4m$ coordinates corresponding to the S_{ijk} variables. Note that these coordinates uniquely determine the remaining n coordinates (assuming without loss of generality that every variable of Ψ appears in some constraint), according to Equations (2)-(4).

Now, if Ψ is satisfiable, then using the satisfying assignment for x and the intended values for the S_{ijk}'s we obtain an element of $\mathcal{S}$ with m non-zero entries. Note that for each constraint involving variables x_i, x_j, x_k, exactly one of the four variables $S_{ijk}(\cdot, \cdot)$ is non-zero.

On the other hand, note that if the function $S_{ijk}(\cdot, \cdot)$ has exactly one non-zero entry it must be that the induced values of (x_i, x_j, x_k) satisfy the constraint $x_k = 1 + x_i \cdot x_j$ (which one can see either by trying all such S_{ijk} or noting that each of the four different satisfying assignments to (x_i, x_j, x_k) gives a unique such S_{ijk}). Since every S_{ijk} is constrained to have an odd number of non-zero entries by Equation (1), it means that whenever S_{ijk} induces values of (x_i, x_j, x_k) that do not satisfy $x_k = 1 + x_i \cdot x_j$, it must hold that S_{ijk} has three non-zero entries. Therefore, we see that if $\mathsf{Opt}(\Psi) \leq 1 - \delta$, it must hold that every element of $\mathcal{S}$ has at least $(1 + 2\delta)m$ non-zero entries.

To summarize, we obtain that it is NP-hard to approximate the minimum weight element of an affine subspace (or equivalently, the Nearest Codeword Problem) to within a constant factor $1 + 2\delta$.

3.2 Reduction to Minimum Distance

To get the hardness result for the Min Dist problem, we would like to alter the reduction in the previous section so that it produces a linear subspace rather than an affine one. The only non-homogenous part of the subspace produced are the equations (1) constraining each S_{ijk} to have an odd number of entries. To produce a linear subspace, we are going to replace the constant 1 with a variable x_0, which is intended to take the value 1. In other words, we replace Equation (1) with the following equation:

$$S_{ijk}(0,0) + S_{ijk}(0,1) + S_{ijk}(1,0) + S_{ijk}(1,1) = x_0 \tag{1'}$$

However, in order to make this work we need to ensure that every assignment where x_0 is set to 0 has large weight, and this requires adding some more components to the reduction.

We first observe that the system of constraints relating S_{ijk} to (x_0, x_i, x_j, x_k) is invertible. Namely, we have Equations (1')-(4), and inversely, that

$$\begin{aligned} S_{ijk}(0,0) &= x_i + x_j + x_k & \qquad S_{ijk}(0,1) &= x_0 + x_j + x_k \\ S_{ijk}(1,0) &= x_0 + x_i + x_k & \qquad S_{ijk}(1,1) &= x_0 + x_k. \end{aligned}$$

Now, if $x_0 = 0$ but at least one of (x_i, x_j, x_k) is non-zero, it must hold that S_{ijk} has at least two non-zero entries. Thus, if it happens that for a large fraction

(more than $1/2$) of constraints at least one of (x_i, x_j, x_k) is non-zero, it must be the case that the total weight of the S_{ijk}'s is larger than m. But of course, we have no way to guarantee such a condition on (x_i, x_j, x_k).

However, we can construct what morally amounts to a separate dummy instance of Max NAND that has this property, and then let it use the same x_0 variable as Ψ. Towards this end, let $C \subseteq \mathbb{F}_2^N$ be a linear code of relative distance $1/2 - \epsilon$. Here $\epsilon > 0$ will be chosen sufficiently small and for reasons that will become clear momentarily, the dimension of the code will be exactly n so that one can take $N = O(n)$.

Now we introduce $N + N^2$ new variables which we think of as a vector $y \in \mathbb{F}_2^N$ and matrix $Y \in \mathbb{F}_2^{N \times N}$. The vector y should be an element of C and the matrix Y should be an element of $C \otimes C$. The intention is that $Y = y \cdot y^\top$, or in other words, that for every $i, j \in [N]$ we have $Y_{ij} = y_i \cdot y_j$.

Analogously to the S_{ijk} functions intended to check the NAND constraints of Ψ, we now introduce for every $i, j \in [N]$ a function $Z_{ij} : \mathbb{F}_2^2 \to \mathbb{F}_2$ that is intended to check the constraint $Y_{ij} = y_i \cdot y_j$, and that is supposed to be the indicator of the assignment to the variables (y_i, y_j). We then impose the analogues of the constraints (1')-(4), viz.

$$Z_{ij}(0,0) + Z_{ij}(0,1) + Z_{ij}(1,0) + Z_{ij}(1,1) = x_0 \tag{5}$$

$$Z_{ij}(1,0) + Z_{ij}(1,1) = y_i \tag{6}$$

$$Z_{ij}(0,1) + Z_{ij}(1,1) = y_j \tag{7}$$

$$Z_{ij}(1,1) = Y_{ij}. \tag{8}$$

Finally, we impose two sets of additional constraints. First, we require that $y = C(x)$ is the encoding of x under C (recall that C has dimension exactly n so we can view it as a one-to-one linear map $C : \mathbb{F}_2^n \to \mathbb{F}_2^N$). The purpose of this is to ensure that x is non-zero if and only if y is. Second, we require that the matrix Y is symmetric and that the diagonal of Y equals y (using Lemma 1 this is going to enable us to get a better distance bound in the soundness analysis which turns out to be critical for the reduction to work).

The final subspace $\mathcal{S}$ will consist of the projection to the $4m$ different S_{ijk} variables and the $4N^2$ different Z_{ij} variables, but with each of the S_{ijk} variables repeated some $r \approx N^2/m$ number of times in order to make these two sets of variables of comparable size. Note that by Equations (1)-(4) and (5)-(8) these variables uniquely determine x_0, x, y and Y. Furthermore, because of the invertibility of these constraints, we have that if some S_{ijk} or Z_{ij} is non-zero it must hold that one of x_0, x, y and Y are non-zero. Figure 1 gives an overview of the different components of the reduction and their relations.

Let us then study the the completeness and soundness of the reduction. The completeness easily follows as in the previous section; using the intended indicator functions for the S_{ijk}'s and Z_{ij}'s we obtain a codeword of weight $N^2 + rm$. For the soundness, we now proceed by a simple case analysis.

As in the previous section, when x_0 is non-zero, each S_{ijk} and Z_{ij} must have at least one non-zero entry and all the δ fraction of the S_{ijk}'s corresponding to

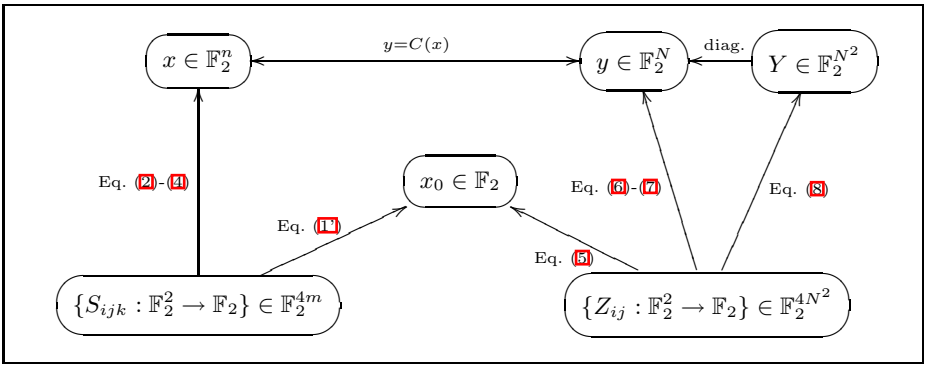


Fig. 1. The different components of the reduction to Min Dist(2). An arrow from one component to another indicates that the second component is a linear function of the first, with the label indicating the nature of this linear function.

unsatisfied NAND constraints of Ψ must have at least three non-zero entries, giving a total weight of $N^2 + (1 + 2\delta)rm$.

It remains to consider the case that x_0 is zero. Let us first look at the subcase that y is non-zero. Since $y \in C$ is a non-zero codeword, at least $(1/2 - \epsilon)N$ of its coordinates are non-zero. Thus, for at least $(3/4 - 2\epsilon)N^2$ pairs $(y_i, y_j) \neq (0, 0)$. For each such pair, the corresponding Z_{ij} function is non-zero, and as argued earlier, has at least two non-zero entries, which means that the total weight of the Z_{ij}'s is at least

$$2 \cdot (3/4 - 2\epsilon) \cdot N^2 = \left(\frac{3}{2} - 4\epsilon\right) \cdot N^2.$$

The next subcase is that x_0 and y are zero. In this case x must be zero (because of the constraint $y = C(x)$) and thus the matrix Y is non-zero. In this case, it is easily verified from Equations (5)-(8) that for each $i, j \in [N]$ such that Y_{ij} is non-zero, it must be that Z_{ij} has four non-zero entries. However, the distance of the code $C \otimes C$ to which Y belongs is only $(1/2 - \epsilon)^2 < 1/4$, so it seems as though we just came short of obtaining a large distance. However, since we added the constraints that Y is symmetric and that its diagonal equals y (which is zero), Lemma 1 now implies that Y in fact has $(1/2 - \epsilon)^2 \cdot \frac{3}{2} > (1/4 - 2\epsilon)\frac{3}{2}$ fraction non-zero entries. As mentioned above, each corresponding Z_{ij} function has four non-zero entries giving a total weight of at least

$$4 \cdot (3/8 - 3\epsilon) \cdot N^2 = \left(\frac{3}{2} - 12\epsilon\right) \cdot N^2.$$

In summary, this gives if Ψ is satisfiable the minimum distance of $\mathcal{S}$ is $rm + N^2$, whereas if $\mathsf{Opt}(\Psi) \leq 1 - \delta$ every non-zero vector in $\mathcal{S}$ must have weight at least

$$\min\left((1 + 2\delta)rm + N^2, \left(\frac{3}{2} - 12\epsilon\right) \cdot N^2\right).$$

Choosing $\epsilon > 0$ sufficiently small and

$$r \approx \frac{N^2}{2(1+2\delta)m}$$

we obtain that it is NP-hard to approximate MIN DIST(2) to within some factor $\delta' > 1$. We have not yet proved that $\mathcal{S}$ has good rate and distance – see the full version [4] for details.

4 Interlude: Linear Approximations to Nonlinear Codes

In our hardness result for MIN DIST(q), we need explicit constructions of certain codes which can be thought of as serving as linear approximations to some nonlinear codes. In particular, we need a sequence of linear codes $C_1, \ldots, C_{q-1}$ over $\mathbb{F}_q$ with the following two properties:

1. $d(C_e) \gtrsim (1 - e/q) \cdot N$ for $1 \leq e \leq q-1$.
2. If $x \in C_1$ then $x^e \in C_e$ for $1 \leq e \leq q-1$. Here x^e denotes a vector that is componentwise e^{th} power of x.

In other words, C_e should contain the nonlinear code $\{x^e\}_{x \in C_1}$, while still having a reasonable distance. In this sense we can think of C_e as a linear approximation to a nonlinear code. To obtain such a sequence of codes, we use pseudorandom generators for low-degree polynomials. Such pseudorandom generators were recently constructed by Viola [17] (building on [5, 14]), who showed that the sum of d PRGs for linear functions fool degree d polynomials. Using his result, and PRGs against linear functions of optimal seed length $\log_q n + O(1 + \log_q 1/\epsilon)$ (see e.g., Appendix A of [5]), one obtains the following theorem.

Theorem 4. *For every prime power q, $d > 0$, $\epsilon > 0$ there is a constant $c := c(q, d, \epsilon)$ such that for every $n > 0$, there is a polynomial time constructible (multi)set $R \subseteq \mathbb{F}_q^n$ of size $|R| \leq c \cdot n^d$ such that, for any polynomial $f : \mathbb{F}_q^n \to \mathbb{F}_q$ of total degree at most d, it holds that*

$$\sum_{a \in \mathbb{F}_q} \left| \Pr_{x \sim R}[f(x) = a] - \Pr_{x \sim \mathbb{F}_q^n}[f(x) = a] \right| \leq \epsilon. \tag{9}$$

A simple corollary of (9) and the Schwarz-Zippel Lemma is the following.

Corollary 1. *If $d = q - 1$ the (multi)set $R \subseteq \mathbb{F}_q^n$ constructed in Theorem 4 has the property that for every non-zero polynomial $f : \mathbb{F}_q^n \to \mathbb{F}_q$ of total degree at most $e \leq q - 1$,*

$$\Pr_{x \sim R}[f(x) \neq 0] \geq 1 - e/q - \epsilon. \tag{10}$$

Now define, for $1 \leq e \leq q-1$, C_e to be the set of all vectors $(f(x))_{x \in R}$ where $f : \mathbb{F}_q^n \mapsto \mathbb{F}_q$ is a polynomial of total degree at most e. Clearly, C_e is a linear subspace of $\mathbb{F}_q^{|R|}$. As observed in Corollary 1, the relative distance of C_e

is essentially $1 - e/q$. Moreover, any $v \in C_1$ is the evaluation vector of a degree one polynomial, and hence v^e is the evaluation vector of a degree e polynomial, and therefore $v^e \in C_e$ as desired. Also, note that the dimension of C_e is of order n^e and in particular C_{q-1} has constant rate.

5 Reduction to Min Dist(q) for $q \geq 3$

We now describe a general reduction from the Max NAND problem to the Min Dist(q) problem for any constant prime power q. The basic idea is the same as in the $\mathbb{F}_2$ case but some additional work is needed both in the reduction itself and in its analysis. For simplicity we here assume that $q \geq 3$ as the binary case was already handled in the previous section.

As before, let n be the number of variables in the Max NAND instance Ψ and m the number of constraints. Fix some small enough parameter ϵ and let $R \subseteq \mathbb{F}_q^n$ be the ϵ-pseudorandom set for degree $q-1$ polynomials $\mathbb{F}_q^n \to \mathbb{F}_q$ given by Theorem 4. Let $N = |R| = O(n^{q-1})$.

For $0 \leq d \leq q-1$, let $P_d \subseteq \mathbb{F}_q^N$ be the linear subspace of all polynomials of degree at most d in n variables with coefficients in $\mathbb{F}_q$, evaluated at points on R. I.e., all vectors in P_d are of the form $(p(x))_{x \in R}$ for some polynomial $p \in \mathbb{F}_q[X_1, \ldots, X_n]$ with $\deg(p) \leq d$. Note that P_d is a linear code and by Corollary 1, its relative distance is at least $1 - d/q - \epsilon$. From here on, we will ignore the parameter $\epsilon > 0$; it can be chosen to be sufficiently small (independent of q and the inapproximability for Max NAND) and hence the effect of this can be made insignificant.

We define $C = P_1$ and for $\alpha \in \mathbb{F}_q^n$ we write $C(\alpha) \in \mathbb{F}_q^N$ for the encoding of α under C; this corresponds to the evaluations of the linear polynomial $\sum_{i=1}^n \alpha_i X_i$ at all points $(X_1, \ldots, X_n)$ in R. Conversely, for a codeword $y \in C$ we write $\alpha = C^{-1}(y) \in \mathbb{F}_q^n$ for the (unique) decoding of y.

We now construct a linear code $\mathcal{C}'(\Psi)$ with variables as described in Figure 2. As in the $\mathbb{F}_2$ case, the final code $\mathcal{C}(\Psi)$ will consist of the projection of these variables to the Z_{ij}'s and the S_{ijk}'s, which determine the remaining variables by the constraints that we shall define momentarily.

Before we describe the constraints defining $\mathcal{C}'(\Psi)$ it is instructive to describe the intended values of these variables. Loosely speaking, the different Y variables are supposed to be an encoding of an assignment $\alpha \in \mathbb{F}_2^n$ to Ψ, the function S_{ijk} is a check that α satisfies the equation $x_k = 1 + x_i \cdot x_j$, and the Z_{ij} functions

1. For every $0 \leq e \leq 2(q-1)$ a vector $Y^e \in \mathbb{F}_q^N$.
2. For every $0 \leq e, f \leq q-1$ a matrix $Y^{e,f} \in \mathbb{F}_q^{N^2}$.
3. For every $1 \leq i, j \leq N$ a function $Z_{ij} : \mathbb{F}_q^2 \to \mathbb{F}_q$ (i.e., a vector in $\mathbb{F}_q^{q^2}$).
4. For every equation $x_k = 1 + x_i \cdot x_j$ in Ψ, a function $S_{ijk} : \mathbb{F}_2^2 \to \mathbb{F}_q$ (i.e., a vector in $\mathbb{F}_q^4$).

Fig. 2. Variables of $\mathcal{C}'(\Psi)$

1. Y^e is supposed to be $C(\alpha)^e$ (where we think of $\mathbb{F}_2^n$ as a subset of $\mathbb{F}_q^n$ in the obvious way).
2. $Y^{e,f}$ is supposed to be $C(\alpha)^e \cdot (C(\alpha)^f)^\top$ (i.e., we should have $Y^{e,f}(i,j) = C(\alpha)_i^e C(\alpha)_j^f$.
3. Z_{ij} is supposed to be the indicator function of $(C(\alpha)_i, C(\alpha)_j)$ (i.e., $Z_{ij}(x,y)$ should be 1 if $x = C(\alpha)_i$ and $y = C(\alpha)_j$; and 0 otherwise).
4. S_{ijk} is supposed to be the indicator function of (α_i, α_j) (i.e., $S_{ijk}(a,b) = 1$ if $\alpha_i = a$ and $\alpha_j = b$; and 0 otherwise).

Fig. 3. Intent of variables of $\mathcal{C}'(\Psi)$

check that the Y variables resemble a valid encoding of some α. Specifically, the variables are supposed to be assigned as described in Figure 3.

We categorize the constraints of $\mathcal{C}'(\Psi)$ as being of two different types, namely *basic constraints* that aim to enforce rudimentary checks of Items 1 and 2 of Figure 3, and *consistency constraints* that aim to use the Z_{ij}'s and S_{ijk}'s to check that the $Y^{e,f}$ matrices are consistent with an encoding of a good assignment to Ψ. As a comparison with the reduction for $\mathbb{F}_2$ in Section 3, the basic constraints correspond to the horizontal arrows on the upper side of Figure 1, and the consistency constraints correspond to the other arrows, i.e., Equations (1')-(8). Keeping the interpretation from Figure 3 in mind, the basic constraints that we impose are given in Figure 4.

1. For $0 \leq e \leq q-1$, $Y^e \in P_e$.
2. For $q \leq e \leq 2(q-1)$, $Y^e = Y^{e-(q-1)}$.
3. For $0 \leq e, f \leq q-1$:
   (a) $Y^{e,f} \in P_e \otimes P_f$.
   (b) The diagonal of $Y^{e,f}$ equals Y^{e+f}.
4. For $0 \leq e \leq q-1$, the rows (resp. columns) of $Y^{0,e}$ (resp. $Y^{e,0}$) are identical (and therefore equal to Y^e as this is the diagonal).
5. The matrix $Y^{q-1,q-1}$ is symmetric[3].

Fig. 4. Basic constraints of $\mathcal{C}'(\Psi)$

Note that all entries of the matrix $Y^{0,0}$ must be equal, and that in the intended assignment they should equal the constant 1. For notational convenience let us write $Y_0 \in \mathbb{F}_q$ for the value of the entries of $Y^{0,0}$ (this variable plays the same role as the variable x_0 in the reduction for $\mathbb{F}_2$ in Section 3).

We then turn to the consistency constraints of $\mathcal{C}'(\Psi)$, which are described in Figure 5.

The four equations (11) are the same as Equations (1)-(4) from the $\mathbb{F}_2$ reduction, the only difference being that they are now constraints over $\mathbb{F}_q$. Note that instead of Y_0 we would like to use the constant 1 in the above constraint, but as

[3] In general we could add the constraint that $Y^{e,f} = (Y^{f,e})^\top$ for every e, f, but it turns out we only need it for the case $e = f = q-1$.

1. For every constraint $x_k = 1 + x_i \cdot x_j$ of Ψ, four constraints on S_{ijk}:

$$\begin{aligned} Y_0 &= \sum_{a,b\in\mathbb{F}_2} S_{ijk}(a,b) & \alpha_i &= \sum_{a,b\in\mathbb{F}_2} a \cdot S_{ijk}(a,b) \\ \alpha_j &= \sum_{a,b\in\mathbb{F}_2} b \cdot S_{ijk}(a,b) & \alpha_k &= \sum_{a,b\in\mathbb{F}_2} (1 \oplus a \cdot b) \cdot S_{ijk}(a,b). \end{aligned} \tag{11}$$

   (Here $\oplus$ denotes addition in $\mathbb{F}_2$ and the remaining summations are over $\mathbb{F}_q$.)
2. For every $i, j \in [N]$, q^2 constraints on Z_{ij}: for every $0 \le e, f \le q-1$ it must hold that

$$Y^{e,f}(i,j) = \sum_{x,y\in\mathbb{F}_q} x^e y^f Z_{ij}(x,y). \tag{12}$$

Fig. 5. Consistency constraints of $\mathcal{C}'(\Psi)$

we are not allowed to do this we use Y_0, which, as mentioned above, is intended to equal 1. Note also that $Y^1 = C(\alpha)$, and thus α is implicitly defined by Y^1. If one wanted to be precise, one would write $C^{-1}(Y^1)_i$ instead of α_i in the above equations.

The function S_{ijk} is an invertible linear transformation of $\{Y_0, \alpha_i, \alpha_j, \alpha_k\}$ and hence is non-zero if and only if one of those four variables are non-zero. Similarly, from (12) it follows that Z_{ij} is an invertible linear transformation of the set of (i,j)'th entries of the q^2 different matrices $\{Y^{e,f}\}_{0\le e,f\le q-1}$. In particular Z_{ij} is non-zero if and only if the (i,j)'th entry of some matrix $Y^{e,f}$ is non-zero.

The final code $\mathcal{C}(\Psi)$ contains the projection of these variables to the functions Z_{ij} and the functions S_{ijk}, with each S_{ijk} repeated $r \ge 1$ times. Note that $\mathcal{C}(\Psi)$ is a subspace of $\mathbb{F}_q^M$ where $M = (qN)^2 + 4rm$. The completeness and soundness are as follows.

Lemma 2 (Completeness). *If* $\mathsf{Opt}(\Psi) = 1$ *then*

$$d(\mathcal{C}(\Psi)) \le N^2 + rm.$$

Lemma 3 (Soundness). *If* $\mathsf{Opt}(\Psi) \le 1 - \delta$ *then*

$$d(\mathcal{C}(\Psi)) \ge \min\left(N^2 + (1+\delta)rm, (1+1/q)N^2\right).$$

Lemma 4 ($\mathcal{C}$ is a Good Code). *The dimension of* $\mathcal{C}(\Psi)$ *is* $\Omega(N^2)$*, and the distance is at least* N^2*.*

Setting $r \approx \frac{N^2}{(1+\delta)qm}$, Lemmas 2-4 give Theorem 1 (for the case $q \ge 3$). Proofs of the three lemmas can be found in the full version [4].

References

[1] Ajtai, M.: The shortest vector problem in L_2 is NP-hard for randomized reductions. In: Proc. 30th ACM Symposium on the Theory of Computing, pp. 10–19 (1998) 475

[2] Arora, S., Lund, C., Motawani, R., Sudan, M., Szegedy, M.: Proof verification and the hardness of approximation problems. Journal of the ACM 45(3), 501–555 (1998) 477
[3] Arora, S., Safra, S.: Probabilistic checking of proofs: A new characterization of NP. Journal of the ACM 45(1), 70–122 (1998) 477
[4] Austrin, P., Khot, S.: A Simple Deterministic Reduction for the Gap Minimum Distance of Code Problem. arXiv report 1010.1481 476, 481, 484
[5] Bogdanov, A., Viola, E.: Pseudorandom bits for polynomials. SIAM J. Comput. 39(6), 2464–2486 (2010) 481
[6] Cai, J., Nerurkar, A.: Approximating the SVP to within a factor $(1 + 1/\dim^{\epsilon})$ is NP-hard under randomized reductions. Journal of Computer and Systems Sciences 59(2), 221–239 (1999) 475
[7] Cheng, Q., Wan, D.: Complexity of decoding positive-rate reed-solomon codes. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part I. LNCS, vol. 5125, pp. 283–293. Springer, Heidelberg (2008) 474, 475
[8] Cheng, Q., Wan, D.: A deterministic reduction for the gap minimum distance problem. In: Proceedings of the ACM Symposium on the Theory of Computing, pp. 33–38 (2009) 474, 475
[9] Dumer, I., Micciancio, D., Sudan, M.: Hardness of approximating the minimum distance of a linear code. In: Proc. 40th IEEE Symposium on Foundations of Computer Science (1999) 474, 475
[10] Feige, U., Goldwasser, S., Lovász, L., Safra, S., Szegedy, M.: Interactive proofs and the hardness of approximating cliques. Journal of the ACM 43(2), 268–292 (1996) 477
[11] Haviv, I., Regev, O.: Tensor-based hardness of the shortest vector problem to within almost polynomial factors. In: Proceedings of the ACM Symposium on the Theory of Computing, pp. 469–477 (2007) 475
[12] Khot, S.: Hardness of approximating the shortest vector problem in high L_p norms. In: Proc. 44th IEEE Symposium on Foundations of Computer Science (2003) 475
[13] Khot, S.: Hardness of approximating the shortest vector problem in lattices. In: Proc. 45th IEEE Symposium on Foundations of Computer Science, pp. 126–135 (2004) 475
[14] Lovett, S.: Unconditional pseudorandom generators for low degree polynomials. Theory of Computing 5(1), 69–82 (2009) 481
[15] Micciancio, D.: The shortest vector problem is NP-hard to approximate to within some constant. SIAM Journal on Computing 30(6), 2008–2035 (2000) 475
[16] Vardy, A.: The intractability of computing the minimum distance of a code. IEEE Transactions on Information Theory 43(6), 1757–1766 (1997) 474
[17] Viola, E.: The sum of d small-bias generators fools polynomials of degree d. Computational Complexity 18(2), 209–217 (2009) 475, 481

Recoverable Values for Independent Sets

Uriel Feige and Daniel Reichman

Department of Computer Science and Applied Mathematics,
The Weizmann Institute of Science, Rehovot, Israel
uriel.feige@weizmann.ac.il, daniel.reichman@gmail.com

Abstract. The notion of *recoverable value* was advocated in work of Feige, Immorlica, Mirrokni and Nazerzadeh [Approx 2009] as a measure of quality for approximation algorithms. There this concept was applied to facility location problems. In the current work we apply a similar framework to the maximum independent set problem (MIS). We say that an approximation algorithm has *recoverable value* ρ, if for every graph it recovers an independent set of size at least $\max_I \sum_{v \in I} \min[1, \rho/(d(v)+1)]$, where $d(v)$ is the degree of vertex v, and I ranges over all independent sets in G. Hence, in a sense, from every vertex v in the maximum independent set the algorithm recovers a value of at least $\rho/(d_v+1)$ towards the solution. This quality measure is most effective in graphs in which the maximum independent set is composed of low degree vertices. It easily follows from known results that some simple algorithms for MIS ensure $\rho \geq 1$. We design a new randomized algorithm for MIS that ensures an expected recoverable value of at least $\rho \geq 7/3$. In addition, we show that approximating MIS in graphs with a given k-coloring within a ratio larger than $2/k$ is unique games hard. This rules out a natural approach for obtaining $\rho \geq 2$.

1 Introduction

The notion of *recoverable value* was advocated in work of Feige, Immorlica, Mirrokni and Nazerzadeh [5] as a measure of quality for approximation algorithms. This notion leads to greater expressive power for stating the guarantees of approximation algorithms (compared to the standard notion of approximation ratio), by this leading to greater differentiation among the performance guarantees of different algorithms. The hope is that this concept will lead to the design of new algorithms with superior performance with respect to the recoverable value measure (regardless of whether the classic approximation ratio differs from that of existing algorithms), and moreover, that these algorithms will have better performance in practice (at least in some interesting special cases).

In [5], the term PASS approximation (where PASS is an acronym for *PArameterized by the Signature of the Solution*) was used in order to capture the two main features that we wish the recoverable value to have. One feature is that the recoverable value is expressed in terms of properties of the (unknown) solution, rather than of the input instance. The other is that the property of the solution that it refers to is not some aggregate property (such as average

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 486–497, 2011.

degree), but rather some *signature* in which the contribution of each individual component of the solution is considered separately. Rather than try to present general principles here, let us focus on the problem studied in the current paper, that of maximum independent set, and specialize the notion of recoverable value to this problem.

We use the following notation. All graphs in this work are undirected. The degree $d(v)$ of vertex v is the number of neighbors of v. The set of neighbors of v is $N(v)$. The average degree of a graph is denoted by d_{avg}, and $d_{avg(U)} = \frac{1}{|U|} \sum_{v \in U} d(v)$ denotes the average over degrees of vertices in a set U. An independent set in a graph is a set of vertices I such that every two vertices in I are non-neighbors. We shall refer to the problem of finding an independent set of maximum cardinality as MIS. The independence ratio of $G = (V, E)$ is $\alpha(G) = |I_{max}|/|V|$, where $|I_{max}|$ is the size of a maximum independent set I. In the maximum weight independent set problem MWIS, every vertex v has a nonnegative weight w_v and the goal is to find an independent set of maximum weight.

We now present our notion of recoverable value for MIS. For an independent set I, we define its *signature* to be the sequence of degrees of vertices in I. The value that we shall want to recover from each vertex of I depends on its degree. With every degree d we associate a recoverable value of $0 \leq \rho_d \leq 1$, and wish our approximation algorithms to find an independent set of size at least $\sum_{v \in I} \rho_{d(v)}$. The independent set I is not known to the algorithm – it is only used in analyzing its performance measure. Hence this performance guarantee holds simultaneously with respect to all independent sets in the graph, including the independent set I that happens to maximizes the expression $\sum_{v \in I} \rho_{d(v)}$ (this I might not be the maximum independent set in the graph). The notion of recoverable value easily generalizes to MWIS, by multiplying each term by $w(v)$. For randomized algorithms, one considers the expected size (or weight for MWIS) of the independent set that they return.

Intuitively, the smaller the degree of v the more likely algorithms are to place v in an independent set (because v excludes a smaller number of other vertices), and hence the higher we would like its recoverable value to be. Hence it is natural for ρ_d to be a non-increasing function of d. We find it convenient to introduce a parameter ρ (that we shall attempt to maximize later) and to consider the function $\rho_d = \min[1, \rho/(d+1)]$. (Not allowing the recoverable value to exceed 1 is necessary, as we cannot find a solution larger than the optimal solution.) For graphs of minimum degree at least $\rho - 1$, such approximation algorithms need to find an independent set of size at least $\sum_{v \in I} \rho/(d(v) + 1)$.

We refer to ρ in the expression $\min[1, \rho/(d(v) + 1)]$ as the *canonical* recoverable value (though we sometimes omit the word canonical for brevity). For comparison with some previously published algorithms, it is useful to note that for every set U of vertices, $\sum_{v \in U} 1/(d_v + 1) \geq |U|/(d_{avg(U)} + 1)$ (with equality only if all vertices in U have the same degree). Hence for a given value of ρ, our notion of recoverable value that is based on the signature of I (its degree sequence) is more demanding than had we only considered the average degree of vertices in I.

1.1 Our Results

It is not hard to see that some known algorithms achieve a canonical recoverable value $\rho \geq 1$. It can be shown that they do not achieve a value of ρ bounded away from 1 (see [6]).

One can readily observe that $\rho_0 = \rho_1 = 1$. We use a procedure that we call *2-elimination* to that that for MIS one may enforce $\rho_2 = 1$ as well.

Thereafter, we design relatively simple new algorithms that achieve a recoverable value of $\rho \geq 2$ for MWIS and $\rho \geq 7/3$ for MIS. It is NP-hard to achieve $\rho = 4$, as this will imply exact solution of MIS in 3-regular graphs, a problem that is NP-hard and APX-hard [3]. However, in graphs in which the minimum degree δ is large we provide an algorithm achieving $\rho \geq \Omega(\log \delta / \log \log \delta)$ for MWIS.

Our investigations of the best recoverable value achievable by our approaches also lead us to consider MIS in k-colored graphs (graphs for which a k-coloring is given). For this problem a $2/k$ approximation ratio is known [13], and we show that improving it would refute the unique games conjecture.

Our proofs in some cases provide simple alternatives to previously published related results. For a more detailed version of the current manuscript, see [6].

1.2 Related Work

The strong inapproximability results for MIS [12] justify searching other measures of performance guarantees, and the notion of recoverable value is one such candidate. It considers degrees of vertices, and hence it is instructive to recall known approximation algorithms for MIS and MWIS and how their performance depends on the degree sequence of the graph.

Random permutation. Choosing a permutation uniformly at random and taking all vertices that appear prior to their neighbors in the order induced by the permutation produces an independent set whose expected weight is at least $\sum_{v \in V} \frac{w_v}{d(v)+1}$ (see [1], for example). This guarantees a recoverable value of $\rho \geq 1$.

Greedy. For MIS, iteratively picking a minimum degree vertex, adding it to an independent set I and deleting the vertex and its neighbors from the graph is guaranteed to find an independent set of size at least $\sum_{v \in V} \frac{1}{d(v)+1}$ [7,17]. Halldorsson and Radhakrishnan [11] showed that this greedy algorithm produces an independent set of size at least $|V| \frac{1+\alpha^2}{d_{avg}+1+\alpha}$ (where α denotes the fraction of vertices in the maximum independent set). For MWIS, *weighted greedy* that iteratively picks a vertex v with minimum $w_v/(d_v + 1)$ is guaranteed to find an independent set of size at least $\sum_{v \in V} \frac{w_v}{d(v)+1}$ [16].

LP. An integer programming formulation of MWIS is:
maximize $\sum_{i \in V} w_i x_i$
subject to
$x_i + x_j \leq 1$ for every edge (i, j).
$x_i \in \{0, 1\}$ for every vertex i.

Consider the LP relaxation of this program where each $x_i \in [0,1]$. A well known result due to Nemhauser and Trotter [15] asserts that there is an optimal solution for the relaxation such that for every $i, x_i \in \{0, \frac{1}{2}, 1\}$. Moreover, such an optimal solution can be found in polynomial time.

LP+greedy. Consider the following algorithm. Find an optimal half integral solution to the LP, discard all the vertices assigned 0, keep all the vertices assigned 1, and run the greedy algorithm on the graph induced by all vertices that are assigned 1/2. This algorithm was analyzed for connected graphs. Hochbaum [13] proved an approximation ratio of $\frac{2}{d_{avg}+1}$, and Halldorsson and Radhakrishnan [11] (based on their improved analysis of the greedy algorithm) proved an approximation ratio of $\frac{5}{2d_{avg}+3}$.

SDP. Using semidefinite programming Halldorsson [10] provided approximation ratios of $\Omega(\frac{\log d_{avg}}{d_{avg} \log\log d_{avg}})$ for MIS, and $\Omega(\frac{\log \delta}{\delta \log\log \delta})$ for MWIS on δ-*inductive* (a.k.a. δ-*degenerate*) graphs (in some permutation over the vertices δ is the maximum backward degree).

Local Search. In graphs of degree bounded by Δ, algorithms based on local search were shown to achieve an approximation ratio of roughly $5/(\Delta+3)$, with some distinction between the cases of odd and even Δ (see [3,4]). We remark that in the special case of Δ-regular graphs, the notion of recoverable value becomes equivalent to the traditional notion of approximation ratio, and our results are not as strong in this case as those achieved by local search. On the other hand, our algorithms are much faster than the local search algorithms.

In terms of hardness results, Austrin, Khot and Safra [2] proved that approximating independent set in graphs of maximum degree Δ within a ratio larger than $\frac{(\log \Delta)^2}{\Delta}$ is unique games hard. Our hardness results for finding independent sets in graphs where a k-coloring is given match the $2/k$ bounds achieved by known approximation algorithms [13]. Results for related problems on hypergraphs appeared in [9], and hardness results for MIS in graphs with bounded chromatic number but when no coloring is given were shown in [8].

1.3 Our Techniques

We begin by showing that For MIS we show that $\rho_0 = \rho_1 = \rho_2 = 1$. (See Section 2 for subtleties involved in the statement of this result.)

Thereafter, following a recipe suggested in [5] (though there the problem considered was different, facility location), we present an algorithm based on the so called recoverable value LP (see Theorem 2). This gives recoverable value of $\rho = 2$ for MWIS. For MIS we improve over this bound by using a new combination of some of the algorithms presented in Section 1.2. (A new combination is indeed needed, as we also provide examples showing that plain use of these algorithms does not even achieve ρ bounded away from 1.)

Given a graph G, choose a random permutation π on the vertices. We say that vertex v is in *layer* L_i with respect to π if exactly $i-1$ of v's neighbors

appear before v in π. For $k \geq 1$, let G_k be the subgraph of G induced on the vertices of the first i layers. The random permutation algorithm referred to in Section 1.2 simply returns G_1 which is an independent set. Our new algorithms will instead consider G_k with small value of k (depending on the context, we shall take $k \in \{2, 3, \delta+1\}$, where δ denotes the minimum degree), and on G_k run some algorithm from Section 1.2. The advantage of our approach (in the context of recoverable value) is that low degree vertices are more likely to end up in G_k. Moreover, G_k (for small values of k) has special structure that makes finding large independent sets easier. For example, it turns out that G_k is k-colorable. This suffices for obtaining a recoverable value of $\rho = 2$ (for MWIS on the original G), but does not guarantee anything better (by our new hardness of approximation results for approximating MIS in k-colored graphs). The fact that G_k is $(k-1)$-inductive allows us to obtain a recoverable value of $\rho = \Omega(\log \delta / \log\log \delta)$ (which improves over $\rho = 2$ when δ is large). Our unconditional improvement over $\rho = 2$ uses G_3 and applies only to MIS. For G_3 we considered its average degree, which is no longer a deterministic property but rather a function of several random variables (number of vertices in each layer). After showing that the $\frac{5}{2d_{avg}+3}$ approximation ratio of [11] extends to disconnected graphs (see Theorem 4), we show that the random variables can safely be replaced by their expectations (which are deterministic quantities), by this considerably simplifying the analysis. It was previously known that working with expectations often simplifies the analysis of randomized algorithms (through the use of linearity of the expectation), but the reason why it applies in our context is more delicate than usual, and may have applications also elsewhere. This leads to a canonical recoverable value $\rho \geq 15/7$. This can be improved to $\rho \geq 7/3$ by designing a new approximation algorithm for MIS in graphs of small average degree.

2 Handling Low Degree Vertices

It is instructive to consider separately ρ_d for $0 \leq d \leq 3$.

For isolated vertices, $\rho_0 = 1$ as they can be included in the optimal solution without changing the degree of any other vertex. Note an important point here. Our notion of recoverable value treats every vertex of I separately. Hence we can remove isolated vertices without effecting the remaining vertices in I. This might not have been so simple had we considered aggregate properties such as the average degree of vertices in I. Removing an isolated vertex increases the average degree for the remaining vertices.

Likewise, $\rho_1 = 1$ as every vertex of degree 1 can be included in the independent set, and its neighbor removed. At most one of these two vertices is in any independent set. For other vertices in I, this process can only lower their degrees, and hence their recoverable value does not decrease (since ρ_d is non-increasing).

The above arguments also imply that we can remove recursively vertices of degree at most 1 with no harm to the recoverable value. Hence we may assume that all graphs have minimum degree at least 2.

We show that for MIS $\rho_2 = 1$ (see Theorem 1 for an exact statement). Consider a graph G of minimum degree 2. Let u be a vertex of degree 2, and let v and w be its neighbors. Consider the following process termed *2-elimination*. If there is an edge (v, w) we add u to the independent set (and remove vertices v and w) because at most one of u, v, w is in any independent set, and u can replace any of v and w in an independent set. If v and w are not neighbors of each other remove u from G, merge v and w to become a new vertex u' whose neighbors are the original neighbors of v and w (except for u that was removed from the graph). Call the resulting graph G'. Any independent set I' in G' can be extended to an independent set I in G whose size is larger by 1. If $u' \in I'$, replace it by v and w. If $u' \notin I'$, then include u in I.

Consider how the recoverable value changes when transforming from G to G' where we assume canonical recoverable values. Without loss of generality, either $u \in I$ or both v and w are in I. In G', we take an independent set I' that is induced by I in a natural way (if $u \in I$ than u is simply lost, if $v, w \in I$ then $u' \in I'$). As $|I| = |I'| - 1$, we wish to show that their recoverable values differ by at most 1. If $u \in I$ then I' differs from I by the recoverable value of u which is indeed at most 1. If $v, w \in I$, then the recoverable values of G and G' differ by

$$\min[1, \frac{\rho}{d_v + 1}] + \min[1, \frac{\rho}{d_w + 1}] - \min[1, \frac{\rho}{d_{u'} + 1}] \tag{1}$$

Using the facts that $d_v, d_w \geq 2$ and $d_{u'} \leq d_v + d_w - 2$ the value of (1) is at most 1 whenever $\rho \leq 10/3$ (the worst choice of parameters being $d_v = d_w = 3$ and $d_{u'} = 4$). Hence if one considers canonical recoverable values, then if $\rho \geq 3$ this implies (by definition) that $\rho_2 = 1$, and if $\rho \leq 10/3$ then 2-elimination achieves $\rho_2 = 1$ (without affecting ρ for vertices of degrees larger than 2).

The above discussion establishes:

Theorem 1. *When using canonical recoverable values for MIS then regardless of the value of ρ one may assume that $\rho_0 = \rho_1 = \rho_2 = 1$.*

Note that there is some fixed $\epsilon > 0$ such that $\rho_3 \leq 1 - \epsilon$. This follows from the fact that approximating MIS in 3-regular graphs in APX-hard [3].

3 Algorithms for MWIS

In this section we assume that $G = (V, E)$ has been preprocessed to include all isolated vertices in the output independent set. This allows us to simplify notation from $\min[1, \rho/(d(v) + 1)]$ to $\rho/(d(v) + 1)$.

Theorem 2. *Let $G = (V, E)$ be a weighted graph without isolated vertices. There is a polynomial time algorithm for MWIS achieving a recoverable value of $\rho = 2$. Namely, the output of the algorithm is an independent set of weight at least $\sum_{v \in I} \frac{2w_v}{d(v)+1}$.*

Proof. We present two different polynomial time algorithms that achieve the desired bounds. One is based on linear programming. The other is much faster, but randomized.

LP algorithm. Consider the following recoverable value LP (the RV LP). x_i is a variable that indicates whether vertex i is in the independent set.

maximize $\sum_{i \in V} \frac{w_i}{d(i)+1} x_i$
subject to
$x_i + x_j \leq 1$ for every edge (i, j).
$0 \leq x_i \leq 1$ for every vertex i.

The indicator vector of any independent set I is a feasible solution to the recoverable value LP. Moreover, the value of the LP then would be the recoverable value with respect to I (up to a scaling factor of ρ). Hence the optimal value of the LP is at least the desired recoverable value with respect to best independent set I (scaled by $1/\rho$). Treating $\frac{w_i}{d(i)+1}$ as a weight of vertex i, the results of [15] imply that this LP has a half-integral optimal solution, and moreover that such a solution can be found in polynomial time.

Include in the independent set all vertices with $x_i = 1$ (getting credit w_i which is at least twice the credit $w_i/(d(i) + 1)$ that the LP got for them), and discard all vertices of $x_i = 0$ (the LP got no credit for them). Let $G_{1/2}$ be the weighted graph induced on all vertices assigned 1/2. Running weighted greedy on this graph ensures a solution of value $\sum_{i \in V_{1/2}} w_i/(d_i+1)$, whereas the LP (by having $x_i = 1/2$) got only half this credit. Hence after the rounding we obtain an integral solution of value at least twice that of the RV LP, implying $\rho \geq 2$.

A fast randomized algorithm. Choose a random permutation and consider G_2 as in Section 1.3. This graph is a forest (can be verified by orienting edges towards earlier vertices in the permutation). As every vertex v within I belongs to G_2 with probability $\frac{2}{d(v)+1}$, the expected weight of an independent set within G_2 is at least $\sum_{v \in I} \frac{2w_v}{d(v)+1}$. MWIS can be found in forests in linear time, proving Theorem 2. ∎

The bounds $\sum_{v \in V} \frac{w_v}{d(v)+1}$ (achieved by the random permutation algorithm of Section 1.2) and $\max_I \sum_{v \in I} 2w_v/(d(v) + 1)$ of Theorem 2 are incomparable. Nevertheless, it is not hard to see that both algorithms of Theorem 2 ensure not only the bound $\max_I \sum_{v \in I} 2w_v/(d(v) + 1)$ claimed in the statement of the theorem, but also the bound $\sum_{v \in V} \frac{w_v}{d(v)+1}$. For the LP algorithm this is attained by the feasible solution that assigns 1/2 to all variables. For the fast randomized algorithm this follows because returning the first layer is a legitimate output for it.

We now improve the recoverable value in the special case when the minimum degree δ in the input graph is sufficiently high.

Theorem 3. *Let $G = (V, E)$ be a weighted graph with minimum degree δ. There is a polynomial time algorithm for MWIS achieving a recoverable value of $\rho = \Omega(\log \delta / \log \log \delta)$.*

Proof. Choose a random permutation and consider $G_{\delta+1}$ as defined in Section 1.3. This graph is a δ-inductive (orienting edges towards earlier vertices in the permutation, no vertex has outdegree larger than δ). As every vertex v within I belongs to $G_{\delta+1}$ with probability $\frac{\delta+1}{d(v)+1}$ (here we used the assumption that $d(v) \geq \delta$), the expected weight of an independent set within $G_{\delta+1}$ is at least $\sum_{v \in I} \frac{w_v(\delta+1)}{d(v)+1}$. Run the SDP algorithm from [10] with approximation ratio $\Omega(\frac{\log \delta}{\delta \log \log \delta})$ (on δ-inductive graphs) to obtain a recoverable value with $\rho = \Omega(\log \delta / \log \log \delta)$. ∎

4 Algorithms for MIS

Here we show that for unweighted graphs there are randomized algorithms achieving a recoverable value of ρ strictly larger than 2. As in Section 2, we wish to simplify notation from $\min[1, \rho/(d(v)+1)]$ to $\rho/(d(v)+1)$. This requires that the input graph $G = (V, E)$ has no vertices of degree less than 2. As we say in the previous section, this can be assumed without loss of generality .

The basic idea of our algorithm is as follows. Pick a random permutation over the vertices and consider the graph G_3 induced on layers $L_1 \cup L_2 \cup L_3$. The expected size of $I \cap G_3$ is now $\sum_{v \in I} \frac{3}{d(v)+1}$. (Here we use the fact that the minimum degree in G is 2). One would expect all three layers L_1,L_2 and L_3 to be of equal size. Moreover, every vertex in L_1 contributes no edge to G_3, every vertex in L_2 contributes at most one edge to G_3, and every vertex in L_3 contributes at most two edges to G_3. Hence one would expect the average degree of G_3 to be at most 2. Recall that the algorithm of [11] (LP+greedy) obtains an approximation ratio of $\frac{5}{2d_{avg}+3}$ (on connected graphs). Hence, applying this algorithm to G_3 (and using the bounds of [11] even though G_3 need not be connected), one may hope to attain recoverable value of $\frac{15}{7} \sum_{v \in I} \frac{1}{d(v)+1}$.

Summarizing, we have algorithm PLG (Permute, LP, Greedy).

1. Pick a random permutation over the vertices and consider the graph G_3 induced on layers $L_1 \cup L_2 \cup L_3$.
2. Find a half-integral solution to the LP for MIS on G_3. Put the vertices with value 1 in the independent set and remove the vertices with value 0.
3. Run the greedy algorithm on the subgraph induced on the vertices that remain from G_3.

The above argument of why PLG achieves a value of $\rho = 15/7$ had two gaps in it. One relates to the assumption that G_3 is connected (which might not hold), and the other to the assumption that sizes of layers are equal to their expectations. We shall deal with each one of them separately.

As noted, the approximation ratio of $\frac{5}{2d_{avg}+3}$ of the algorithm of [11] assumes that the graph is connected (plugging in this expression with $d_{avg} < 1$ lead to ratios larger than 1 which of course cannot hold). Nevertheless, when the average degree is at least 2, the following theorem shows that the bound of $\frac{5}{2d_{avg}+3}$ does

apply, regardless of whether the graph is connected or not. Observe that all connected graphs are captured by the theorem (either they are trees and then greedy solves them optimally, or their average degree is at least 2), and hence our proof can also replace the one given in [11] for connected graphs.

Theorem 4. *If G is a graph of average degree at least 2 then MIS can be approximated within ratio of $\frac{5}{2d_{avg}+3}$.*

Proof. Recall that by the results of [11], the approximation ratio of the greedy algorithm on graphs with average degree d_{avg} and independence ratio α is $f(\alpha) = \frac{1+\alpha^2}{(d_{avg}+1+\alpha)\alpha}$. This ratio as a function of α is decreasing in the range $(0, 1/2]$. For $\alpha = 1/2$ we obtain the desired approximation ratio of $\frac{5}{2d_{avg}+3}$. Hence to prove Theorem 4 it suffices to show that we can assume that $\alpha(G) \leq 1/2$.

Consider an optimal half integral solution to the standard LP relaxation of the MIS problem (as implied by [15]). Let ONE denote the set of variables receiving 1, $ZERO$ the set of variables receiving 0, and $HALF$ the set of variables receiving 1/2. Let H be the subgraph induced on the vertices whose corresponding variables are in $HALF$. The independence ratio of H is at most 1/2, as desired. Necessarily $|ONE| \geq |ZERO|$ (otherwise they would both be in $HALF$). Remove ONE and $ZERO$ and all edges connected to them (and thus only H is left), and add instead $|ONE|$ isolated edges (one edge for each vertex of ONE), thus obtaining a new graph G'. Observe that the size of the maximum independent set does not change, but $\alpha(G') \leq 1/2$ as desired. It remains to analyze the average degree of G'. In the removal phase $|ONE| + |ZERO|$ vertices and at least $|ZERO|$ edges are removed (every vertex in $ZERO$ has an edge to ONE, otherwise it could be moved to $HALF$). Thereafter $2|ONE|$ vertices and $|ONE|$ edges are added. Hence altogether exactly $|ONE| - |ZERO|$ vertices and at most $|ONE| - |ZERO|$ edges are added. If the average degree of G is at least 2, the average degree of G' cannot be larger than that of G.

Summarizing, we have shown how to transform G into a new graph G' for which $\alpha(G') \leq 1/2$, the average degree of G' is at most that of G, and the maximum independent sets in G and G' have the same size. Applying greedy to G' is the same as taking ONE into the solution and applying greedy only on H, which is precisely the algorithm of [11]. By the properties of G', the approximation ratio is at least as desired by Theorem 4. ∎

Apparently, the result of Theorem 4 can be extended also to $d_{avg} \geq 3/2$ (Halldorsson, private communication), though this is not needed in our paper.

Now let us deal with the issue of expectations. Let us first explain the problem. We have various information about expectations. Namely, $E[|I \cap V(G_3)|] = \sum_{v \in I} 3/(d(v)+1)$, and $E[|L_1|] = E[|L_2|] = E[|L_3|]$. Moreover, the number of edges in G_3 is at most $2|L_3| + |L_1|$. If things behave exactly as expectation the average degree of G_3 is at most 2, Theorem 4 applies and we get an approximation ratio of $\frac{5}{7}\sum_{v \in I} 3/(d(v)+1)$ as desired. However, there is also variability in the above random variables, and it can be quite large. (The complete bipartite

graph $K_{3,d}$ illustrates this variability.) With some probability the average degree in G_3 may be larger than 2, and with some probability smaller. Hence the approximation ratio on G_3 is a random variable. Moreover, when the average degree of G_3 is too small, the bounds of Theorem 4 no longer hold. Moreover, the size of the maximum independent set in G_3 might be correlated with the average degree in complicated ways.

We present here a very simple way to handle all the above complications.

Theorem 5. *The expected size the independent set found by algorithm PLG is at least* $\frac{15}{7}\sum_{v\in I}\frac{1}{d(v)+1}$*, where* I *is any independent set in* G*.*

Proof. Our proof uses two important facts. One is that Theorem 4 applies also to disconnected graphs. The other (which the reader may verify) is that algorithm PLG when run on a disconnected graph gives the same outcome (or more formally, the same probability distribution on outcomes) as that when PLG is run on each connected component separately.

Let $\epsilon > 0$ be any desired level of accuracy with which we want to estimate the performance guarantee of PLG. Let G be an arbitrary graph on which we run PLG, let n be the number of its vertices, and let I be an independent set in G. As a thought experiment, make N disjoint copies of G, where N is chosen to be sufficiently large as a function of ϵ and n. Call the resulting graph on nN vertices G_N, and let I_N be the independent set composed from the N copies of I. Run PLG on G_N. With respect to PLG, the random variables L_1, L_2, L_3, and $I_N \cap G_3$ are all concentrated around their expectation within relative errors that are $o(\epsilon)$ (by our large choice of N and the fact that these random variables are each a sum of N bounded and independent random variables, one for each copy of G). Hence on G_N, with probability that can be made $(1-\epsilon/2)$ PLG finds an independent set of size at least $(1-\epsilon/2)\frac{15}{7}\sum_{v\in I_N}\frac{1}{d(v)+1}$. Hence in expectation, per copy of G, the size of the independent set found is at least $(1-\epsilon)\frac{15}{7}\sum_{v\in I_N}\frac{1}{d(v)+1}$. Letting ϵ tend to 0 Theorem 5 is proved. ∎

Theorem 5 can be strengthened by replacing the algorithm of [11] (used in the proof of Theorem 4) by an algorithm that approximates MIS in graphs with average degree 2 within a ratio better than 5/7. In [6] we show such an algorithm with approximation ratio 7/9. This implies:

Theorem 6. *A canonical recoverable value* $\rho = 7/3$ *is achievable for MIS.*

5 MIS in k-Colored Graphs

In analyzing algorithm PLG we only used the fact that G_3 has small average degree. However, G_3 has other structural properties as well. It is 3-colorable, and furthermore, the 3-coloring can be found efficiently (e.g., by coloring the vertices of G_3 inductively in the order in which they appear in the permutation that generated G_3). We use the term k-colored graph to denote a graph with a given k-coloring. The following proposition is known [13].

Proposition 1. *MIS on k-colored graphs can be approximated within $2/k$.*

One may hope that Proposition 1 can be improved, leading to an approximation ratio better than 2/3 for G_3, hence potentially replacing or even surpassing the 5/7 bound that we used for G_3 based on Theorem 4.

We show that Proposition 1 is nearly tight, unless vertex cover can be approximated within a ratio better than 2. (In particular, this implies UGC-hardness, since hardness of unique games is a stronger assumption than inapproximability of VC beyond a ratio of 2, see [14].)

Let G be an n-vertex graph in which one wants to approximate VC within a ratio better than 2. As is well known (e.g., [11]), this is the same as distinguishing for some $\epsilon > 0$ whether $\alpha(G)$, the size of MIS in G, satisfies $\alpha(G) \geq (1/2 - \epsilon)n)$ or $\alpha(G) \leq \epsilon n$.

For $k > 2$ construct from G a k-colored graph G' as follows. For every vertex $v \in G$, the graph G' contains k copies $v_1, \ldots v_k$. All vertices with the same index i form an independent set (hence a color class). Between any two distinct color classes other than class k, place a bipartite graph mimicking the edge pattern of G, connecting vertex v_i with vertex u_j (where $k \neq j \neq i \neq k$) if (v, u) (or (u, v)) is an edge in G. Each vertex v_k of the kth class is connected only to its own copies v_i in the other classes. Hence the bipartite graph between class k and any other class is simply a perfect matching.

Lemma 1. *In the reduction above, $\alpha(G') = n + (k-2)\alpha(G)$.*

This gives a gap of $\frac{n+O(\epsilon kn)}{kn/2-O(\epsilon kn)} = (1 + o(1))\frac{2}{k}$ between *no* and *yes* instances. To prove Lemma 1, we shall use the following lemma.

Lemma 2. *There is a MIS I' in G' such that for every vertex $v \in G$, either $v_i \in I'$ for all $i \neq k$, or $v_k \in I'$.*

Proof. Consider an arbitrary independent set I' in G'. If either no copy or only one copy of v is in I', then without loss of size of I' we may take this to be v_k. If at least two copies of v are in I', then neither one of them can be v_k. But then, we can add all other v_i with $i \neq k$ to I', since none of them can be a neighbor of a vertex already in I'. ∎

Lemma 2 implies that all vertices $v \in G$ for which $v_k \notin I'$ form an independent set in G. Lemma 1 easily follows. Applying Lemma 1 and [14] we immediately obtain:

Theorem 7. *Let k be an integer greater than 2. Assume that for every $\epsilon > 0$ it is NP-hard to distinguish between graphs with independent set of size at least $(\frac{1}{2} - \epsilon)n$ to graphs with independent sets of size at most ϵn. Then, for arbitrary $\delta > 0$ it is NP-hard to approximate MIS on k-colored graphs within a factor larger than $\frac{2}{k} + \delta$. In particular, approximating MIS on k-colored graphs within a factor larger than $\frac{2}{k} + \delta$ is unique-games hard.*

Acknowledgements

Work supported in part by the Israel Science Foundation (grant No. 873/08). We thank Magnus Halldorsson for sharing with us his thoughts on the range of values of d_{avg} for which Theorem 4 holds.

References

1. Alon, N., Spencer, J.: The Probablistic Method. Wiley, Chichester (2008)
2. Austrin, P., Khot, S., Safra, S.: Inapproximability of vertex cover and independent set in bounded degree graphs. In: CCC (2009)
3. Berman, P., Fujito, T.: On approximation properties of the independent set problem for low degree graphs. Theory of Computing Systems
4. Chlebík, M., Chlebíková, J.: On approximability of the independent set problem for low degree graphs. In: Kralovic, R., Sýkora, O. (eds.) SIROCCO 2004. LNCS, vol. 3104, pp. 47–56. Springer, Heidelberg (2004)
5. Feige, U., Immorlica, N., Mirrokni, V.S., Nazerzadeh, H.: Pass approximation. In: Serna, M., Shaltiel, R., Jansen, K., Rolim, J. (eds.) APPROX 2010, LNCS, vol. 6302. Springer, Heidelberg (2010)
6. Feige, U., Reichman, D.: Recoverable values for independent sets (Detailed version of current paper), http://www.arxiv.org/PS_cache/arxiv/pdf/1103/1103.5609v1.pdf
7. Griggs, J.: Lower bounds on the independence number in terms of the degrees. Journal of Combinatorial Theory, Series B 34(1), 22–39 (1983)
8. Guruswami, V., Kemal Sinop, A.: The complexity of finding independent sets in bounded degree (hyper)graphs of low chromatic number. In: SODA (2011)
9. Guruswami, V., Saket, R.: On the inapproximability of vertex cover on k-partite k-uniform hypergraphs. In: Abramsky, S., Gavoille, C., Kirchner, C., Meyer auf der Heide, F., Spirakis, P.G. (eds.) ICALP 2010. LNCS, vol. 6198, pp. 360–371. Springer, Heidelberg (2010)
10. Halldorsson, M.M.: Approximations of weighted independent set and hereditary subset problems. Journal of Graph Algorithms and Applications 4, 1–16 (2000)
11. Halldorsson, M.M., Radhakrishnan, J.: Greed is good: Approximating independent sets in sparse and bounded-degree graphs. Algorithmica 18, 145–163 (1997)
12. Hastad, J.: Clique is hard to approximate within $n^{1-\epsilon}$. Acta Mathematica 182, 105–142 (1999)
13. Hochbaum, D.: Efficient bounds for the stable set, vertex cover, and set packing problems. Discrete Appl. Math 6, 243–254 (1983)
14. Khot, S., Regev, O.: Vertex cover might be hard to approximate to within $2-\epsilon$. In: CCC (2003)
15. Nemhauser, G., Trotter, L.: Vertex packings: Structural properties and algorithms. Math. Programming 8, 232–248 (1975)
16. Sakai, M., Togasaki, M., Yamazaki, K.: A note on greedy algorithms for the maximum weighted independent set problem. Discrete Applied Mathematics 126, 313–322 (1999)
17. Wei, V.K.: A lower bound on the stability number of a simple graph. Bell Laboratories Technical Memorandum 8 I- I 12 17.9 (1981)

Vertex Cover in Graphs with Locally Few Colors

Fabian Kuhn[1] and Monaldo Mastrolilli[2]

[1] Faculty of Informatics, University of Lugano (USI), 6904 Lugano, Switzerland
fabian.kuhn@usi.ch
[2] Dalle Molle Institute for Artificial Intelligence (IDSIA), 6928 Manno, Switzerland
monaldo@idsia.ch

Abstract. In [13], Erdős et al. defined the local chromatic number of a graph as the minimum number of colors that must appear within distance 1 of a vertex. For any $\Delta \geq 2$, there are graphs with arbitrarily large chromatic number that can be colored so that (i) no vertex neighborhood contains more than Δ different colors (*bounded local colorability*), and (ii) adjacent vertices from two color classes induce a complete bipartite graph (*biclique coloring*).

We investigate the weighted vertex cover problem in graphs when a locally bounded coloring is given. This generalizes the vertex cover problem in bounded degree graphs to a class of graphs with arbitrarily large chromatic number. Assuming the Unique Game Conjecture, we provide a tight characterization. We prove that it is UGC-hard to improve the approximation ratio of $2 - 2/(\Delta + 1)$ if the given local coloring is not a biclique coloring. A matching upper bound is also provided. Vice versa, when properties (i) and (ii) hold, we present a randomized algorithm with approximation ratio of $2 - \Omega(1)\frac{\ln \ln \Delta}{\ln \Delta}$. This matches known inapproximability results for the special case of bounded degree graphs.

Moreover, we show that the obtained result finds a natural application in a classical scheduling problem, namely the precedence constrained single machine scheduling problem to minimize the total weighted completion time. In a series of recent papers it was established that this scheduling problem is a special case of the minimum weighted vertex cover in graphs $G_{\mathbf{P}}$ of incomparable pairs defined in the dimension theory of partial orders. We show that $G_{\mathbf{P}}$ satisfies properties (i) and (ii) where $\Delta - 1$ is the maximum number of predecessors (or successors) of each job.

Keywords: approximation, local coloring, scheduling, vertex cover

1 Introduction

Vertex Cover is one of the most studied problems in combinatorial optimization: Given a graph $G = (V, E)$ with weights w_i on the vertices, find a subset $V' \subseteq V$, minimizing the objective function $\sum_{i \in V'} w_i$, such that for each edge $\{u, v\} \in E$, at least one of u and v belongs to V'.

The related bibliography is vast and cannot be covered in this introductory note. We mention here that vertex cover cannot be approximated within

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 498–509, 2011.

a factor of 1.3606 [11], unless P=NP. Moreover, if the Unique Game Conjecture (UGC) [23] holds, Khot and Regev [24] show that vertex cover is hard to approximate within any constant factor better than 2. On the other side several simple 2-approximation algorithms are known (see e.g. [30,20]). Hochbaum [20] uses the natural linear program (LP) relaxation and a threshold rounding approach to obtain better than 2 approximation algorithms when a k-coloring of the graph is given as input. An optimal solution to the LP assigns a non-negative real value to each vertex of the input graph G. It is well known that such a solution is half-integral [30]. Let S_1 be the vertices of the input graph which attain value 1 in this solution; and let S_2 be the vertices which attain value 1/2. The vertices S_1 together with a cover of the subgraph induced by S_2 are sufficient to cover the whole graph G. A k-coloring of the subgraph induced by S_2 gives an independent set of value at least $w(S_2)/k$, where $w(S_2)$ is the sum of the vertex weights in S_2. This yields a $(2-2/k)$-approximation for the minimum weighted vertex cover problem. For graphs with degree bounded by d, this directly leads to a $(2-2/d)$-approximation.

This basic approach has been considerably improved by Halperin [19]. The improvement is obtained by replacing the LP relaxation with a stronger semidefinite program (SDP) relaxation, and using a fundamental result by Karger et al. [22]. The algorithm in [19] achieves a performance ratio of $2-(1-o(1))\frac{2\ln\ln d}{\ln d}$, which improves the previously known [18] ratio of $2-\frac{\ln d+O(1)}{d}$. Under the UGC [23], Austrin, Khot and Safra [5] have recently proved that it is NP-hard to approximate vertex cover in bounded degree graphs to within a factor $2-(1+o(1))\frac{2\ln\ln d}{\ln d}$. This exactly matches the algorithmic result of Halperin [19] up to the $o(1)$ term. For general vertex cover, the currently best approximation ratio is due to Karakostas [21], achieving a performance of $2-\Theta(1/\sqrt{\ln n})$.

Brook's theorem states that except for complete graphs and odd cycles, graphs with maximum degree d can be colored with d colors. In this paper we consider the vertex cover problem in graphs with bounded local chromatic number, a generalization of the bounded degree case with arbitrarily large chromatic number.

For an undirected graph $G=(V,E)$ and a vertex $u\in V$, we use $N(u):=\{v\in V:\{u,v\}\in E\}$ to denote the set of neighbors of u. Given a graph G, a valid vertex coloring of G is a function $\varphi:V\to\mathbb{N}$ such that $\varphi(u)\neq\varphi(v)$ whenever $\{u,v\}\in E$. In [13], Erdős et. al introduced the notion of local colorings.

Definition 1 (Local Coloring). *Let k be a positive integer. A k-local coloring of a graph G is a valid vertex coloring φ such that for every vertex $u\in V$, $|\{\varphi(v):v\in N(u)\}|<k$.*

Note that the definition implies that in each closed neighborhood $\{u\}\cup N(u)$, the number of different colors is bounded by k. The *local chromatic number* $\psi(G)$ of a graph G is the minimum k such that G admits a k-local coloring [13]. Since any valid coloring with k colors also is a local k-coloring, clearly, $\psi(G)\leq\chi(G)$. Interestingly, it can also be shown that the local chromatic number is always at least as large as the fractional chromatic number, i.e., $\psi(G)\geq\chi_f(G)$ [26].

Given a valid vertex coloring φ and an integer i, let $C_i:=\{v\in V:\varphi(v)=i\}$ be the set of vertices with color i. Further, for integers $i\neq j$, we use $N_j(C_i):=$

$C_j \cap \bigcup_{v \in C_i} N(v)$ to denote the vertices with color j that have a neighbor with color i. We consider colorings with the following density condition.

Definition 2 (Biclique Coloring). *A coloring φ of a graph G is called a biclique coloring if for any two colors i and j, the subgraph induced by $N_i(C_j)$ and $N_j(C_i)$ is either empty or a complete bipartite graph.*

We will consider vertex cover in graphs for which a local biclique coloring φ is given. For any fixed $k \geq 3$, it is shown in [13] that there are n-vertex graphs with local chromatic number k and chromatic number $\Theta(\log \log n)$. This is shown using biclique colorings (cf. Def. 1.3 and Lemma 1.1 in [13]). Consequently, there are graphs that have chromatic number $\Theta(\log \log n)$ and admit a 3-local biclique coloring.

Contribution: In this paper we study the vertex cover problem in graphs with bounded local colorings. Assuming the UGC [23], the provided results give a tight characterization of the problem. The two main results are summarized as follows.

Theorem 1. *The vertex cover problem in graphs $G = (V, E)$ for which a $(\Delta+1)$-local biclique coloring φ of G is given as input admits a randomized polynomial-time algorithm with approximation ratio $2 - \Omega(1)\frac{\ln \ln \Delta}{\ln \Delta}$.*

Theorem 2. *Assuming the UGC, it is NP-hard to approximate the vertex cover problem in graphs for which a $(\Delta+1)$-local coloring is given as input, within any constant factor better than $2 - 2/(\Delta + 1)$.*

The result stated in Theorem 1 matches (up to the constant factor in the lower order term) a known inapproximability result [5]. In Section 3 we provide a matching upper bound for the inapproximability result of Theorem 2.

Besides generalizing the bounded degree case to a class of graphs with arbitrarily large chromatic number, we show that Theorem 1 finds a natural application in the precedence constrained single machine scheduling problem to minimize the weighted sum of completion times, known as $1|prec|\sum w_j C_j$ in standard scheduling notation. In a series of papers [1,9,10] it was established that this scheduling problem is a special case of minimum weighted vertex cover in graphs $G_{\mathbf{P}}$ of incomparable pairs defined in the dimension theory of partial orders. We prove the following in this paper.

Theorem 3. *For any graph of incomparable pairs $G_{\mathbf{P}}$, a $(\Delta + 1)$-local biclique coloring of $G_{\mathbf{P}}$ can be computed in polynomial time, where $\Delta - 1$ is the maximum number of predecessors (or successors) of each job.*

Together with Theorem 1, this improves the previously best $(2 - 2/\max\{\Delta, 2\})$-approximation algorithm described in [2]. Due to space limitations, omitted proofs will appear in the full version of the paper.

Review of the SDP Approach for Bounded Degree Graphs: In [22] the authors consider the problem of coloring graphs using semidefinite programming. Given a graph $G = (V, E)$ on n vertices, and a real number $k \geq 2$, a *vector k-coloring* [22] of G is an assignment of unit vectors $v_i \in \mathbb{R}^n$ to each

vertex $i \in V$, such that for any two adjacent vertices i and j the dot product of their vectors satisfies the inequality $v_i \cdot v_j \leq -1/(k-1)$. They show that it is possible to check if a graph admits a vector k-coloring by using semidefinite programming. Moreover, they prove that vector k-colorable graphs have a "large" independent set when k is "small".

Theorem 4 ([22]). *For every integral $k \geq 2$, a vector k-colorable graph $G = (V, E)$ with maximum degree d has an independent set I of value $\Omega(\frac{w(V)}{d^{1-2/k}\sqrt{\ln d}})$.*

In [25], it is proved that the following program is a semidefinite relaxation of the vertex cover problem. Moreover, it can be solved within an additive error of $\varepsilon > 0$ in polynomial time in $\ln \frac{1}{\varepsilon}$ and n using the ellipsoid method.

$$\begin{array}{ll} \min \sum_{u=1}^{n} w_u \frac{1+v_0 \cdot v_u}{2} & \\ \text{s.t. } (v_i - v_0)(v_j - v_0) = 0, & \{i,j\} \in E \\ \quad \| v_u \| = 1, & u \in V \cup \{0\}, v_u \in \mathbb{R}^{n+1} \end{array} \tag{1}$$

Note that in an "integral" solution of (1) (corresponding to a vertex cover), vectors for vertices that are picked coincide with v_0, while the other vectors coincide with $-v_0$. It is shown in [25] that the integrality gap is $2 - \varepsilon$, for any $\varepsilon > 0$, i.e., for every $\varepsilon > 0$ there is a graph G_ε such that $vc(G_\varepsilon)/sd(G_\varepsilon)$ is at least $2 - \varepsilon$, where $vc(G_\varepsilon)$ and $sd(G_\varepsilon)$ denote the minimum vertex cover value and the optimum value of (1).

Halperin [19] uses (1) to provide an efficient randomized algorithm that approximates vertex cover in graphs with maximum degree d. The improvement is obtained as follows by using a threshold rounding approach. Solve relaxation (1) and let $S_1 = \{u \in V \mid v_0 \cdot v_u \geq x\}$ and $S_2 = \{u \in V \mid -x \leq v_0 \cdot v_u < x\}$, where x is a small positive number. As in Hochbaum's approach [20], it holds that the vertices S_1 together with a cover of the subgraph induced by S_2 are sufficient to cover the whole graph G. Moreover, for any two adjacent vertices i and j in the graph $G[S_2]$ induced by S_2, (1) implies that $v_i \cdot v_j \leq -1 + 2x$. This has the important consequence that $G[S_2]$ is a vector k-colorable graph, where $k = \frac{2-2x}{1-2x}$ is close to 2 for small x. We can then use[1] Theorem 4 to obtain a large valued independent set I of $G[S_2]$. The returned vertex cover is $S_1 \cup (S_2 \setminus I)$ and the result of [19] follows by choosing a suitable value for x.

The problem we consider in this paper is a classical problem in scheduling theory, known as $1|prec|\sum w_j C_j$ in standard scheduling notation (see e.g. Graham et al. [16]). It is defined as the problem of scheduling a set $N = \{1, \ldots, n\}$ of n jobs on a single machine, which can process at most one job at a time. Each job j has a processing time p_j and a weight w_j, where p_j and w_j are nonnegative integers. Jobs also have precedence constraints between them that are specified in the form of a *partially ordered set* (*poset*) $\mathbf{P} = (N, P)$. The goal is to find a

[1] In Theorem 4 the integrality assumption on k is not technically necessary, and it can be easily generalized to fractional k. As remarked in [19], by using exactly the same analysis and rounding technique as in [22], it is possible to compute an independent set of value at least $\Omega(\frac{w(S_2)}{d^{x/(1-x)}\sqrt{x \ln d}})$ for $k = \frac{2-2x}{1-2x}$.

non-preemptive schedule which minimizes $\sum_{j=1}^{n} w_j C_j$, where C_j is the time at which job j completes in the given schedule.

The described problem was shown to be strongly NP-hard already in 1978 [27,28]. For the general version of $1|prec|\sum w_j C_j$, several 2-approximation algorithms are known [32,17,9,8,29]. Until recently, no inapproximability results were known, and closing the approximability gap has been listed as one of ten outstanding open problems in scheduling theory (e.g., [33]). In [4], it is proved that the problem does not admit a PTAS, assuming that NP-complete problems cannot be solved in randomized subexponential time. Moreover, if a fixed cost present in all feasible schedules is ignored then the problem is as hard to approximate as vertex cover [4]. Recently, Bansal and Khot [6] showed that the gap for the general problem indeed closes assuming a variant of the UGC [23].

In a series of papers [1,9,10] it was proved that $1|prec|\sum w_j C_j$ is a special case of minimum weighted vertex cover in some special graphs $G_{\mathbf{P}}$ that depend on the input poset $\mathbf{P}$. More precisely, it is shown that any feasible solution to the vertex cover problem in graphs $G_{\mathbf{P}}$ can be turned in polynomial time into a feasible solution to $1|prec|\sum w_j C_j$ without deteriorating the objective value. This result was achieved by investigating different integer LP formulations and relaxations [31,9,10] of $1|prec|\sum w_j C_j$, using linear ordering variables δ_{ij} such that the variable δ_{ij} has value 1 if job i precedes job j in the corresponding schedule, and 0 otherwise.

Dushnik and Miller [12] introduced dimension as a parameter of partial orders in 1941. There is a natural way to associate with a poset $\mathbf{P}$ a hypergraph $\mathbf{H_P}$, called the *hypergraph of incomparable pairs*, so that the dimension of $\mathbf{P}$ is the chromatic number of $\mathbf{H_P}$ [15]. Furthermore, the fractional dimension of $\mathbf{P}$, a generalization due to Brightwell and Scheinerman [7] is equal to the fractional chromatic number of $\mathbf{H_P}$. It turns out [3] that graph $G_{\mathbf{P}}$ is the (ordinary) graph obtained by removing from $\mathbf{H_P}$ all edges of cardinality larger than two. This allows to apply the rich vertex cover theory to $1|prec|\sum w_j C_j$ together with the dimension theory of partial orders. One can, e.g., conclude that the scheduling problem with two-dimensional precedence constraints is solvable in polynomial time, as $G_{\mathbf{P}}$ is bipartite in this case [15,10], and the vertex cover problem is well-known to be solvable in polynomial time on bipartite graphs. Further, these connections between the $1|prec|\sum w_j C_j$ and the vertex cover problem on $G_{\mathbf{P}}$, and between dimension and coloring, yield a framework for obtaining $(2-2/f)$-approximation algorithms for classes of precedence constraints with bounded (fractional) dimension f [2,1]. It yields the best known approximation ratios for all previously considered special classes of precedence constraints, like semi-orders, convex bipartite orders, interval orders, interval dimension 2, bounded in-degree posets.

2 Vertex Cover Using Bounded Local Biclique Colorings

In this section we provide and analyze an approximation algorithm for the vertex cover problem in graphs $G = (V, E)$ for which a $(\Delta + 1)$-local biclique coloring φ of G is given.

The presented approximation algorithm follows the threshold rounding approach used for the bounded-degree vertex cover problem [19]: first solve the SDP (1) and, based on this solution and a parameter x that will be determined later, two vertex sets, S_1 and S_2, are computed as follows.

$$S_1 = \{u \in V \mid v_0 \cdot v_u \geq x\} \quad \text{and} \quad S_2 = \{u \in V \mid -x \leq v_0 \cdot v_u < x, \}$$

The cover is obtained by picking the vertices in S_1 together with a cover of the subgraph induced by S_2.

However, unlike in [19], Theorem 4 does not apply to graph $G[S_2]$. Indeed, the graph degree is not bounded and the theorem does not generalize to these graphs since the rounding procedure and analysis in [22] are strongly based on the assumption that the graph has "few", i.e. $O(n)$ edges.

We show that graph $G[S_2]$ has a "large" independent set by using a new rounding procedure to compute it that works as follows. The vertices in S_2 are first grouped into overlapping *clusters*. For every two colors i and j (of the given coloring φ) such that the number of edges connecting vertices with the two colors is non-zero, there are two clusters $N_i(C_j)$ and $N_j(C_i)$. Note that $N_j(C_i) \neq \emptyset$ iff $N_i(C_j) \neq \emptyset$ in $G[S_2]$. Because we assume that the coloring φ is a local $(\Delta+1)$-coloring, each vertex $u \in V$ belongs to at most Δ clusters. Furthermore, for all i and j, clusters $N_i(C_j)$ and $N_j(C_i)$ are connected by complete bipartite sub-graphs (note that this property also holds when restricting the graph G to the vertex set S_2). It can be easily proved, that all vectors v_i corresponding to vertices from the same cluster almost point in the same direction. This essentially follows from having a biclique coloring and because, by the definition of the set S_2, vectors corresponding to adjacent vertices in S_2 are almost antipodal. For each cluster $N_i(C_j)$, we arbitrarily choose one *representative* vertex. Let R be the set of representatives of all clusters $N_i(C_j)$. Further, for a vertex u, let R_u be the set of representatives of clusters $N_i(C_j)$ for which $u \in N_i(C_j)$. By only using the vectors of the representatives, we compute a subset $I' \subseteq S_2$ as follows: vertex $u \in S_2$ belongs to I' if and only if every representative $a \in R_u$ of u satisfies $v_a \cdot r \geq c$, where c is a parameter and r is a random $(n+1)$-dimensional vector r from the $(n+1)$-dimensional standard normal distribution, i.e., the components of r are independent Gaussian random variables with mean 0 and variance 1. We show that the set I' has a large independent set. Formally, we have

$$I' = \left\{ u \in S_2 : \bigwedge_{a \in R_u} v_a \cdot r \geq c \right\}. \tag{2}$$

The complete procedure is summarized in Algorithm 1.

We first show that for all $u \in S_2$, the vectors corresponding to representatives of u's clusters point in almost the same direction as the vector v_u corresponding to vertex u.

1. Solve SDP (1)
2. Let $S_1 = \{u \in [n] \mid v_0 \cdot v_u \geq x\}$, $S_2 = \{u \in [n] \mid -x \leq v_0 \cdot v_u < x\}$.
3. Find an IS I in $G[S_2]$ as follows:
   (a) Compute the set R of representatives
   (b) Let $R_u = \{a \in R : \{u, a\} \subseteq N_i(C_j)$ for any two colors i and j of the given coloring $\varphi\}$
   (c) Choose a random vector r and define
   $$I' = \{u \in S_2 \mid \bigwedge_{a \in R_u} v_a \cdot r \geq c\}$$
   (d) Get I by removing one vertex of every edge in $G[I']$ from I'
4. Output the constructed vertex cover $S_1 \cup (S_2 \setminus I)$

Algorithm 1. Vertex Cover Approximation Algorithm

Lemma 1. *Let $u \in S_2$ be a vertex, $a \in R_u$ be the representative vertex of any cluster to which u belongs, and $x \in (0, 1)$. We have $v_u \cdot v_a \geq 1 - 8x + o(x)$.*

The expected size of the set I' is $\sum_{u \in S_2} \Pr[u \in I']$. Using (2), we have $\Pr[u \in I'] = \Pr\left[\bigwedge_{a \in R_u} v_a \cdot r \geq c\right]$. In the following we compute a lower bound on the above probability. The subsequent analysis uses some basic properties of the normal distribution. Let X be a standard normal random variable (X has mean 0 and variance 1). We use $\mathcal{N}(x)$ to denote the probability that X is at least x. Hence, $\mathcal{N}(x) := \Pr(X \geq x) = \int_x^\infty \phi(t)dt$, where $\phi(t) = e^{-t^2/2}/\sqrt{2\pi}$ is the density of X .

Lemma 2. *For any $u \in S_2$ and any constant $\gamma > 0$*

$$\Pr\left[\bigwedge_{a \in R_u} v_a \cdot r \geq c\right] \geq \mathcal{N}(\alpha c) - \Delta \cdot \mathcal{N}\left(\frac{c}{\beta\sqrt{x}}\right),$$

where $\alpha = \gamma + 1 + o(1)$ and $\beta = \frac{4+o(1)}{\gamma}$.

Proof. Assume, without loss of generality, that $v_u = (1, 0, \ldots, 0)$. By Lemma 1, for every $a \in R_u$ $v_u \cdot v_a \geq 1 - \delta$, where $\delta = 8x + o(x)$. Then the first component $v_{a,1}$ of $v_a = (v_{a,1}, v_{a,2}, \ldots, v_{a,n+1})$ must be at least $1 - \delta$. Let $A = (1 - \delta, 0, \ldots, 0)$ and $B_a = (0, v_{a,2}, \ldots, v_{a,n+1})$ for any $a \in R_u$.

Since v_a is a unit length vector, we have that $(1 - \delta)^2 + \sum_{j=2}^{n+1} v_{a,j}^2 \leq 1$, which which gives an upper bound on the length of B_a:

$$\| B_a \|^2 = \sum_{j=2}^{n+1} v_{a,j}^2 \leq 2\delta - \delta^2 < 2\delta.$$

We then get

$$\begin{aligned}
\Pr\left[\bigwedge_{a\in R_u} v_a \cdot r \geq c\right] &\geq \Pr\left[\bigwedge_{a\in R_u} (A\cdot r \geq (\gamma+1)c \wedge B_a \cdot r \geq -\gamma c)\right] \\
&= 1 - \Pr\left[A\cdot r < (\gamma+1)c \vee \bigvee_{a\in R_u} B_a \cdot r < -\gamma c\right] \\
&\geq 1 - \Pr\left[A\cdot r < (\gamma+1)c\right] - \sum_{a\in R_u} \Pr\left[B_a \cdot r < -\gamma c\right] \\
&= \Pr\left[\frac{A}{\|A\|}\cdot r \geq \frac{(\gamma+1)c}{\|A\|}\right] - \sum_{a\in R_u} \Pr\left[-\frac{B_a}{\|B_a\|}\cdot r > \frac{\gamma c}{\|B_a\|}\right] \\
&\geq \mathcal{N}\left(\frac{(\gamma+1)c}{1-\delta}\right) - \Delta\cdot\mathcal{N}\left(\frac{\gamma c}{\sqrt{2\delta}}\right) \\
&= \mathcal{N}(\alpha c) - \Delta\cdot\mathcal{N}\left(\frac{c}{\beta\sqrt{x}}\right).
\end{aligned}$$

The last inequality follows because the sum of k independent Gaussian random variables with variances $\sigma_1^2,\ldots,\sigma_k^2$ is a Gaussian random variable with variance $\sum_{i=1}^k \sigma_i^2$. Consequently, the the dot product of a unit vector with r is a standard normal random variable. □

The total weight of the vertices that are removed from I' is upper bounded by the weight of vertices of the edges in the graph $G[I']$ induced by I'. The next lemma bounds the probability that a vertex $u \in S_2$ is in I' and that u has a neighbor $u' \in S_2$ that is also in I'.

Lemma 3. *Consider a vertex $u \in S_2$. The probability that u as well as some neighbor u' of u are in I' is upper bound by*

$$\Pr\left[u \in I' \wedge \exists u' \in S_2 : \{u,u'\} \in E \wedge u' \in I'\right] \leq \Delta\cdot\mathcal{N}\left(\frac{c}{\sqrt{x}}\right).$$

Based on Lemmas 2 and 3, we can now lower bound the expected size of the computed independent set I and we can thus obtain a bound on the expected approximation ratio of Algorithm 1.

Theorem 5. *Choosing $x = \frac{1-o(1)}{25}\cdot\frac{\ln\ln\Delta}{\ln\Delta}$, $c = (1+o(1))\cdot\sqrt{\frac{2x}{1-25x}\ln\Delta}$, and $\gamma = 4$, Algorithm 1 has an expected approximation ratio of $2 - \frac{2-o(1)}{25}\cdot\frac{\ln\ln\Delta}{\ln\Delta}$.*

3 Vertex Cover in Graphs with Bounded Local Chromatic Number

Consider the vertex cover problem in graphs G_Δ for which a local $(\Delta+1)$-coloring is given. Theorem 6 shows that the approximation ratio achievable from relaxation (1) is no better than $2 - \frac{2}{\Delta+1}$ if only the bounded local colorability, but not the biclique condition holds.

Theorem 6. *For any fixed $\Delta \geq 2$ and $\varepsilon > 0$, there is a local $(\Delta+1)$-colorable graph $G_{\Delta,\varepsilon}$ for which*

$$\frac{vc(G_{\Delta,\varepsilon})}{sd(G_{\Delta,\varepsilon})} \geq 2 - \frac{2}{\Delta+1} - \varepsilon,$$

where $vc(G)$ and $sd(G)$ denote the size of a minimum weighted vertex cover of G and the solution value for the corresponding SDP relaxation (1), respectively.

Under the Unique Game Conjecture, Khot and Regev [24] proved that vertex cover is NP-hard to approximate better than $2-\varepsilon$, for any $\varepsilon > 0$.

Theorem 7 ([24]). *Assuming the Unique Game Conjecture, for arbitrarily small constants $\varepsilon, \delta > 0$, there is a polynomial time reduction mapping a SAT formula ϕ to an n-vertex graph G such that if ϕ is satisfiable then G has an independent set of size $(\frac{1}{2}-\varepsilon)n$ and if ϕ is unsatisfiable, then G has no independent set of size δn.*

By using Theorem 7, the reduction and the analysis in the proof of Theorem 6 can be easily adapted to obtain the following conditional hardness.

Theorem 8. *Assuming the Unique Game Conjecture, it is NP-hard to approximate the vertex cover problem in graphs for which a local $(\Delta+1)$-coloring is given as input, within any constant factor better than $2-2/(\Delta+1)$.*

A matching upper bound can be obtained by showing that an independent set of value at least $\frac{\sum_{i=1}^{n} w_i}{\Delta+1}$ is computable in polynomial time. Indeed, the following result shows that Turan's theorem (as proved by Caro and Wei) for bounded degree graphs can be generalized to graphs with bounded local colorings.

Theorem 9. *For any weighted graph $G=(V,E)$, there exists an independent set of value at least $\sum_{v\in V} \frac{w_v}{\Delta_v+1}$, where $\Delta_v = |\{c(u) : u \in N(v)\}|$ is the number of different colors in the neighborhood of $v \in V$ of any proper coloring c of G.*

By using standard techniques, an upper bound that matches the lower bound of Theorem 2 can be obtained in deterministic polynomial time.

Corollary 1. *There exists a $(2-\frac{2}{\Delta+1})$-approximation algorithm for the vertex cover problem in graphs for which a local $(\Delta+1)$-coloring is given as input.*

4 The Scheduling Application

Problem $1|prec|\sum w_jC_j$ is a classical and fundamental problem in scheduling theory [33]. Its complexity certainly depends on the poset complexity In particular, the dimension of the input poset has been established to be an important parameter for the approximability of the problem [2,1]. Another natural parameter of partial orders is given by the poset in- or out-degree [14], namely the job maximum number of predecessors or successors, respectively. One of the first NP-complete proofs [28] for $1|prec|\sum w_jC_j$ shows that the problem remains

strongly NP-hard even if every job has at most two predecessors (or successors) in the poset. In [2], the authors present a $(2-2/\max\{\Delta,2\})$-approximation algorithm, where $\Delta-1$ is the minimum between the in- and the out-degree of the input poset. In this section, we show how to use Theorem 1 to improve this by showing how to compute a Δ-local biclique coloring of $G_{\mathbf{P}}$.

Consider a *partially ordered set* $\mathbf{P}=(N,P)$. When neither $(x,y)\in P$ nor $(y,x)\in P$, we say that x and y are incomparable, denoted by $x||y$. We call $\text{inc}(\mathbf{P})=\{(x,y)\in N\times N : x||y \text{ in } P\}$ the set of *incomparable pairs* of $\mathbf{P}$. For any integer $k\geq 2$, a subset $S=\{(x_i,y_i) : 1\leq i\leq k\}\subset \text{inc}(\mathbf{P})$ is called an *alternating cycle* when $x_i\leq y_{i+1}$ in P, for all $i=1,2,\ldots,k$, and where $y_{k+1}=y_1$. An alternating cycle $S=\{(x_i,y_i) : 1\leq i\leq k\}$ is *strict* if $x_i\leq y_j$ in P if and only if $j=i+1$, for all $i,j=1,2,\ldots,k$.

For a poset $\mathbf{P}$, the *hypergraph of incomparable pairs of* $\mathbf{P}$ [15], denoted $\mathbf{H_P}=(V,E)$, is the hypergraph that satisfies the following conditions: (1) V is the set $\text{inc}(\mathbf{P})$ of incomparable pairs of $\mathbf{P}$; and (2) E consists of those subsets of V that form strict alternating cycles. The *graph* $G_{\mathbf{P}}$ *of incomparable pairs of* $\mathbf{P}$ is the ordinary graph determined by all edges of size 2 in $\mathbf{H_P}$ Hence, in $G_{\mathbf{P}}$, there is an edge between incomparable pairs (i,j) and (k,ℓ) if an only if $(i,\ell),(k,j)\in P$.

In a series of papers [1,10,9], it was proved that $1|prec|\sum w_jC_j$ is equivalent to a weighted vertex cover problem on the graph of incomparable pairs $G_{\mathbf{P}}$ of the poset $\mathbf{P}$ characterizing the precedence constraints of the scheduling problem. More precisely, given a scheduling instance S with precedence constraints $\mathbf{P}$, we need to consider the following *weighted* version $G^S_{\mathbf{P}}$ of $G_{\mathbf{P}}$. For all incomparable pairs $(i,j)\in\text{inc}(\mathbf{P})$, the weight of vertex (i,j) in $G_{\mathbf{P}}$ is $p_i\cdot w_j$, where p_i is the processing time of job i and w_j is the weight of process j.

Let $\mathbf{P}=(N,P)$ be a poset. For $j\in N$, define the *degree of* j $\deg(j)$ as the number of elements comparable (but not equal) to j in $\mathbf{P}$. Given $j\in N$, let $D(j)$ denote the set of all elements which are less than j, and $U(j)$ those which are greater than j in P. Let $\deg_D(j):=|D(j)|$ be the *in-degree* of j and the *maximum in-degree* $\Delta_D(\mathbf{P}):=\max\{\deg_D(j) : j\in N\}$. The *out-degree* of j $\deg_U(j)$ and the *maximum out-degree* $\Delta_U(\mathbf{P})$ are defined analogously (see also [14]). The maximum vertex degree in the graph of incomparable pairs $G_{\mathbf{P}}$ is bounded by $(\Delta_D(\mathbf{P})+1)\cdot(\Delta_U(\mathbf{P})+1)$. Hence, if both $\Delta_D(\mathbf{P})$ and $\Delta_U(\mathbf{P})$ are bounded, $G_{\mathbf{P}}$ has bounded degree and therefore, the bounded degree vertex cover approximation of [19] can be used to approximate the scheduling problem $1|prec|\sum w_jC_j$ with precedence constraints $\mathbf{P}$. If only either the in-degree or the out-degree of $\mathbf{P}$ is bounded, $G_{\mathbf{P}}$ does not have bounded degree. However, we will now show that in this case $G_{\mathbf{P}}$ has a good local biclique coloring.

Theorem 10. *Let* $\mathbf{P}=(N,P)$ *be a poset and let* $\Delta=1+\min\{\Delta_D(\mathbf{P}),\Delta_U(\mathbf{P})\}$. *Then, we can efficiently compute a* $(\Delta+1)$*-local biclique coloring of* $G_{\mathbf{P}}=(V,E)$.

Proof. We assume that $\Delta-1=\Delta_D(\mathbf{P})$ is the largest in-degree. The case $\Delta-1=\Delta_U(\mathbf{P})$ can be proven analogously. We first show how to compute a $(\Delta+1)$-local coloring of $G_{\mathbf{P}}$ (cf. Def. 1). Partition the incomparable pairs into $|N|$ color classes: $C_i=\{(i,j)\in\text{inc}(\mathbf{P})\}$ for $i\in[|N|]$. It is easy to check that every C_i forms an independent set. Moreover any incomparable pair $(i,j)\in\text{inc}(\mathbf{P})$ is adjacent to

$(k,\ell) \in \mathrm{inc}(\mathbf{P})$ if (necessary condition) $(k,j) \in P$. Since the in-degree of j is bounded by $\Delta - 1$, it follows that the number of distinct pairs (k,j) such that $(k,j) \in P$ is bounded by Δ (it is Δ and not $\Delta - 1$ because we also have to consider the pair $(j,j) \in P$). Therefore any incomparable pair (i,j) has neighbors in at most Δ clusters and thus the coloring is $(\Delta+1)$-local.

In order to show that the coloring is a biclique coloring (cf. Def. 2), assume that $\{(i,a),(j,c)\} \in E$ and $\{(i,b),(j,d)\} \in E$. The claim follows by proving that $\{(i,a),(j,d)\} \in E$ and $\{(i,b),(j,c)\} \in E$. By the assumption we have: $(i,c) \in P$, $(j,a) \in P$, $(i,d) \in P$ and $(j,b) \in P$. Since $(j,a) \in P \wedge (i,d) \in P$ we have $\{(i,a),(j,d)\} \in E$, and $(i,c) \in P \wedge (j,b) \in P$ implies $\{(i,b),(j,c)\} \in E$. □

Acknowledgments. The second author is indebted to Nikos Mutsanas and George Karakostas for many interesting discussions. We would like to thank Jiři Sgall for suggesting the term "biclique coloring." The research is supported by the Swiss National Science Foundation project 200020-122110/1 and by Hasler Foundation Grant 11099.

References

1. Ambühl, C., Mastrolilli, M.: Single machine precedence constrained scheduling is a vertex cover problem. Algorithmica 53(4), 488–503 (2009)
2. Ambühl, C., Mastrolilli, M., Mutsanas, N., Svensson, O.: Scheduling with precedence constraints of low fractional dimension. In: Fischetti, M., Williamson, D.P. (eds.) IPCO 2007. LNCS, vol. 4513, pp. 130–144. Springer, Heidelberg (2007)
3. Ambühl, C., Mastrolilli, M., Svensson, O.: Approximating precedence-constrained single machine scheduling by coloring. In: Díaz, J., Jansen, K., Rolim, J.D.P., Zwick, U. (eds.) APPROX 2006 and RANDOM 2006. LNCS, vol. 4110, pp. 15–26. Springer, Heidelberg (2006)
4. Ambühl, C., Mastrolilli, M., Svensson, O.: Inapproximability results for sparsest cut, optimal linear arrangement, and precedence constrained scheduling. In: FOCS, pp. 329–337 (2007)
5. Austrin, P., Khot, S., Safra, M.: Inapproximability of vertex cover and independent set in bounded degree graphs. In: IEEE Conference on Computational Complexity, pp. 74–80 (2009)
6. Bansal, N., Khot, S.: Optimal Long-Code test with one free bit. In: Foundations of Computer Science (FOCS), pp. 453–462 (2009)
7. Brightwell, G.R., Scheinerman, E.R.: Fractional dimension of partial orders. Order 9, 139–158 (1992)
8. Chekuri, C., Motwani, R.: Precedence constrained scheduling to minimize sum of weighted completion times on a single machine. Discrete Applied Mathematics 98(1-2), 29–38 (1999)
9. Chudak, F.A., Hochbaum, D.S.: A half-integral linear programming relaxation for scheduling precedence-constrained jobs on a single machine. Operations Research Letters 25, 199–204 (1999)
10. Correa, J.R., Schulz, A.S.: Single machine scheduling with precedence constraints. Mathematics of Operations Research 30(4), 1005–1021 (2005)
11. Dinur, I., Safra, S.: On the hardness of approximating minimum vertex cover. Annals of Mathematics 162(1), 439–485 (2005)

12. Dushnik, B., Miller, E.: Partially ordered sets. American Journal of Mathematics 63, 600–610 (1941)
13. Erdös, P., Füredi, Z., Hajnal, A., Komjáth, P., Rödl, V., Seress, Á.: Coloring graphs with locally few colors. Discrete Mathematics 59(1-2), 21–34 (1986)
14. Felsner, Trotter: On the fractional dimension of partially ordered sets. DMATH: Discrete Mathematics 136, 101–117 (1994)
15. Felsner, S., Trotter, W.T.: Dimension, graph and hypergraph coloring. Order 17(2), 167–177 (2000)
16. Graham, R., Lawler, E., Lenstra, J.K., Rinnooy Kan, A.H.G.: Optimization and approximation in deterministic sequencing and scheduling: A survey. Annals of Discrete Mathematics, vol. 5, pp. 287–326. North-Holland, Amsterdam (1979)
17. Hall, L.A., Schulz, A.S., Shmoys, D.B., Wein, J.: Scheduling to minimize average completion time: off-line and on-line algorithms. Mathematics of Operations Research 22, 513–544 (1997)
18. Halldórsson, M.M., Radhakrishnan, J.: Greed is good: Approximating independent sets in sparse and bounded-degree graphs. Algorithmica 18(1), 145–163 (1997)
19. Halperin, E.: Improved approximation algorithms for the vertex cover problem in graphs and hypergraphs. SIAM J. Comput. 31(5), 1608–1623 (2002)
20. Hochbaum, D.S.: Efficient bounds for the stable set, vertex cover and set packing problems. Discrete Applied Mathematics 6, 243–254 (1983)
21. Karakostas, G.: A better approximation ratio for the vertex cover problem. ACM Transactions on Algorithms 5(4) (2009)
22. Karger, D.R., Motwani, R., Sudan, M.: Approximate graph coloring by semidefinite programming. J. ACM 45(2), 246–265 (1998)
23. Khot, S.: On the power of unique 2-prover 1-round games. In: STOC, pp. 767–775 (2002)
24. Khot, S., Regev, O.: Vertex cover might be hard to approximate to within 2-epsilon. J. Comput. Syst. Sci. 74(3), 335–349 (2008)
25. Kleinberg, J.M., Goemans, M.X.: The lovász theta function and a semidefinite programming relaxation of vertex cover. SIAM J. Discrete Math. 11(2), 196–204 (1998)
26. Körner, J., Pilotto, C., Simonyi, G.: Local chromatic number and Sperner capacity. Journal on Combinatorial Theory, Series B 95(1), 101–117 (2005)
27. Lawler, E.L.: Sequencing jobs to minimize total weighted completion time subject to precedence constraints. Annals of Discrete Mathematics 2, 75–90 (1978)
28. Lenstra, J.K., Rinnooy Kan, A.H.G.: The complexity of scheduling under precedence constraints. Operations Research 26, 22–35 (1978)
29. Margot, F., Queyranne, M., Wang, Y.: Decompositions, network flows and a precedence constrained single machine scheduling problem. Operations Research 51(6), 981–992 (2003)
30. Nemhauser, G.L., Trotter, L.E.: Vertex packings: Structural properties and algorithms. Mathematical Programming 8, 232–248 (1975)
31. Potts, C.N.: An algorithm for the single machine sequencing problem with precedence constraints. Mathematical Programming Study 13, 78–87 (1980)
32. Schulz, A.S.: Scheduling to minimize total weighted completion time: Performance guarantees of LP-based heuristics and lower bounds. In: Cunningham, W.H., Queyranne, M., McCormick, S.T. (eds.) IPCO 1996. LNCS, vol. 1084, pp. 301–315. Springer, Heidelberg (1996)
33. Schuurman, P., Woeginger, G.J.: Polynomial time approximation algorithms for machine scheduling: ten open problems. Journal of Scheduling 2(5), 203–213 (1999)

Maximizing Polynomials Subject to Assignment Constraints

Konstantin Makarychev and Maxim Sviridenko

IBM Thomas J. Watson Research Center

Abstract. We study the q-adic assignment problem. We first give an $O(n^{(q-1)/2})$-approximation algorithm for the Koopmans–Beckman version of the problem improving upon the result of Barvinok. Then, we introduce a new family of instances satisfying "tensor triangle inequalities" and give a constant factor approximation algorithm for them. We show that many classical optimization problems can be modeled by q-adic assignment problems from this family. Finally, we give several integrality gap examples for the natural LP relaxations of the problem.

1 Introduction

In 1963 Lawler [13] suggested the following problem that he called q-adic assignment problem: We are given a set $\{1,\dots,n\} = [n]$ of indices and a $2q$-dimensional array $c_{u,v}$ for $u, v \in [n]^q$. The goal is to find a permutation $\pi : [n] \to [n]$ optimizing (maximizing or minimizing) the expression

$$\sum_{u\in[n]^q} c_{u,\pi(u)}, \tag{1}$$

where $\pi(u) = (\pi(u_1),\dots,\pi(u_q))$ for a vector $u = (u_1,\dots,u_q)$. We will call the maximization (minimization) problem the maximum (minimum) q-adic assignment problem. This problem naturally generalizes the famous quadratic assignment problem (QAP) originally defined by Koopmans and Beckman [12] for which $q = 2$ and $c_{u,v} = w_u d_v$ for all $u, v \in [n]^2$. The problem with decomposable coefficients $c_{u,v} = w_u d_v$ will be called the Koopmans–Beckman version of the q-adic assignment problem throughout the paper.

Similarly, to the quadratic assignment problem, the general q-adic assignment problem has many practical applications and can be used to model variety of optimization problems. For example, Winter and Zimmerman [18] used the cubic assignment problem for scheduling. Burkard, Cela and Klinz [7] suggested applications of the 4-adic (or quartic) assignment problem in VLSI synthesis. The Koopmans-Beckman version of the problem can also be viewed as a variant of the hypergraph isomorphism problem studied by Babai et al. [3] where we would like to maximize the product of weights of hyperedges that are mapped into each other. (However, since techniques used in hypergraph or graph isomorphism algorithms heavily use the fact that all hyperedges must be mapped these techniques cannot be used for the q-adic assignment problem.) But despite enormous amount of applications, the problem remains highly intractable both from

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 510–520, 2011.

theoretical [15] and practical viewpoints even when $q = 2$. For larger q, e.g. for $q = 4$, the problem cannot be solved exactly in practice even if $n = 14$ (see [6]).

1.1 Overview of the Results

In this paper, we concentrate on the maximization version of the problem with nonnegative coefficients. There are no known approximation algorithms that have a proven performance guarantee for the general case. The Koopmans–Beckman version of the maximum q-adic assignment problem was considered by Barvinok [4] who designed an approximation algorithm with performance guarantee $\varepsilon n^{q/2}$ and running time $n^{O(q/\varepsilon)}$ for any $\varepsilon > 0$. The best known approximation algorithm with performance guarantee $O(\sqrt{n})$ for the special case of $q = 2$, i.e. the Maximum QAP, was given in [15] (see also [16]). In this paper, we present a new $O(n^{(q-1)/2})$-approximation algorithm for the Koopmans–Beckman version of the maximum q-adic assignment problem thus improving upon the result of Barvinok [4] and matching the performance guarantee of [15] for the Maximum QAP. Note, that we cannot hope to get a poly-logarithmic approximation ratio, since such possibility was ruled out (under very plausible complexity assumptions) even for the maximum quadratic assignment problem in [15] (using the same method as in [15], one can show that under slightly stronger assumptions (see [5]), there exists $\delta > 0$ and q such that the maximum q-adic assignment problem is NP-hard to approximate within a factor of n^{δ}). Thus, one of our main goals is to identify special families of instances that can be solved with a much better performance guarantee. Arkin, Hassin and Sviridenko [2] and, then Nagarajan and Sviridenko [16] showed how to get a constant approximation ratio in the Koopmans–Beckman version of the maximum quadratic assignment problem, if the coefficients d_v satisfy the triangle inequality. We give a very general analog of this result for the general (i.e., not the Koopmans–Beckman version) maximum q-adic assignment problem. We further show that many combinatorial optimization problems can be expressed as special cases of this problem. In the full version of the paper, we give an integrality gap examples of $\Omega\left(\frac{n^{q-1}}{\ln n}\right)$ for the general case, and $\Omega\left(\frac{n^{(q-1)/2}}{\sqrt{\ln n}}\right)$ for the Koopmans–Beckman case of the maximum q-adic assignment problem.

1.2 Tensor Triangle Inequality

Let $K = \{k_1, \ldots, k_{q-1}\}$ where $k_1, \ldots, k_{q-1} \in [n]$ are arbitrary $(q-1)$ distinct indices, i.e. $k_a \neq k_b$ for all $a \neq b$. For each vector $v \in [n]^q$ and coordinate $s \in [q]$, let $\mathcal{V}_s(v, K)$ be the set of $(q-1)!$ vectors obtained from v by replacing all coordinates except the coordinate s with indices $k_1, \ldots, k_{q-1}$ in an arbitrary order. For example for $q = 3$, $s = 1$, vector $v = (v_1, v_2, v_3)$ and $K = \{k_1, k_2\} \subset [n]$ we get $\mathcal{V}_1(v, K) = \{(v_1, k_1, k_2), (v_1, k_2, k_1)\}$.

Definition 1.1. *We say that an instance of the maximum q-adic assignment problem satisfies the* tensor triangle inequality *if for every $K = \{k_1, \ldots, k_{q-1}\}$ and $u, v \in [n]^q$,*

$$c_{u,v} \le \sum_{s=1}^{q} \sum_{v' \in \mathcal{V}_s(v,K)} c_{uv'}. \tag{2}$$

Note that the total number of terms on the right hand side of (2) is $q!$ and the total number of different sets K is $\binom{n}{q-1}$. We now discuss various special cases of the tensor triangle inequality:

1. For $q = 2$ the 4-dimensional tensor $(c_{(i,j),(p,q)})$ for $i, j, p, q \in [n]$ satisfies the tensor triangle inequality if for any $r \in [n]$ the following inequality holds

$$c_{(i,j),(p,q)} \le c_{(i,j),(p,r)} + c_{(i,j),(r,q)}. \tag{3}$$

   In the Koopmans–Beckman version of the problem, $c_{(i,j),(p,q)} = w_{ij}d_{pq}$ and the condition (3) just becomes the triangle inequality $d_{pq} \le d_{pr} + d_{rq}$. If in addition d_{pq} is symmetric, i.e. $d_{pq} = d_{qp}$ we obtain the maximum quadratic assignment problem with the triangle inequality for which constant factor approximation algorithms are known [2,16].

   If w_{ij} is the weight of the edge (i, j) in a directed graph, $d_{pq} = 1$ if $p < q$ and $d_{pq} = 0$ if $p \ge q$, then the tensor triangle inequality is also satisfied, and the problem is exactly equivalent to the Maximum Acyclic Subgraph Problem.

   More generally, the condition (3) models the Koopmans–Beckman variant of the maximum quadratic assignment problem in which matrix d_{pq} is not symmetric but satisfies the triangle inequality. There was no constant factor algorithm known for this problem before and the techniques from [2,16] do not seem to generalize to this case.
2. In the Betweenness Problem, we are given the set T of triples (i, j, k), we would like to find a permutation π and maximize the number of triples such that either $\pi(i) < \pi(j) < \pi(k)$ or $\pi(i) > \pi(j) > \pi(k)$. This problem has applications in computational biology and was studied in [9,14]. This problem is modeled by the q-adic assignment problem in the Koopmans–Beckman form with $q = 3$. We define $c_{(i,j,k),(p,q,r)} = w_{(i,j,k)}d_{(p,q,r)}$ such that $w_{(i,j,k)} = 1$ for each $(i, j, k) \in T$, $w_{(i,j,k)} = 0$, for each $(i, j, k) \notin T$ and $d_{(p,q,r)} = 1$ if $p < q < r$ or $p > q > r$ and $d_{(p,q,r)} = 0$, otherwise. The condition (3) is satisfied since if $d_{pqr} = 1$ then at least one of the term one right hand side of (3) must be one for any choice of k_1, k_2. We note however that the random assignment gives 3 approximation for the Betweenness Problem, our algorithm gives a worse constant.
3. Finally, the condition (3) models the generalization of the Maximum Acyclic Subgraph Problem considered by Guruswami et al. [10]. In this problem for each customer we are given a list of preferences in the form $i_1 > i_2 > \cdots > i_q$ the goal is to find a permutation π that maximizes the total number of satisfied customers, where a customer is satisfied if his preferences are satisfied, i.e. $\pi(i_1) > \pi(i_2) > \cdots > \pi(i_q)$. Guruswami et al. [10] show that assuming the Unique Games Conjecture there is no approximation algorithm for this problem with performance guarantee better than $q!$ (a random assignment gives a $q!$ approximation). Since this problem can be modelled using

condition (3) it shows that our $e^2 q!$ approximation for the maximum q-adic assignment problem is optimal up to a factor e^2.

2 Linear Programming Relaxation

For any vector $u \in [n]^q$, let u_s be the s-th coordinate of vector u. The maximum q-adic assignment problem can be formulated as the problem of maximizing the value of degree q polynomial subject to assignment constraints:

$$\max \sum\nolimits_{u,v\in[n]^q} c_{u,v} \prod_{s=1}^{q} x_{u_s v_s}, \tag{4}$$

$$\sum\nolimits_{i\in[n]} x_{ij} = 1, \ \ j \in [n], \tag{5}$$

$$\sum\nolimits_{j\in[n]} x_{ij} = 1, \ \ i \in [n], \tag{6}$$

$$x_{ij} \in \{0, 1\}. \tag{7}$$

We linearize the objective function (4) and relax the integrality constraint (7). Let P_q be the set of all permutations on q elements. For each $u \in [n]^q$ and $\phi \in P_q$, let $u(\phi)$ denote a vector v with coordinates $v_s = u_{\phi(s)}$. In our linear programming relaxation we keep assignment variables x_{ij} for $i, j \in [n]$ and we define new variables $y_{u,v}$ for $u, v \in [n]^q$ that have the value $\prod_{s=1}^{q} x_{u_s v_s}$ for integer variables x_{ij}.

Let $\mathcal{F} \subseteq [n]^q$ be the set of index vectors such that for each $u \in \mathcal{F}$ there is a pair of coordinates s, s' and $u_s = u_{s'}$. The assignment constraints (5) and (6) imply that $\prod_{s=1}^{q} x_{u_s v_s} = 0$ if $u \in \mathcal{F}$ or $v \in \mathcal{F}$. In other words, the set $\mathcal{F}$ consists of infeasible vectors $u \in [n]^q$. Therefore, we can assume that $c_{u,v} = 0$ and add the constraint $y_{u,v} = 0$ if $u \in \mathcal{F}$ or $v \in \mathcal{F}$ to our linear programming relaxation. We now formally define our linear programming relaxation

$$\max \sum\nolimits_{u,v\in[n]^q} c_{u,v} y_{u,v}, \tag{8}$$

$$\sum\nolimits_{u\in[n]^q : u_s = i} y_{u,v} = x_{iv_s}, \ \ v \in [n]^q, s \in [q], i \in [n], \tag{9}$$

$$\sum\nolimits_{v\in[n]^q : v_s = j} y_{u,v} = x_{u_s j}, \ \ u \in [n]^q, s \in [q], j \in [n], \tag{10}$$

$$y_{u,v} = 0, \ \ u \in \mathcal{F} \text{ or } v \in \mathcal{F}, \tag{11}$$

$$y_{u,v} = y_{\phi(u),\phi(v)}, \ \ u, v \in [n]^q, \phi \in P_q, \tag{12}$$

$$\sum\nolimits_{i\in[n]} x_{ij} = 1, \ \ j \in [n], \tag{13}$$

$$\sum\nolimits_{j\in[n]} x_{ij} = 1, \ \ i \in [n], \tag{14}$$

$$y_{u,v} \geq 0, \ \ u, v \in [n]^q, \tag{15}$$

$$x_{ij} \geq 0, \ \ i, j \in [n]. \tag{16}$$

The constraints (9) are valid for an integral solution of the maximum q-adic assignment problem since $\sum_{u\in[n]^q:u_s=i} y_{u,v} = \sum_{u\in[n]^q:u_s=i} \prod_{s'=1}^{q} x_{u_{s'}v_{s'}}$ and we can iteratively apply the equations (13) for each coordinate of vector u except coordinate s. Analogously, we can show that constraints (10) are valid. The validity of constraints (12) follows from the fact that the product $\prod_{s=1}^{q} x_{u_s v_s}$ does not depend on the order of factors.

In the full version of the paper, we show that our linear programming relaxation has integrality gap $\Omega(n^{q-1}/\log n)$ for the general case of the maximum q-adic assignment problem. However, in the Koopmans–Beckman form, i.e. when $c_{u,v} = w_u d_v$ for all $u, v \in [n]^q$ the integrality gap is $O(n^{(q-1)/2})$. We present an approximation algorithm in Section 4. (An almost matching $\Omega(n^{(q-1)/2}/\sqrt{\log n})$-integrality gap example is presented in the full version of the paper.)

3 Problems Satisfying the Tensor Triangle Inequality

The following lemma provides us with an upper bound on the value of the linear programming relaxation.

Lemma 3.1. *Let (x^*, y^*) be an optimal solution of the linear programming relaxation (8)–(16). If the input instance of the maximum q-adic assignment problem satisfies the tensor triangle inequality, i.e. the inequality (2), then for the optimal value LP^* of the linear programming relaxation (8)–(16) we have*

$$LP^* \le \binom{n}{q-1}^{-1} \sum_{u,v\in[n]^q} c_{u,v} \left(\sum_{s=1}^{q} x^*_{u_s v_s} \right)$$

Proof. We apply the inequality (2) to each term in the expression $LP^* = \sum_{u,v\in[n]^q} c_{u,v} y^*_{u,v}$ and average over different choices of the set K. We obtain

$$\begin{aligned} LP^* &\le \sum_{u,v\in[n]^q} \left(\binom{n}{q-1}^{-1} \sum_{K:|K|=q-1} \sum_{s=1}^{q} \sum_{v'\in\mathcal{V}_s(v,K)} c_{uv'} \right) y^*_{u,v} \\ &= \binom{n}{q-1}^{-1} \sum_{u,v'\in[n]^q} c_{u,v'} \left(\sum_{s=1}^{q} \sum_{v\in[n]^q:v_s=v'_s} y^*_{u,v} \right) \\ &= \binom{n}{q-1}^{-1} \sum_{u,v\in[n]^q} c_{u,v} \left(\sum_{s=1}^{q} x_{u_s,v_s} \right), \end{aligned}$$

where the last equality follows from constraints (10). The first equality is derived by considering which terms in the expression $LP^* = \sum_{u,v\in[n]^q} c_{u,v} y^*_{u,v}$ generate the term $c_{u,v'} y^*_{u,v}$ after applying the tensor triangle inequality.

The next Lemma holds for the general case of the maximum q-adic assignment problem and will be used later in another context.

Lemma 3.2. *There exists a randomized polynomial time algorithm that finds an integral solution to the linear program (8)–(16) with expected value at least*

$$\frac{\left(1-\frac{1}{q}\right)^{2(q-1)}}{q!}\binom{n}{q-1}^{-1}\sum_{u,v\in[n]^q} c_{u,v}\left(\sum_{s=1}^{q} x^*_{u_s v_s}\right).$$

Proof. The matrix $X^* = (x^*_{ij})$ for $i,j \in [n]$ is doubly stochastic since it satisfies the constraints (13) and (14). Therefore, it can be represented as a convex combination of permutation matrices, i.e.

$$X^* = \sum_{\tau\in[N]} \lambda_\tau \Pi_\tau \tag{17}$$

where $N \le n^2$, $\sum_{\tau=1}^{N} \lambda_\tau = 1$, $\lambda_\tau \ge 0$ for all $\tau \in [N]$ and each Π_τ is a permutation matrix.

Our randomized algorithm chooses one permutation matrix with probability distribution given by coefficients λ_τ. The chosen permutation matrix corresponds to an integral solution of the linear program (8)–(16), it also corresponds to a random mapping $\varphi : [n] \to [n]$. Moreover, (17) implies that $\Pr[\varphi(i) = j] = x^*_{ij}$ for any $i,j \in [n]$. We also define the second random mapping $\psi : [n] \to [n]$ that corresponds to a permutation on $[n]$ chosen uniformly at random.

We now combine two random mappings φ and ψ. Choose the set $X \subseteq [n]$ at random such that each index $i \in [n]$ belongs to X independently with probability $1/q$. We define the mapping π as follows: if $i \in X$, then $\pi(i) = \varphi(i)$, if $i \notin X$ and $\psi(i) \notin \varphi(X)$, then $\pi(i) = \psi(i)$, all other elements are mapped in an arbitrary way.

We estimate probability, that $\pi(u) = v$ for $u, v \in [n]^q \setminus \mathcal{F}$. For every $s \in [q]$, let $A_s(u,v)$ be the following event

$$A_s(u,v) = \{\varphi(u_s) = v_s\} \bigwedge \{\psi(u_{s'}) = v_{s'} \text{ for all } s' \in [q], s' \ne s\}.$$

Since the permutations φ and ψ are defined independently, we have

$$\Pr[A_s(u,v)] = \frac{x^*_{u_s,v_s}}{(q-1)!}\binom{n}{q-1}^{-1}.$$

Let $B_s(u,v)$ be the event $B_s(u,v) = \{u_s \in X\} \bigwedge \{u_{s'} \notin X$ and $v_{s'} \notin \varphi(X)$ for all $s' \in [q], s' \ne s\}$. Since each $i \in [n]$ is chosen to the set X independently with probability $1/q$ and since $|\cup_{s'\ne s}\{u_{s'}, \varphi^{-1}(v_{s'})\}| \le 2(q-1)$ (always, and thus independently of $A_s(u,v)$) we have

$$\Pr\left[B_s(u,v) \mid A_s(u,v)\right] \ge \frac{1}{q}\left(1-\frac{1}{q}\right)^{2(q-1)}.$$

For every $s \in [q]$, events $A_s(u,v)$ and $B_s(u,v)$ together imply the event $\{\pi(u) = v\}$, also events $B_s(u,v)$ are disjoint for different $s \in [q]$, therefore

$$\Pr[\pi(u) = v] \ge \sum_{s=1}^{q} \Pr[A_s(u,v) \wedge B_s(u,v)] \ge \sum_{s=1}^{q} \frac{x^*_{u_s,v_s}}{(q-1)!}\binom{n}{q-1}^{-1} \times \frac{1}{q}\left(1-\frac{1}{q}\right)^{2(q-1)}.$$

We are now ready to estimate the expected value of the solution.

$$\mathbb{E}\left[\sum_{u\in[n]^q} c_{u,\pi(u)}\right] = \sum_{u,v\in[n]^q} c_{u,v}\Pr[\pi(u)=v]$$
$$\geq \sum_{u,v\in[n]^q} c_{u,v} \times \sum_{s=1}^{q} \frac{x^*_{u_s,v_s}}{(q-1)!}\binom{n}{q-1}^{-1} \times \frac{1}{q}\left(1-\frac{1}{q}\right)^{2(q-1)}$$
$$= \frac{\left(1-\frac{1}{q}\right)^{2(q-1)}}{q!}\binom{n}{q-1}^{-1} \sum_{u,v\in[n]^q} c_{u,v}\left(\sum_{s=1}^{q} x^*_{u_s,v_s}\right).$$

Combining Lemmas 3.1 and 3.2 we obtain the following theorem.

Theorem 3.1. *There exists a randomized approximation algorithm with performance guarantee $e^2 q!$ for the maximum q-adic assignment problem satisfying the generalized triangle inequality (2).*

In the full version of the paper, we also prove the following result.

Theorem 3.2. *There exists a randomized approximation algorithm for the maximum q-adic assignment problem (1) that finds a solution with expected value at least $LP^*/(e^2(q-1)!\binom{n}{q-1})$, i.e. the performance guarantee of the algorithm is $e^2(q-1)!\binom{n}{q-1} = O(n^{q-1})$.*

4 Koopmans–Beckman Case: The Problem with Decomposable Coefficients

We now give an algorithm for the Koopmans–Beckman variant of the problem (an extension of our earlier work [15]). In this case, all weights $c_{u,v}$ have the form $c_{u,v} = w_u d_v$. It is convenient to think that vectors $u, v \in [n]^q$ are hyperedges and w_u and d_v are weights of these hyperedges. We write $i \in u$, if for some s, $i = u_s$. If $u' \in [n]^q$ and $u'' \in [n]^q$ do not have common vertices $i \in [n]$, then we say that u' and u'' are disjoint.

Theorem 4.1. *There exists a polynomial-time randomized approximation algorithm for the Koopmans–Beckman version of the q-adic assignment problem with the approximation ratio $O(q^2 n^{(q-1)/2})$.*

Proof. The input of the algorithm is a collection of weights $\{w_u : u \in [n]^q\}$ and $\{d_v : v \in [n]^q\}$. The output is a permutation $\pi : [n] \to [n]$. Fix a parameter

$$M = \left\lceil \frac{1}{q}\sqrt{(q-1)!\binom{n}{q-1}} \right\rceil = \left\lceil \frac{1}{q}\sqrt{\frac{n!}{(n-q+1)!}} \right\rceil.$$

Solve the linear program. Denote the solution by (x^*, y^*) and the value of the objective function by LP^*. Now, greedily find a set of disjoint "heavy" (according to the LP) hyperedges $\mathcal{E} = \{e \in [n]^q\}$ using the following algorithm.

– **Initialization:** Set $\mathcal{U} = [n]^q$; $\mathcal{E} = \varnothing$.
– **while** ($\mathcal{U}$ is not empty)
  • Find the most expensive edge $e \in \mathcal{U}$ with respect to the cost function

$$\text{lp-cost}(e) = \sum_{v \in [n]^q} w_e d_v y^*_{e,v}.$$

  • Add e to the set $\mathcal{E}$.
  • For every vertex $i \in e$, we denote $e(i) = e$.
  • For every $i \in e$, and $s \in \{1, \ldots, q\}$: we iteratively, set $\mathcal{A}_{i,s} = \{u \in \mathcal{U} : u_s = i\}$ and then remove elements of $\mathcal{A}_{i,s}$ from the set $\mathcal{U}$. Thus, we partition all hyperedges in $\mathcal{U}$ that intersect with e into disjoint sets $\mathcal{A}_{i,s}$ (where $i \in e$, $s \in \{1, \ldots, q\}$), such that for every $u \in \mathcal{A}_{i,s}$, $u_s = i$. Note also that deleting hyperedges from $\mathcal{U}$ we ensure that each $e(i)$ is well-defined and each vertex i has at most one hyperedge $e(i)$ associated with it. Moreover, each hyperedge belongs to exactly one set $\mathcal{A}_{i,s}$.
  • Let $\mathcal{B}_{i,s}$ be the set of M hyperedges u from $\mathcal{A}_{i,s}$ with the maximum value of w_u. If $|\mathcal{A}_{i,s}| < M$, then let $\mathcal{B}_{i,s} = \mathcal{A}_{i,s}$.

Note, that $\cup_{i \in [n]} \cup_{s \in \{1,\ldots,q\}} \mathcal{A}_{i,s} = [n]^q$ and $\mathcal{A}_{i,s} \cap \mathcal{A}_{i',s'} = \varnothing$ for $(i,s) \neq (i',s')$.

By convention, we define $\mathcal{A}_{i,s} = \varnothing$ if i does not belong to any edge $e \in \mathcal{E}$. Then we have

$$LP^* = \sum_{u \in [n]^q} \sum_{v \in [n]^q} w_u d_v y^*_{u,v} = \sum_{i \in [n]} \sum_{s=1}^{q} \sum_{u \in \mathcal{A}_{i,s}} \sum_{v \in [n]^q} w_u d_v y^*_{u,v}$$

$$= \underbrace{\sum_{i \in [n]} \sum_{s=1}^{q} \sum_{u \in \mathcal{A}_{i,s} \setminus \mathcal{B}_{i,s}} \sum_{v \in [n]^q} w_u d_v y^*_{u,v}}_{LP^*_I} + \underbrace{\sum_{i \in [n]} \sum_{s=1}^{q} \sum_{u \in \mathcal{B}_{i,s}} \sum_{v \in [n]^q} w_u d_v y^*_{u,v}}_{LP^*_{II}}.$$

Denote the first term in the sum LP^*_I and the second term LP^*_{II}. We now consider two cases. In each case we run a separate approximation algorithm. Our final algorithm could be viewed as an algorithm that runs these two approximation algorithms and chooses the best solution.

I. First, assume, that $LP^*_I \geq LP^*/2$. Let $m_{i,s} = \min\{w_u : u \in \mathcal{B}_{i,s}\}$, if $|\mathcal{B}_{i,s}| = M$; and $m_{i,s} = 0$, otherwise. Note, that

$$\max\{w_u : u \in \mathcal{A}_{i,s} \setminus \mathcal{B}_{i,s}\} \leq m_{i,s} \leq \frac{1}{M} \sum_{u \in \mathcal{B}_{i,s}} w_u. \tag{18}$$

We have

$$LP^*_I = \sum_{i \in [n]} \sum_{s=1}^{q} \sum_{u \in \mathcal{A}_{i,s} \setminus \mathcal{B}_{i,s}} \sum_{v \in [n]^q} w_u d_v y^*_{u,v} \leq \sum_{i \in [n]} \sum_{s=1}^{q} \sum_{u \in \mathcal{A}_{i,s} \setminus \mathcal{B}_{i,s}} \sum_{v \in [n]^q} m_{i,s} d_v y^*_{u,v}$$

$$\leq \sum_{i \in [n]} \sum_{s=1}^{q} \sum_{u \in [n]^q : u_s = i} \sum_{v \in [n]^q} m_{i,s} d_v y^*_{u,v} = \sum_{i \in [n]} \sum_{s=1}^{q} \sum_{v \in [n]^q} m_{i,s} d_v x^*_{i,v_s}.$$

In the last equality, we used LP constraint (9). Then, using upper bound (18) on $m_{i,s}$, we get

$$LP_I^* = \sum_{i\in[n]}\sum_{s=1}^{q}\sum_{v\in[n]^q} m_{i,s} d_v x^*_{i,v_s} \le \sum_{i\in[n]}\sum_{s=1}^{q}\sum_{v\in[n]^q}\left(\frac{1}{M}\sum_{u\in\mathcal{B}_{i,s}} w_u\right) d_v x^*_{i,v_s}$$
$$\le \frac{1}{M}\sum_{i\in[n]}\sum_{s=1}^{q}\sum_{u:u_s=i}\sum_{v\in[n]^q} w_u d_v x^*_{i,v_s} = \frac{1}{M}\sum_{s=1}^{q}\sum_{u\in[n]^q}\sum_{v\in[n]^q} w_u d_v x^*_{u_s,v_s}.$$

Hence,

$$\sum_{s=1}^{q}\sum_{u,v\in[n]^q} w_u d_v x^*_{u_s,v_s} \ge \frac{1}{2} M \cdot LP^*.$$

By Lemma 3.2, there exists a randomized polynomial-time algorithm that finds a solution of cost

$$ALG_I \ge \frac{(1-\frac{1}{q})^{2(q-1)}}{q!}\binom{n}{q-1}^{-1}\sum_{u,v\in[n]^q} w_u d_v \left(\sum_{s=1}^{q} x^*_{u_s v_s}\right)$$
$$\ge \frac{1}{2e^2}\frac{\frac{1}{q}\sqrt{(q-1)!\binom{n}{q-1}}}{q!\binom{n}{q-1}} LP^* \ge \frac{LP^*}{2e^2 q^2 n^{\frac{q-1}{2}}}.$$

II. Now, assume, that $LP_{II}^* \ge LP^*/2$. For every $e \in \mathcal{E}$, select independently a random $v \in [n]^q$ with probability $y^*_{e,v}$, and set $\phi(e) = v$. Then, pick elements $e \in \mathcal{E}$ in a random order and for each $s = 1, \dots, q$ set $\pi(e_s) = \phi(e)_s$, if the vertex $\phi(e)_s$ does not have a preimage yet; and discard the vertex e_s, otherwise. In the end, map all discarded vertices in an arbitrary way.

We need to estimate the probability that $\pi(e) = v$. The probability that $\phi(e) = v$ equals $y^*_{e,v}$. Let $N(v)$ be the number of e' for which $\phi(e')$ intersects v. Then we can estimate the expectation

$$E[N(v) \mid \phi(e) = v] \le 1 + \sum_{e'\in\mathcal{E}}\sum_{v':v'\cap v\neq\varnothing} y^*_{e'v'}$$
$$= 1 + \sum_{e'\in\mathcal{E}}\sum_{s=1}^{q}\sum_{s'=1}^{q}\sum_{v':v'_{s'}=v_s} y^*_{e'v'} = 1 + \sum_{e'\in\mathcal{E}}\sum_{s=1}^{q}\sum_{s'=1}^{q} x^*_{e'_{s'}v_s}.$$

Recall, that all hyperedges e' in $\mathcal{E}$ are disjoint. Thus the expression above can be bounded above by $1 + \sum_{i=1}^{n}\sum_{s=1}^{q} x^*_{iv_s} = 1 + q$. This bound implies that $\Pr[\pi(e) = v] = Pr[\phi(e) = v] \cdot E\left[\frac{1}{N(v)} \mid \phi(e) = v\right] \ge y^*_{ev}/(q+1)$, where the last inequality follows from the Jensen's inequality for convex function $1/x$. Hence, the cost of the solution returned by the algorithm is at least

$$ALG_{II} \ge \sum_{e\in\mathcal{E}}\sum_{v\in[n]^q} w_u d_v \frac{y^*_{ev}}{q+1}.$$

On the other hand (using $|B_{i,s}| \leq M$),

$$LP^*_{II} = \sum_{i=1}^{n}\sum_{s=1}^{q}\sum_{u\in\mathcal{B}_{i,s}} \underbrace{\left(\sum_{v\in[n]^q} w_u d_v y^*_{u,v}\right)}_{\text{lp-cost}(u)} \leq \sum_{i=1}^{n}\sum_{s=1}^{q}\sum_{u\in\mathcal{B}_{i,s}} \underbrace{\left(\sum_{v\in[n]^q} w_{e(i)} d_v y^*_{e(i),v}\right)}_{\text{lp-cost}(e(i))}$$

$$\leq qM \sum_{e\in\mathcal{E}}\sum_{i\in e}\left(\sum_{v\in[n]^q} w_e d_v y^*_{e,v}\right) = q^2 M \sum_{e\in\mathcal{E}}\left(\sum_{v\in[n]^q} w_e d_v y^*_{e,v}\right).$$

Combining the inequalities, we get

$$ALG_{II} \geq \frac{LP^*_{II}}{q^2(q+1)M} \geq \frac{LP^*}{4n^{\frac{(q-1)}{2}}q(q+1)}.$$

Thus, we have showed that the approximation ratio of the algorithm is $O(q^2 n^{\frac{(q-1)}{2}})$.

References

1. Adams, W.P., Johnson, T.A.: Improved Linear Programming-based Lower Bounds for the Quadratic Assignment Problem. DIMACS Series in Discrete Mathematics and Theoretical Computer Science 16, 43–77 (1994)
2. Arkin, E., Hassin, R., Sviridenko, M.: Approximating the Maximum Quadratic Assignment Problem. Information Processing Letters 77, 13–16 (2001)
3. Babai, L., Codenotti, P.: Isomorphism of Hypergraphs of Low Rank in Moderately Exponential Time. In: Proc. 39th Ann. IEEE Symp. on Theory of Computing (FOCS 2008), pp. 667–676. IEEE Comp. Soc. Press, Los Alamitos (2008)
4. Barvinok, A.: Estimating L^∞ norms by L^{2k} norms for functions on orbits. Found. Comput. Math. 2(4), 393–412 (2002)
5. Bellare, M., Goldwasser, S., Lund, C., Russel, A.: Efficient Probabilistically Checkable Proofs and Applications to Approximation. In: Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing (STOC 1993), pp. 294–304 (1993)
6. Burkard, R.E., Cela, E.: Heuristics for biquadratic assignment problems and their computational comparison. European Journal of Operations Research 83, 283–300 (1995)
7. Burkard, R., Cela, E., Klinz, B.: On the biquadratic assignment problem. In: Quadratic Assignment and Related Problems, New Brunswick, NJ. DIMACS Ser. Discrete Math. Theoret. Comput. Sci., vol. 16, pp. 117–146. Amer. Math. Soc., Providence (1994)
8. Cela, E.: The Quadratic Assignment Problem: Theory and Algorithms. Springer, Heidelberg (1998)
9. Chor, B., Sudan, M.: A Geometric Approach to Betweenness. SIAM J. Discrete Math. 11(4), 511–523 (1998)
10. Guruswami, V., Håstad, J., Manokaran, R., Raghavendra, P., Charikar, M.: Beating the Random Ordering is Hard: Every ordering CSP is approximation resistant. Electronic Colloquium on Computational Complexity (ECCC) 18, 27 (2011)

11. Hassin, R., Levin, A., Sviridenko, M.: Approximating the minimum quadratic assignment problems. ACM Transactions on Algorithms 6(1) (2009)
12. Koopmans, T.C., Beckman, M.: Assignment problems and the location of economic activities. Econometrica 25, 53–76 (1957)
13. Lawler, E.: The quadratic assignment problem. Management Science 9, 586–599 (1962/1963)
14. Makarychev, Y.: Simple Linear Time Approximation Algorithm for Betweenness, Microsoft Research Technical Report MSR-TR-2009-74
15. Makarychev, K., Manokaran, R., Sviridenko, M.: Maximum Quadratic Assignment Problem: Reduction from Maximum Label Cover and LP-based Approximation Algorithm. In: Abramsky, S., Gavoille, C., Kirchner, C., Meyer auf der Heide, F., Spirakis, P.G. (eds.) ICALP 2010. LNCS, vol. 6198, pp. 594–604. Springer, Heidelberg (2010)
16. Nagarajan, V., Sviridenko, M.: On the maximum quadratic assignment problem. Mathematics of Operations Research 34(4), 859–868 (2009)
17. Queyranne, M.: Performance ratio of polynomial heuristics for triangle inequality quadratic assignment problems. Operations Research Letters 4, 231–234 (1986)
18. Winter, T., Zimmermann, U.: Real-time dispatch of trams in storage yards, Valparaiso. Mathematics of industrial systems, vol. IV (1996); Ann. Oper. Res. 96, 287–315 (2000)

A Polynomial-Time Algorithm for Estimating the Partition Function of the Ferromagnetic Ising Model on a Regular Matroid*

Leslie Ann Goldberg[1] and Mark Jerrum[2]

[1] Department of Computer Science, University of Liverpool, Ashton Building, Liverpool L69 3BX, United Kingdom
[2] School of Mathematical Sciences Queen Mary, University of London, Mile End Road, London E1 4NS, United Kingdom

Abstract. We investigate the computational difficulty of approximating the partition function of the ferromagnetic Ising model on a regular matroid. Jerrum and Sinclair have shown that there is a fully polynomial randomised approximation scheme (FPRAS) for the class of graphic matroids. On the other hand, the authors have previously shown, subject to a complexity-theoretic assumption, that there is no FPRAS for the class of binary matroids, which is a proper superset of the class of graphic matroids. In order to map out the region where approximation is feasible, we focus on the class of regular matroids, an important class of matroids which properly includes the class of graphic matroids, and is properly included in the class of binary matroids. Using Seymour's decomposition theorem, we give an FPRAS for the class of regular matroids.

1 Introduction

Classically, the Potts model [10] in statistical physics is defined on a graph. Let q be a positive integer and $G = (V, E)$ a graph with edge weights $\boldsymbol{\gamma} = \{\gamma_e : e \in E\}$; the weight $\gamma_e > -1$ represents a "strength of interaction" along edge e. The q-state Potts partition function specified by this weighted graph is

$$Z_{\text{Potts}}(G; q, \boldsymbol{\gamma}) = \sum_{\sigma: V \to [q]} \prod_{e=\{u,v\} \in E} \big(1 + \gamma_e \, \delta(\sigma(u), \sigma(v))\big), \tag{1}$$

where $[q] = \{1, \ldots, q\}$ is a set of q spins or colours, and $\delta(s, s')$ is 1 if $s = s'$, and 0 otherwise. The partition function is a sum over "configurations" σ which assign spins to vertices in all possible ways. We are concerned with the computational complexity of approximately evaluating the partition function (1) and generalisations of it. For reasons that will become apparent shortly, we shall be concentrating on the case $q = 2$, which is the familiar Ising model. In this special case, the two spins correspond to two possible magnetisations at a vertex (or "site"), and the edges (or "bonds") model interactions between

* This work was partially supported by the EPSRC grant "Computational Counting"

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 521–532, 2011.

sites. In the *ferromagnetic* case, when $\gamma_e > 0$, for all $e \in E$, the configurations σ with many adjacent like spins make a greater contribution to the partition function $Z_{\text{Potts}}(G; q, \boldsymbol{\gamma})$ than those with few; in the *antiferromagnetic* case, when $-1 < \gamma_e < 0$, the opposite is the case.

An equivalent way of looking at (1) is as a restriction of the (multivariate) Tutte polynomial, which is defined as follows:

$$\widetilde{Z}(G; q, \boldsymbol{\gamma}) = \sum_{A \subseteq E} \gamma_A \, q^{\kappa(A)-|V|}, \tag{2}$$

where $\gamma_A = \prod_{e \in A} \gamma_e$ and $\kappa(A)$ denotes the number of connected components in the graph (V, A) [12, (1.2)]. This is perhaps not the most usual expression for the multivariate Tutte polynomial of a graph, but it conveniently generalises to the Tutte polynomial of a matroid (4), also [12, (1.3)], which is the main subject of this article. Although (2) and (1) are formally quite different, they agree when q is a positive integer, up to a factor of $q^{-|V|}$.

Jaeger, Vertigan and Welsh [7] were the first to consider the computational complexity of computing the Tutte polynomial. They considered the classical bivariate Tutte polynomial in which the edge weights are constant, i.e., $\gamma_e = \gamma$ for all $e \in E$. Their approach was to fix q and γ, and consider the computational complexity of computing (2) as a function of the instance graph G. Jaeger et al. showed, amongst other things, that computing the Tutte polynomial exactly is #P-hard when $q > 1$ and $\gamma \in (-1, \infty) - \{0\}$. In particular, this means that the partition function (1) of the Potts model is computationally intractable, unless P = #P.

In the light of this intractability result, it is natural to consider the complexity of approximate computation in the sense of "fully polynomial approximation scheme" or FPRAS. Before stating the known results, we quickly define the relevant concepts. A *randomised approximation scheme* is an algorithm for approximately computing the value of a function $f : \Sigma^* \to \mathbb{R}$. The approximation scheme has a parameter $\varepsilon > 0$ which specifies the error tolerance. A *randomised approximation scheme* for f is a randomised algorithm that takes as input an instance $x \in \Sigma^*$ (e.g., for the problem of computing the bivariate Tutte polynomial of a graph, the graph G) and a rational error tolerance $\varepsilon > 0$, and outputs a rational number z (a random variable of the "coin tosses" made by the algorithm) such that, for every instance x,

$$\Pr\left[e^{-\varepsilon} f(x) \leq z \leq e^{\varepsilon} f(x)\right] \geq \frac{3}{4}. \tag{3}$$

The randomised approximation scheme is said to be a *fully polynomial randomised approximation scheme*, or *FPRAS*, if it runs in time bounded by a polynomial in $|x|$ and ε^{-1}.

For the problem of computing the bivariate Tutte polynomial in the antiferromagnetic situation, i.e., $-1 < \gamma < 0$, there is no FPRAS for (2) unless RP = NP [3]. This perhaps does not come as a great surprise, since, in the special case $\gamma = -1$, the equivalent expression (1) counts proper colourings of a graph.

So we are led to consider the ferromagnetic case, $\gamma > 0$. Even here, Goldberg and Jerrum [2] have recently provided evidence of computational intractability (under a complexity theoretic assumption that is stronger than RP $\neq$ NP) when $q > 2$.

The sequence of results so far described suggest we should focus on the special case $q = 2$ and $\gamma > 0$, i.e., the ferromagnetic Ising model. Here, at last, there is a positive result to report, as Jerrum and Sinclair [8] have presented an FPRAS for the partition function (2), with $q = 2$ and arbitrary positive weights $\boldsymbol{\gamma}$. As hinted earlier, the Tutte polynomial makes perfect sense in the much wider context of an arbitrary matroid (see Sections 2 and 3 for a quick survey of matroid basics, and of the Tutte polynomial in the context of matroids). The Tutte polynomial of a graph is merely the special case where the matroid is restricted to be graphic. It is natural to ask whether the positive result of [8] extends to a wider class of matroids than graphic. One extension of graphic matroids is to the class of binary matroids. Goldberg and Jerrum [4] recently provided evidence of computational intractability of the ferromagnetic Ising model on binary matroids, under the same strong complexity-theoretic assumption mentioned earlier.

Sandwiched between the graphic (computationally easy) and binary matroids (apparently computationally hard) is the class of regular matroids. Since it is interesting to locate the exact boundary of tractability, we consider here the computational complexity of estimating (in the FPRAS sense) the partition function of the Ising model on a regular matroid. We show that there is an FRPAS in this situation.

Theorem 1. *There is an FPRAS for the following problem.*

Instance. *A binary matrix representing a regular matroid $\mathcal{M}$. A set $\boldsymbol{\gamma} = \{\gamma_e : e \in E(\mathcal{M})\}$ of non-negative rational edge weights.*

Output. *$\widetilde{Z}(\mathcal{M}; 2, \boldsymbol{\gamma})$. (The extension of the Tutte polynomial $\widetilde{Z}$ to matroids is formally defined in (4)).*

Aside from the existing FPRAS for graphic matroids, which also works, by duality, for so-called cographic matroids, the main ingredient in our algorithm is Seymour's decomposition theorem for regular matroids. This theorem has been applied on at least one previous occasion to the design of a polynomial-time algorithm. Golynski and Horton [6] use the approach in their algorithm for finding a minimum-weight basis of the cycle space (or circuit space) of a regular matroid. The decomposition theorem states that every regular matroid is either graphic, cographic, a special matroid on 10 elements named R_{10}, or can be decomposed as a certain kind of sum (called "1-sum", "2-sum" or "3-sum") of two smaller regular matroids (see Theorem 2). Since we know how to handle the base cases (graphic, cographic and R_{10}), it seems likely that the decomposition theorem will yield a polynomial time algorithm quite directly. However, there is a catch (which does not arise in [6]). When we pull apart a regular matroid into two smaller ones, say into the 3-sum of $\mathcal{M}_1$ and $\mathcal{M}_2$, four subproblems are generated for each of the parts $\mathcal{M}_1$ and $\mathcal{M}_2$. This is fine if the decomposition is fairly balanced at each step, but that is not always the case. In the case of a highly unbalanced decomposition, we face a combinatorial explosion.

The solution we adopt is to "solve" recursively only the smaller subproblem, say $\mathcal{M}_2$. Then we construct a constant size matroid $\mathcal{I}_3$ that we show is equivalent to $\mathcal{M}_2$ in the context of any 3-sum. We then glue $\mathcal{I}_3$ onto $\mathcal{M}_1$, using the 3-sum operation, in place of $\mathcal{M}_2$. The matroid $\mathcal{I}_3$ is small, just six elements, and has the property that forming the 3-sum with $\mathcal{M}_1$ leaves $\mathcal{M}_1$ unchanged as a matroid, though it acquires some new weights from $\mathcal{I}_3$. (In a sense, $\mathcal{I}_3$ is an identity for the 3-sum operation.) Then we just have to find the partition function of $\mathcal{M}_1$ (with amended weights). Since we have four recursive calls on "small" subproblems, but only one on a "large" one, we achieve polynomially bounded running time.

The fact that $\mathcal{I}_3$ is able to simulate the behaviour of an arbitrary regular matroid in the context of a 3-sum is a fortunate accident of the specialisation to $q = 2$. Since the existing algorithm for the graphic case also makes particular use of the fact that $q = 2$, one gets the impression that the Ising model has very special properties compared with the Potts model in general.

Several technical proofs that are omitted from this extended abstract can be found in the full version of the paper [5].

2 Matroid Preliminaries

A matroid is a combinatorial structure that has a number of equivalent definitions, but the one in terms of a rank function is the most natural here. A set E (the "ground set") together with a rank function $r : E \to \mathbb{N}$ is said to be a *matroid* if the following conditions are satisfied for all subsets $A, B \subseteq E$: (i) $0 \leq r(A) \leq |A|$, (ii) $A \subseteq B$ implies $r(A) \leq r(B)$ (monotonicity), and (iii) $r(A \cup B) + r(A \cap B) \leq r(A) + r(B)$ (submodularity). A subset $A \subseteq E$ satisfying $r(A) = |A|$ is said to be *independent*; a maximal (with respect to inclusion) independent set is a *basis*, and a minimal dependent set is a *circuit*. A circuit with one element is a *loop*. We denote the ground set of matroid $\mathcal{M}$ by $E(\mathcal{M})$ and its rank function by $r_{\mathcal{M}}$. To every matroid $\mathcal{M}$ there is a *dual matroid* $\mathcal{M}^*$ with the same ground set $E = E(\mathcal{M})$ but rank function $r_{\mathcal{M}^*}$ given by $r_{\mathcal{M}^*}(A) = |A| + r_{\mathcal{M}}(E - A) - r_{\mathcal{M}}(E)$. A *cocircuit* in $\mathcal{M}$ is a set that is a circuit in $\mathcal{M}^*$; equivalently, a cocircuit is a minimal set that intersects every basis. A cocircuit with one element is a *coloop*. A thorough exposition of the fundamentals (and beyond) of matroid theory can be found in Oxley's book [9].

Important operations on matroids include contraction and deletion. Suppose $T \subseteq E$ is any subset of the ground set of matroid $\mathcal{M}$. The *contraction* $\mathcal{M}/T$ *of* T *from* $\mathcal{M}$ is the matroid on ground set $E - T$ with rank function given by $r_{\mathcal{M}/T}(A) = r_{\mathcal{M}}(A \cup T) - r_{\mathcal{M}}(T)$, for all $A \subseteq E - T$. The *deletion* $\mathcal{M}\backslash T$ *of* T *from* $\mathcal{M}$ is the matroid on ground set $E - T$ with rank function given by $r_{\mathcal{M}/T}(A) = r_{\mathcal{M}}(A)$, for all $A \subseteq E - T$. These operations are often combined, and we write $\mathcal{M}/T\backslash S$ for the matroid obtained by contracting T from $\mathcal{M}$ and then deleting S from the result. The operations of contraction and deletion are dual in the sense that $(\mathcal{M}\backslash T)^* = \mathcal{M}^*/T$. For compactness, we shall often miss out set brackets, writing $\mathcal{M}/p_1\backslash p_2, p_3$, for example, in place of $\mathcal{M}/\{p_1\}\backslash\{p_2, p_3\}$. The *restriction* $\mathcal{M} \mid S$ of $\mathcal{M}$ to $S \subseteq E$ is the matroid on ground set S that

inherits its rank function from $\mathcal{M}$; another way of expressing this is to say $\mathcal{M} \mid S = \mathcal{M}\backslash(E - S)$.

The matroid axioms are intended to abstract the notion of linear independence of vectors. Some matroids can be represented concretely as a matrix M with entries from a field K, the columns of the matrix being identified with the elements of E. The rank $r(A)$ of a subset $A \subseteq E$ is then just the rank of the submatrix of M formed from columns picked out by A. A matroid that can be specified in this way is said to be *representable over* K. A matroid that is representable over GF(2) is *binary*, and one that is representable over every field is *regular*; the regular matroids form a proper subclass of binary matroids. Another important class of matroids are ones that arise as the *cycle matroid* of an undirected (multi)graph $G = (V, E)$. Here, the edge set E of the graph forms the ground set of the matroid, and the rank of a subset $A \subseteq E$ is defined to be $r(A) = |V| - \kappa(A)$, where $\kappa(A)$ is the number of connected components in the subgraph (V, A). A matroid is *graphic* if it arises as the cycle matroid of some graph, and cographic if its dual is graphic. Both the class of graphic matroids and the class of cographic matroids are strictly contained in the class of regular matroids.

Now for some definitions more specific to the work in this article. A *cycle* in a matroid is any subset of the ground set that can be expressed as a disjoint union of circuits. We let $\mathcal{C}(\mathcal{M})$ denote the set of cycles of a matroid $\mathcal{M}$. If $\mathcal{M}$ is a binary matroid, the symmetric difference of any two cycles is again a cycle [9, Thm. 9.1.2(vi)], so $\mathcal{C}(\mathcal{M})$, viewed as a set of characteristic vectors on $E(\mathcal{M})$, forms a vector space over GF(2), which we refer to as the *circuit space* of $\mathcal{M}$. Indeed, any vector space generated by a set of vectors in $\mathrm{GF}(2)^E$ can be regarded as the circuit space of a binary matroid on ground set E (see the remark following [9, Cor. 9.2.3]). The set of cycles $\mathcal{C}(\mathcal{M})$ of a matroid $\mathcal{M}$ determines the set of circuits of $\mathcal{M}$ (these being just the minimal non-empty cycles), which in turn determines $\mathcal{M}$. Note, from the definition of "cycle", that

$$\mathcal{C}(\mathcal{M}\backslash T) = \{C \in \mathcal{C}(\mathcal{M}) \mid C \subseteq E\backslash T\}.$$

The term "cycle" in this context is not widespread, but is used by Seymour [11] in his work on matroid decomposition; the term "circuit space" is more standard.

Consider two binary matroids $\mathcal{M}_1$ and $\mathcal{M}_2$ with $E(\mathcal{M}_1) = E_1 \cup T$ and $E(\mathcal{M}_2) = E_2 \cup T$ and $E_1 \cap E_2 = \emptyset$. The *delta-sum* of $\mathcal{M}_1$ and $\mathcal{M}_2$ is the matroid $\mathcal{M}_1 \triangle \mathcal{M}_2$ on ground set $E_1 \cup E_2$ with the following circuit space:

$$\mathcal{C}(\mathcal{M}_1 \triangle \mathcal{M}_2) = \big\{C \subseteq E_1 \cup E_2 : C = C_1 \oplus C_2, \\ \text{for some } C_1 \in \mathcal{C}(\mathcal{M}_1) \text{ and } C_2 \in \mathcal{C}(\mathcal{M}_2)\big\},$$

where $\oplus$ denotes symmetric difference [11]. (The right-hand side of the above equation defines a vector space over GF(2), and hence does describe the circuit space of some matroid.) A situation of particular interest occurs when $\mathcal{M}_1 \mid T = \mathcal{M}_2 \mid T = N$ (say), i.e., $\mathcal{M}_1$ and $\mathcal{M}_2$ have the same restriction, namely N, to T. The special case of the delta-sum when $T = \emptyset$ is called the *1-sum* of $\mathcal{M}_1$ and $\mathcal{M}_2$.

(It is just the direct sum of the two matroids.) The special case when T is a singleton that is not a loop or coloop in $\mathcal{M}_1$ or $\mathcal{M}_2$, and $|E(\mathcal{M}_1)|, |E(\mathcal{M}_2)| \geq 3$, is called the *2-sum* of $\mathcal{M}_1$ or $\mathcal{M}_2$ and is denoted $\mathcal{M}_1 \oplus_2 \mathcal{M}_2$. Finally, the special case when $|T| = 3$, T is a circuit in both $\mathcal{M}_1$ and $\mathcal{M}_2$ but contains no co-circuit of either, and $|E(\mathcal{M}_1)|, |E(\mathcal{M}_2)| \geq 7$, is called the *3-sum* of $\mathcal{M}_1$ or $\mathcal{M}_2$ and is denoted $\mathcal{M}_1 \oplus_3 \mathcal{M}_2$ [11].

Our main tool is the following celebrated result of Seymour [11, (14.3)]:

Theorem 2. *Every regular matroid $\mathcal{M}$ may be constructed by means of 1- 2- and 3-sums, starting with matroids each isomorphic to a minor of $\mathcal{M}$, and each either graphic, cographic or isomorphic to a certain 10-element matroid R_{10}.*

It is important for us that a polynomial-time algorithm exists for finding the decomposition promised by this theorem. As Truemper notes [14], such an algorithm is given implicitly in Seymour's paper. However, much more efficient algorithms are known. In particular, [13, (10.6.1)] gives a polynomial-time algorithm for testing whether a binary matroid is graphic or cographic. Also, [14] gives a cubic algorithm for expressing any regular matroid that is not graphic, cographic or isomorphic to R_{10} as a 1-sum, 2-sum or 3-sum. While Seymour's decomposition theorem is in terms of 1-sums, 2-sums and 3-sums, it will be convenient for us to do our preparatory work, in the following section, using the slightly more general notion of a delta-sum.

3 Tutte Polynomial and Decomposition

Suppose $\mathcal{M}$ is a matroid, q is an indeterminate, and $\boldsymbol{\gamma} = \{\gamma_e : e \in E(\mathcal{M})\}$ is a collection of indeterminates, indexed by elements of the ground set of $\mathcal{M}$. The (multivariate) *Tutte polynomial* of $\mathcal{M}$ and $\boldsymbol{\gamma}$) is defined to be

$$\widetilde{Z}(\mathcal{M}; q, \boldsymbol{\gamma}) = \sum_{A \subseteq E(\mathcal{M})} \gamma_A \, q^{-r_{\mathcal{M}}(A)}, \tag{4}$$

where $\gamma_A = \prod_{e \in A} \gamma_e$ [12, (1.3)]. In this article, we are interested in the *Ising model*, which corresponds to the specialisation of the above polynomial to $q = 2$, so we shall usually omit the parameter q in the above notation and assume that q is set to 2. As we are concerned with approximate computation, we shall invariably be working in a environment in which each of the indeterminates γ_e is assigned some real value; furthermore, we shall be focusing on the *ferromagnetic* case, in which those values or *weights* are all non-negative. For convenience, we refer to the pair $(\mathcal{M}, \boldsymbol{\gamma})$ as a "weighted matroid", and we will assume throughout that $\gamma_e \geq 0$ for all $e \in E(\mathcal{M})$. For background material on the multivariate Tutte polynomial, and its relation to the classical 2-variable Tutte polynomial, refer to Sokal's expository article [12].

In order to exploit Theorem 2, we need to investigate how definition (4) behaves when $\mathcal{M}$ is a delta-sum of two matroids. The interaction between the classical *bivariate* Tutte polynomial and 2- and 3-sums has been investigated

previously, for example by Andrzejak [1]. In principle, it would be possible to assure oneself that his proof carries over from the bivariate to the multivariate situation, and then recover the appropriate formulas (such as the one in Lemma 1 below) by appropriate algebraic translations. However, in the case of the 3-sum there is an obstacle. While Andrzejak's identities remain valid, as identities of rational functions on q, they become degenerate (through division by zero) under the specialisation $q = 2$. In fact, this degeneracy is crucial to us, in that it reduces the dimension of the bilinear form in Lemma 1 from five, as in Andrzejak's general result, to four. The significance of this reduction in dimension will become apparent in Section 4. In light of these considerations, and because certain intermediate results in our proofs are in any case required later, we derive the required formulas in the full version of the paper [5]. Here, to save space, we not only omit these proofs, but also state the result that relates to the 3-sum (i.e., the case $|T| = 3$); the statement in the case of a 2-sum has a similar form, but is simpler. Denote by D the matrix

$$D = \begin{pmatrix} 4 & -1 & -1 & -1 \\ -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{pmatrix}.$$

Lemma 1. *Suppose that $(\mathcal{M}_1, \boldsymbol{\gamma}_1)$ and $(\mathcal{M}_2, \boldsymbol{\gamma}_2)$ are weighted binary matroids with $E(\mathcal{M}_1) \cap E(\mathcal{M}_2) = T = \{p_1, p_2, p_3\}$, and suppose also that T is a circuit in both $\mathcal{M}_1$ and $\mathcal{M}_2$. For $i = 1, 2$, let*

$$z^i = \big(\widetilde{Z}(\mathcal{M}_i \backslash T; \boldsymbol{\gamma}_i), \widetilde{Z}(\mathcal{M}_i / p_1 \backslash p_2, p_3; \boldsymbol{\gamma}_i), \widetilde{Z}(\mathcal{M}_i / p_2 \backslash p_1, p_3; \boldsymbol{\gamma}_i), \widetilde{Z}(\mathcal{M}_i / p_3 \backslash p_1, p_2; \boldsymbol{\gamma}_i)\big)^{\mathrm{T}}.$$

Denote by $\boldsymbol{\gamma} = \boldsymbol{\gamma}_1 \triangle \boldsymbol{\gamma}_2$ the weighting on $E(\mathcal{M}_1 \triangle \mathcal{M}_2)$ inherited from $\mathcal{M}_1$ and $\mathcal{M}_2$. Then

$$\widetilde{Z}(\mathcal{M}_1 \triangle \mathcal{M}_2; \boldsymbol{\gamma}) = (z^1)^{\mathrm{T}} D z^2. \tag{5}$$

4 Signatures

The goal of this section is to show that, for every weighted binary weighted matroid $(\mathcal{M}, \boldsymbol{\gamma})$ with distinguished elements p_1, p_2, p_3, there is a small (6-element) weighted matroid $(\mathcal{I}_3, \boldsymbol{\delta})$ that is equivalent to $\mathcal{M}$ in the following sense: if we replace $(\mathcal{M}, \boldsymbol{\gamma})$ by $(\mathcal{I}_3, \boldsymbol{\delta})$ in the context of any 3-sum, the Tutte polynomial (specialised to $q = 2$) of the 3-sum is changed by a factor that is independent of the context. Moreover, the weighted matroid $(\mathcal{I}_3, \boldsymbol{\delta})$ can readily be computed given a "signature" of $\mathcal{M}$. There is also a 2-element weighted matroid $(\mathcal{I}_2, \boldsymbol{\delta})$ that does a similar job for 2-sums, but to save space we state the result for 3-sums only.

For a weighted binary matroid $(\mathcal{M}, \boldsymbol{\gamma})$ with distinguished elements $T = \{p_1, p_2, p_3\}$, the *signature* of $(\mathcal{M}, \boldsymbol{\gamma})$ with respect to T is the vector

$$\sigma(\mathcal{M}; T, \boldsymbol{\gamma}) = \frac{\big(\widetilde{Z}(\mathcal{M}/p_1 \backslash p_2, p_3; \boldsymbol{\gamma}), \widetilde{Z}(\mathcal{M}/p_2 \backslash p_1, p_3; \boldsymbol{\gamma}), \widetilde{Z}(\mathcal{M}/p_3 \backslash p_1, p_2; \boldsymbol{\gamma})\big)}{\widetilde{Z}(\mathcal{M} \backslash T; \boldsymbol{\gamma})}.$$

What we are seeking in the 3-sum case is a small matroid whose signature is equal to that of a given binary matroid $\mathcal{M}$. Such a matroid will be equivalent to $\mathcal{M}$ in the context of any 3-sum. Before describing such a matroid we give two technical lemmas that investigate inequalities between the Tutte polynomial of various minors of a binary matroid. These inequalities will restrict the domain of possible signatures that can occur.

Lemma 2. *Suppose that $(\mathcal{M}, \boldsymbol{\gamma})$ is a weighted binary matroid with distinguished elements $T = \{p_1, p_2, p_3\}$, and that T is a circuit in $\mathcal{M}$. Let*

$$\begin{aligned} z^{\mathrm{T}} &= (z_0, z_1, z_2, z_3, z_4) \\ &= \big(\widetilde{Z}(\mathcal{M}\backslash T; \boldsymbol{\gamma}), \widetilde{Z}(\mathcal{M}/p_1\backslash p_2, p_3; \boldsymbol{\gamma}), \\ &\qquad \widetilde{Z}(\mathcal{M}/p_2\backslash p_1, p_3; \boldsymbol{\gamma}), \widetilde{Z}(\mathcal{M}/p_3\backslash p_1, p_2; \boldsymbol{\gamma}), \widetilde{Z}(\mathcal{M}/T; \boldsymbol{\gamma})\big). \end{aligned}$$

Then the following (in)equalities hold: (i) $z_0 > 0$, (ii) $z_0 \leq z_1, z_2, z_3 \leq 2z_0$, (iii) $\frac{1}{2}z_4 < z_1, z_2, z_3 \leq z_4$ and (iv) $z_1 + z_2 + z_3 = 2z_0 + z_4$.

These inequalities arise as a by-product of the proof of Lemma 1.

Lemma 3. *Under the same assumptions as Lemma 2, $z_1z_2, z_1z_3, z_2z_3 \leq z_0z_4$.*

These inqualities are derived from the Fortuin-Kasteleyn-Ginibre (FKG) inequality. For details, consult the full paper [5].

Denote by $\mathcal{I}_3$ the 6-element matroid with ground set $\{p_1, p_2, p_3, e_1, e_2, e_3\}$, whose circuit space is generated by the circuits $\{p_1, e_1\}$, $\{p_2, e_2\}$, $\{p_3, e_3\}$, and $\{p_1, p_2, p_3\}$. The matroid $\mathcal{I}_3$ can be thought of as the cycle matroid of a certain graph, namely, the graph with parallel pairs of edges $\{p_1, e_1\}$, $\{p_2, e_2\}$ and $\{p_3, e_3\}$ in which edges p_1, p_2 and p_3 form a length-3 cycle in the graph.

Let $T = \{p_1, p_2, p_3\}$. We start by showing that, as long as a signature (s_1, s_2, s_3) satisfies certain equations, which Lemmas 2 and 3 will guarantee, then it is straightforward to compute a weighting $\boldsymbol{\delta}$ so that the weighted matroid $(\mathcal{I}_3, \boldsymbol{\delta})$ has signature $\sigma(\mathcal{I}_3; T, \boldsymbol{\gamma}) = (s_1, s_2, s_3)$.

Lemma 4. *Suppose s_1, s_2 and s_3 satisfy: (i) $2+s_1-s_2-s_3 > 0$, $2-s_1+s_2-s_3 > 0$, $2-s_1-s_2+s_3 > 0$, (ii) $s_1+s_2+s_3-3 \geq 0$, and (iii) $s_1+s_2+s_3-s_2s_3-2 \geq 0$, $s_1+s_2+s_3-s_1s_3-2 \geq 0$ and $s_1+s_2+s_3-s_1s_2-2 \geq 0$; then there are non-negative weights d_1, d_2 and d_3 such that, for any weight function $\boldsymbol{\delta}$ with $\delta_{e_1} = d_1$, $\delta_{e_2} = d_2$ and $\delta_{e_3} = d_3$, $\sigma(\mathcal{I}_3; T, \boldsymbol{\delta}) = (s_1, s_2, s_3)$. The values d_1, d_2 and d_3 can be computed from s_1, s_2 and s_3.*

Proof. Define

$$\begin{aligned} S_1 &= 2 + s_1 - s_2 - s_3 \\ S_2 &= 2 - s_1 + s_2 - s_3 \\ S_3 &= 2 - s_1 - s_2 + s_3 \\ R &= s_1 + s_2 + s_3 - 2, \end{aligned}$$

and define the weights d_1, d_2, d_3 for $\boldsymbol{\delta}$ as follows:

$d_1 = -1 + \sqrt{RS_1/S_2S_3}$, $d_2 = -1 + \sqrt{RS_2/S_1S_3}$, and $d_3 = -1 + \sqrt{RS_3/S_1S_2}$.

A calculation presented in the full paper verifies that these are the required weights.

Temporarily leaving aside the issue of approximation, the way that we will use Lemma 4 is captured in the following corollary.

Corollary 1. *Suppose that* $(\mathcal{M}_1, \boldsymbol{\gamma}_1)$ *and* $(\mathcal{M}_2, \boldsymbol{\gamma}_2)$ *are weighted binary matroids with* $E(\mathcal{M}_1) \cap E(\mathcal{M}_2) = T = \{p_1, p_2, p_3\}$*, and suppose also that* T *is a circuit in both* $\mathcal{M}_1$ *and* $\mathcal{M}_2$*. Let* $(s_1, s_2, s_3) = \sigma(\mathcal{M}_2; T, \boldsymbol{\gamma}_2)$*. Then there are non-negative weights* d_1*,* d_2 *and* d_3 *such that, for any weight function* $\boldsymbol{\delta}$ *with* $\delta_{e_1} = d_1$*,* $\delta_{e_2} = d_2$ *and* $\delta_{e_3} = d_3$*,*

$$\widetilde{Z}(\mathcal{M}_1 \triangle \mathcal{M}_2; \boldsymbol{\gamma}_1 \triangle \boldsymbol{\gamma}_2) = \zeta \widetilde{Z}(\mathcal{M}_1 \triangle \mathcal{I}_3; \boldsymbol{\gamma}_1 \triangle \boldsymbol{\delta}),$$

where

$$\zeta = \widetilde{Z}(\mathcal{M}_2 \backslash T; \boldsymbol{\gamma}_2) / \widetilde{Z}(\mathcal{I}_3 \backslash T; \boldsymbol{\delta}) = \widetilde{Z}(\mathcal{M}_2 \backslash T; \boldsymbol{\gamma}_2) / \sqrt{R/S_1S_2S_3},$$

in the notation of the proof of Lemma 4. The values d_1*,* d_2 *and* d_3 *can be computed from* s_1*,* s_2 *and* s_3 *— they do not otherwise depend upon* $(\mathcal{M}_1, \boldsymbol{\gamma}_1)$ *or* $(\mathcal{M}_2, \boldsymbol{\gamma}_2)$*. Moreover, the values* R*,* S_1*,* S_2 *and* S_3 *are byproducts of this computation.*

The problem with using Corollary 1 to replace the complicated expression $\widetilde{Z}(\mathcal{M}_1 \triangle \mathcal{M}_2; \boldsymbol{\gamma}_1 \triangle \boldsymbol{\gamma}_2)$ with the simpler $\widetilde{Z}(\mathcal{M}_1 \triangle \mathcal{I}_3; \boldsymbol{\gamma}_1 \triangle \boldsymbol{\delta})$ is that, in general, we will not be able to compute the necessary values s_1, s_2 and s_3 exactly. Instead, we will use our FPRAS recursively to approximate these values. Thus, we need a version of Corollary 1 (omitted from this paper, but given in the full version [5]) that accommodates some approximation error. The technical complication to be faced is that the approximations to s_1, s_2 and s_3 that we obtain may not correspond to the signature of any actual matroid.

We also need a matroid, $\mathcal{I}_2$, and results, analogous to but simpler than those mentioned above, which are appropriate for 2-sums, but we omit those here.

We complete this section by investigating the (simple) way in which the special matroids $\mathcal{I}_2$ and $\mathcal{I}_3$ interact with delta-sums with $|T| = 1$ and $|T| = 3$, respectively.

Suppose $(\mathcal{M}, \boldsymbol{\gamma})$ is a weighted binary matroid with distinguished elements $T = \{p_1, p_2, p_3\}$, where T is a cycle in $\mathcal{M}$. Consider the delta-sum $\mathcal{M} \triangle \mathcal{I}_3$. As before, there is a natural correspondence between the ground sets of the two matroids. Let C_1, C_2, C_3 be the three 2-circuits in $\mathcal{I}_3$ including elements p_1, p_2, p_3, respectively. Any cycle C in $\mathcal{M}$ can be transformed in a unique way to a cycle C' in $\mathcal{M} \triangle \mathcal{I}_3$, by adding a subset of circuits from $\{C_1, C_2, C_3\}$. The mapping $C \mapsto C'$ is invertible, and is a bijection between cycles in $\mathcal{M}$ and those in $\mathcal{M} \triangle \mathcal{I}_3$. Now let $\boldsymbol{\gamma}'$ be derived from $\boldsymbol{\gamma}$ by assigning $\gamma_{p_1} = \delta_{e_1}$, $\gamma_{p_2} = \delta_{e_2}$ and $\gamma_{p_3} = \delta_{e_3}$. Then $\widetilde{Z}(\mathcal{M} \triangle \mathcal{I}_3, \boldsymbol{\gamma} \triangle \boldsymbol{\delta}) = \widetilde{Z}(\mathcal{M}, \boldsymbol{\gamma}')$.

Again, a similar observation applies to $\mathcal{I}_2$ under delta-sum with $|T| = 1$, but we omit the details.

Step 1. If $\mathcal{M}$ is graphic, cographic or R_{10} then estimate $\widetilde{Z}(\mathcal{M}, \boldsymbol{\gamma})$ directly.
Step 2. Otherwise use Seymour's decomposition algorithm to express $\mathcal{M}$ as $\mathcal{M}_1 \triangle \mathcal{M}_2$, where $\triangle$ is a 1-, 2- or 3-sum. Recall that $E(\mathcal{M}) = E(\mathcal{M}_1) \oplus E(\mathcal{M}_2)$. Let $T = E(\mathcal{M}_1) \cap E(\mathcal{M}_2)$, $E_1 = E(\mathcal{M}_1) - T$ and $E_2 = E(\mathcal{M}_2) - T$. Noting $E(\mathcal{M}) = E_1 \cup E_2$, let $\boldsymbol{\gamma}_1 : E_1 \to \mathbb{R}^+$ and $\boldsymbol{\gamma}_2 : E_2 \to \mathbb{R}^+$ be the restrictions of $\boldsymbol{\gamma}$ to E_1 and E_2. Assume without loss of generality that $|E(\mathcal{M}_2)| \leq |E(\mathcal{M}_1)|$ (otherwise swap their names).
Step 3. If $\triangle$ is s 3-sum then let $T = \{p_1, p_2, p_3\}$ (say). Execute Steps 4a–7a in Figure 2. If $\triangle$ is a 2-sum then let $T = \{p\}$. Execute Steps 4b–7b of the algorithm. (The details of these steps are omitted from this extended abstract, but they are similar in structure to Steps 4a–7a.) If $\triangle$ is a 1-sum then recursively estimate $\widetilde{Z}(\mathcal{M}_1; \boldsymbol{\gamma}_1)$ with accuracy parameter $\varepsilon|\mathcal{M}_1|/|\mathcal{M}|$, and $\widetilde{Z}(\mathcal{M}_2; \boldsymbol{\gamma}_2)$ with accuracy parameter $\varepsilon|\mathcal{M}_2|/|\mathcal{M}|$, and return the product of the two, which is an estimate of $\widetilde{Z}(\mathcal{M}, \boldsymbol{\gamma})$.

Fig. 1. Algorithm for estimating the Ising partition function of a regular matroid $\mathcal{M}$ given accuracy parameter $\varepsilon \leq 1$

5 The Algorithm

We now have all the ingredients for the algorithm for estimating $\widetilde{Z}(\mathcal{M}; \boldsymbol{\gamma}) = \widetilde{Z}(\mathcal{M}; 2, \boldsymbol{\gamma})$, given a weighted regular matroid $(\mathcal{M}, \boldsymbol{\gamma})$ and an accuracy parameter ε. The base cases for this recursive algorithm are when $\mathcal{M}$ is graphic, cographic or R_{10}. In these cases we estimate $\widetilde{Z}(\mathcal{M}; \boldsymbol{\gamma})$ "directly", which means the following. If $\mathcal{M}$ is R_{10} then we evaluate $\widetilde{Z}(\mathcal{M}; \boldsymbol{\gamma})$ by brute force. If $\mathcal{M}$ is graphic, we form the weighted graph $(G, \boldsymbol{\gamma})$ whose (weighted) cycle matroid is $(\mathcal{M}, \boldsymbol{\gamma})$. Then the partition function of the Ising model on $(G, \boldsymbol{\gamma})$ may be estimated using the algorithm of Jerrum and Sinclair [8]. If $\mathcal{M}$ is cographic, then its dual $\mathcal{M}^*$ is graphic, and

$$\widetilde{Z}(\mathcal{M}; \boldsymbol{\gamma}) = \gamma_E q^{-r_{\mathcal{M}}(E)} \widetilde{Z}(\mathcal{M}^*; \boldsymbol{\gamma}^*),$$

where $E = E(\mathcal{M})$, and $\boldsymbol{\gamma}^*$ is the dual weighting given by $\gamma_e^* = q/\gamma_e = 2/\gamma_e$ for all $e \in E(\mathcal{M})$ [12, 4.14a]. (Ground set elements e with $\gamma_e = 0$ do not cause any problems, because they can just be deleted.) Then we proceed as before, but using $(\mathcal{M}^*, \boldsymbol{\gamma}^*)$ in place of $(\mathcal{M}, \boldsymbol{\gamma})$. The proposed algorithm is presented as Figure 1.

In the full version of the paper [5] there is a technical lemma that extends Corollary 1 to the situation where we have only an approximation to the signature of $\mathcal{M}_2$, and a corresponding lemma for the 2-sum. Given these results it is easy to see that the algorithm is correct. That is, given a regular matroid $\mathcal{M}$ and an accuracy parameter $\varepsilon < 1$, the algorithm returns an estimate $\hat{Z}$ satisfying $e^{-\varepsilon} \widetilde{Z}(\mathcal{M}, \boldsymbol{\gamma}) \leq \hat{Z} \leq e^{\varepsilon} \widetilde{Z}(\mathcal{M}, \boldsymbol{\gamma})$. The rest of this section shows that the running time is at most a polynomial in $|E(\mathcal{M})|$ and ε^{-1}.

Let $T_{\text{decomp}}(m) = O(m^{\alpha_{\text{decomp}}})$ be the time complexity of performing the Seymour decomposition of an m-element matroid (this is our initial splitting

Step 4a. Recusively estimate $z_0 = \widetilde{Z}(\mathcal{M}_2\backslash T;\boldsymbol{\gamma}_2)$, $z_1 = \widetilde{Z}(\mathcal{M}_2/p_1\backslash p_2,p_3;\boldsymbol{\gamma}_2)$, $z_2 = \widetilde{Z}(\mathcal{M}_2/p_2\backslash p_1,p_3;\boldsymbol{\gamma}_2)$ and $z_3 = \widetilde{Z}(\mathcal{M}_2/p_3\backslash p_1,p_2;\boldsymbol{\gamma}_2)$ with accuracy parameter $\varepsilon\varrho|\mathcal{M}_2|/(4|\mathcal{M}|)$.

Step 5a. Using techniques sketched in Section 4, compute d_1, d_2 and d_3 such that, for any weight function $\boldsymbol{\delta}$ with $\delta_{e_1} = d_1$, $\delta_{e_2} = d_2$ and $\delta_{e_3} = d_3$,

$$e^{-\varepsilon|\mathcal{M}_2|/(2|\mathcal{M}|)}\widetilde{Z}(\mathcal{M};\boldsymbol{\gamma}) \le \zeta\widetilde{Z}(\mathcal{M}_1 \triangle \mathcal{I}_3;\boldsymbol{\gamma}_1 \triangle \boldsymbol{\delta}) \le e^{\varepsilon|\mathcal{M}_2|/(2|\mathcal{M}|)}\widetilde{Z}(\mathcal{M};\boldsymbol{\gamma}).$$

Note that our estimate for z_0 gives an estimate for $\zeta = z_0/\sqrt{R/S_1S_2S_3}$ with accuracy parameter at most $\varepsilon|\mathcal{M}_2|/(2|\mathcal{M}|)$. ($R$, S_1, S_2 and S_3 are byproducts of the computation of d_1, d_2 and d_3.)

Step 6a. Recall from Section 4 that $\widetilde{Z}(\mathcal{M}_1 \triangle \mathcal{I}_3,\boldsymbol{\gamma}_1 \triangle \boldsymbol{\delta}) = \widetilde{Z}(\mathcal{M}_1,\boldsymbol{\gamma}')$, where $\boldsymbol{\gamma}'$ is derived from $\boldsymbol{\gamma}_1$ by assigning $\gamma'_{p_1} = \delta_{e_1}$, $\gamma'_{p_2} = \delta_{e_2}$ and $\gamma'_{p_3} = \delta_{e_3}$.

Step 7a. Recursively estimate $\widetilde{Z}(\mathcal{M}_1;\boldsymbol{\gamma}')$ with accuracy parameter $\varepsilon(|\mathcal{M}| - |\mathcal{M}_2|)/|\mathcal{M}|$ and multiply it by the estimate for ζ from Step 5a. Return this value, which is an estimate of $\widetilde{Z}(\mathcal{M},\boldsymbol{\gamma})$.

Fig. 2. The 3-sum case. (ϱ is a sufficiently small positive constant which does not depend upon $\mathcal{M}$ or ε.)

step), and let $T_{\text{base}}(m,\varepsilon) = O(m^{\alpha_{\text{base}}}\varepsilon^{-2})$ be the time complexity of estimating the Ising partition function of an m-edge graph. (From [8, Theorem 5], and the remark following it, we may take $\alpha_{\text{base}} = 15$.) Denote by $T(m,\varepsilon)$ the time-complexity of the algorithm of Figure 1. The recurrence governing $T(m,\varepsilon)$ is now presented, immediately followed by an explanation of its various components. Here, ϱ is a sufficiently small positive constant which does not depend upon $\mathcal{M}$ or ε, and can be taken to be $\varrho = 1/6000$.

$$\begin{aligned} T(m,\varepsilon) \le T_{\text{decomp}}(m) + \max\Big\{ & T_{\text{base}}(m,\varepsilon), \\ & \max_{4\le k\le m/2}\Big(T(m-k+3,\tfrac{\varepsilon(m-k-3)}{m}) + 4T(k,\tfrac{\varepsilon\varrho(k+3)}{4m})\Big), \\ & \max_{2\le k\le m/2}\Big(T(m-k+1,\tfrac{\varepsilon(m-k-1)}{m}) + 2T(k,\tfrac{\varepsilon\varrho(k+1)}{2m})\Big), \\ & \max_{1\le k\le m/2}\Big(T(m-k,\tfrac{\varepsilon(m-k)}{m}) + T(k,\tfrac{\varepsilon k}{m})\Big)\Big\}. \end{aligned}$$

The four expressions within the outer maximisation correspond to the direct case, the 3-sum case, the 2-sum case, and the 1-sum case, respectively. The variable k is to be interpreted as the number of ground set elements in $\mathcal{M}$ that come from $\mathcal{M}_2$ (and hence $m-k$ is the number that come from $\mathcal{M}_1$). Thus, in the case of a 3-sum, for example, $|E(\mathcal{M}_1)| = m-k+3$ and $|E(\mathcal{M}_2)| = k+3$. Note that Step 2 of the algorithm ensures $k \le m/2$. The lower bounds on k come from the corresponding lower bounds on the size of matroids occurring in 3-sums, 2-sums and 1-sums. In this extended abstract we omit the technical calculations necessary to justify the stated error bounds for the recursive calls.

It is a routine exercise demonstrate that $T(m, \varepsilon) = O(m^\alpha \varepsilon^{-2})$, where $\alpha = \max\{\alpha_{\text{base}}, \alpha_{\text{decomp}} + 1, 43\}$. Specifically, we show in the full paper, by induction on m, that $T(m, \varepsilon) \leq C m^\alpha \varepsilon^{-2}$, for some constant C and all sufficiently large m. In our analysis, we do not attempt to obtain the best possible exponent α for the running time. It would certainly be possible to reduce the constant 43 appearing in the formula for the exponent, but there seems little point in doing so, as the best existing value for α_{base} is already too large to make the algorithm feasible in practice.

References

1. Andrzejak, A.: Splitting formulas for Tutte polynomials. J. Combin. Theory Ser. B 70(2), 346–366 (1997)
2. Goldberg, L., Jerrum, M.: Approximating the partition function of the ferromagnetic potts model. In: Abramsky, S., Gavoille, C., Kirchner, C., Meyer auf der Heide, F., Spirakis, P.G. (eds.) ICALP 2010. LNCS, vol. 6198, pp. 396–407. Springer, Heidelberg (2010)
3. Goldberg, L.A., Jerrum, M.: Inapproximability of the Tutte polynomial. Inform. and Comput. 206(7), 908–929 (2008)
4. Goldberg, L.A., Jerrum, M.: Approximating the Tutte polynomial of a binary matroid and other related combinatorial polynomials. arXiv:1006.5234v1 (cs.CC) (2010)
5. Goldberg, L.A., Jerrum, M.: A polynomial-time algorithm for estimating the partition function of the ferromagnetic ising model on a regular matroid. arXiv:1010.6231v1 (cs.CC) (2010)
6. Golynski, A., Horton, J.D.: A polynomial time algorithm to find the minimum cycle basis of a regular matroid. In: Penttonen, M., Schmidt, E.M. (eds.) SWAT 2002. LNCS, vol. 2368, pp. 200–209. Springer, Heidelberg (2002)
7. Jaeger, F., Vertigan, D.L., Welsh, D.J.A.: On the computational complexity of the Jones and Tutte polynomials. Math. Proc. Cambridge Philos. Soc. 108(1), 35–53 (1990)
8. Jerrum, M., Sinclair, A.: Polynomial-time approximation algorithms for the Ising model. SIAM J. Comput. 22(5), 1087–1116 (1993)
9. Oxley, J.G.: Matroid theory. Oxford Science Publications. Oxford Science Publications, The Clarendon Press Oxford University Press, New York (1992)
10. Potts, R.B.: Some generalized order-disorder transformations. Proc. Cambridge Philos. Soc. 48, 106–109 (1952)
11. Seymour, P.D.: Decomposition of regular matroids. J. Combin. Theory Ser. B 28(3), 305–359 (1980)
12. Sokal, A.D.: The multivariate Tutte polynomial (alias Potts model) for graphs and matroids. In: Surveys in Combinatorics 2005. London Math. Soc. Lecture Note Ser., vol. 327, pp. 173–226. Cambridge Univ. Press, Cambridge (2005)
13. Truemper, K.: Matroid decomposition. Academic Press Inc., Boston (1992)
14. Truemper, K.: A decomposition theory for matroids. V. Testing of matrix total unimodularity. J. Combin. Theory Ser. B 49(2), 241–281 (1990)

Rapid Mixing of Subset Glauber Dynamics on Graphs of Bounded Tree-Width

Magnus Bordewich and Ross J. Kang

Durham University, Durham, UK
m.j.r.bordewich@durham.ac.uk, ross.kang@gmail.com

Abstract. Motivated by the 'subgraphs world' view of the ferromagnetic Ising model, we develop a general approach to studying mixing times of Glauber dynamics based on subset expansion expressions for a class of graph polynomials. With a canonical paths argument, we demonstrate that the chains defined within this framework mix rapidly upon graphs of bounded tree-width. This extends known results on rapid mixing for the Tutte polynomial, the adjacency-rank (R_2-)polynomial and the interlace polynomial.

Keywords: Markov chain Monte Carlo, graph polynomials, subset expansion, tree-width, canonical paths, randomised approximation schemes, rapid mixing.

1 Introduction

We analyse a subset-sampling Markov chain on simple graphs that is derived from certain graph functions — usually, in fact, graph polynomials. We show that this chain mixes rapidly on graphs of constant tree-width.

The graph functions $\mathcal{P}$ we consider are formulated using subset expansion. An *edge subset expansion formula for* $\mathcal{P}$ is written as follows: for any simple graph $G = (V, E)$,

$$\mathcal{P}(G) = \sum_{S \subseteq E} w((V, S)) \tag{1}$$

for some graph function w, where (V, S) denotes the graph with vertex set V and edge set S. If the function w is non-negative, that is, $w(G) \geq 0$ for all graphs G, we refer to (1) as an *edge subset weighting for* $\mathcal{P}$ and to w as its *weight function*. In fact, we shall need the weight function to be *positive* — from a statistical physics viewpoint, this results in a so-called 'soft-core model'.

Before moving on, let us anchor the general formula (1) with an example that is prominent in statistical physics, theoretical computer science, and discrete probability. The *partition function of the random cluster model* can be defined for any $G = (V, E)$ and parameters q, μ as

$$Z_{RC}(G; q, \mu) := \sum_{S \subseteq E} q^{\kappa(S)} \mu^{|S|}, \tag{2}$$

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 533–544, 2011.

where $\kappa(S)$ is the number of components in (V, S). For more on the random cluster model, see an extensive treatise by Grimmett [21]. Notice that, if $q, \mu \geq 0$, then $w((V, S)) := q^{\kappa(S)}\mu^{|S|}$ provides an edge subset weighting for $Z_{RC}(G; q, \mu)$. Under a suitable transformation, $Z_{RC}(G; q, \mu)$ is equivalent to the Tutte polynomial [43], defined for any $G = (V, E)$ and parameters x, y as

$$T(G; x, y) := \sum_{S \subseteq E} (x-1)^{r(E)-r(S)}(y-1)^{|S|-r(S)}, \tag{3}$$

where $r(S)$ is the $\mathbb{F}_2$-rank of the incidence matrix for (V, S). A wealth of combinatorial and structural information can be obtained from evaluations of this function. The Tutte polynomial specialises to several key univariate graph polynomials, including the chromatic polynomial of Birkhoff [5]. It specialises to the Jones polynomial in knot theory [28]. By its connection with the random cluster model, it also generalises the partition functions of the Ising [24] and Potts [38] models[1]. Consult the monograph of Welsh [44] for more on these crucial connections. In addition to $Z_{RC}(G; q, \mu)$ and $T(G; x, y)$, we shall highlight a few other specific polynomials from the literature, but for a broad account of the development of graph polynomials, consult the recent surveys by Makowsky [31] and Ellis-Monaghan and Merino [13,14].

It was shown in 1990 by Jaeger, Vertigan and Welsh [25] that, in general, for fixed (rational) values of x and y, the evaluation of $T(G; x, y)$ is #P-hard, except on a few special points and curves in the (x, y)-plane. As a result, there have been substantial efforts since then to pin down the *approximation complexity* of computing $T(G; x, y)$. For large swaths of the (x, y)-plane, it is now known that the computation of $T(G; x, y)$ either does not admit a fully polynomial-time randomised approximation scheme (FPRAS) unless RP = NP, or is at least as hard as #BIS (the problem of counting independent sets in bipartite graphs) under approximation-preserving reductions, cf. Goldberg and Jerrum [19]. The sole positive approximation result applicable to general graphs is the breakthrough FPRAS by Jerrum and Sinclair [27] for the partition function of the ferromagnetic Ising model — this corresponds to computation of $T(G; x, y)$ along the portion of the parabola $(x-1)(y-1) = 2$ with $y > 1$. Various approaches have given efficient approximations in some regions of the Tutte plane for specific classes of graphs — cf. e.g. [1,9,29]. To obtain their seminal result, Jerrum and Sinclair used a Markov chain Monte Carlo (MCMC) method, a principal tool in the design of efficient approximation schemes for counting problems. MCMC methods are widespread in computational physics, computational biology, machine learning, and statistics. There have been steady advances in our understanding of such random processes and in showing how quickly they generate good approximations of useful probability distributions in huge, complex data sets. See the lecture notes of Jerrum [26] or a survey by Randall [39] for an overview of the application of these techniques in theoretical computer science.

[1] If $x, y \geq 1$ or $q, \mu \geq 0$, then, respectively, $T(G; x, y)$ or $Z_{RC}(G; q, \mu)$ generalise the partition functions of the *ferromagnetic* Ising and Potts models.

We postpone the precise statement of our main result, Theorem 1, as it requires a host of definitions, but here we give a cursory description. In this paper, we are interested in the rate of convergence to stationarity of a natural Markov chain closely associated to a subset weighting of $\mathcal{P}$ (of form (1)), when some mild restriction is placed upon the weight function w. That restriction — which we have dubbed λ-*multiplicativity* — is described in Subsection 2.1: for now, we remark that some important graph polynomials and partition functions from statistical physics (e.g. $Z_{RC}(G;q,\mu)$ and $T(G;x,y)$) obey it. The state space of our chain is the set of all edge subsets, upon which we have set up a MCMC method using Glauber dynamics [17]. Each possible transition in the chain is either the addition or deletion of exactly one edge to/from the subset and the transition probabilities are defined according to the weights $w((V,S))$, subject to a Metropolis-Hastings filter [22,34][2]. Our main finding is that on graphs of bounded tree-width this Markov chain converges to the stationary distribution in time that is polynomial in the number of vertices of the graph.

Our approach is inspired in part by the 'subgraphs world' in which Jerrum and Sinclair designed their FPRAS for the partition function of the ferromagnetic Ising model. It is also motivated by recent work of Ge and Štefankovič [16], who introduced the R_2-polynomial in an attempt to devise a FPRAS for #BIS. Their *adjacency-rank polynomial* is defined for any $G=(V,E)$ and parameters q,μ as

$$R_2(G;q,\mu) := \sum_{S\subseteq E} q^{\mathrm{rk}_2(S)}\mu^{|S|}, \tag{4}$$

where $\mathrm{rk}_2(S)$ is the $\mathbb{F}_2$-rank of the *adjacency* matrix for (V,S). Using a combinatorial interpretation of rk_2 applicable only to bipartite graphs, they showed that the edge subset Glauber dynamics (using the weighting in (4)) mixes rapidly on trees. They conjectured that the chain mixes rapidly on all bipartite graphs, cf. Conjecture 1 in [16]. In addition, Ge and Štefankovič showed that the Markov chain for the (soft-core) random cluster model — i.e. weighted according to (2) — mixes rapidly upon graphs of bounded tree-width. We have extended both of these results under a unified framework. In particular, we show that the R_2-polynomial fits in our framework without recourse to the combinatorial interpretation for bipartite graphs, and hence that the Markov chain for the R_2-polynomial mixes rapidly upon all graphs of bounded tree-width. We also remark here that the conjectured rapid mixing of this chain on all bipartite graphs was disproved by Goldberg and Jerrum [18].

The polynomials and Markov chains that we capture in our framework are defined for any graph; however, we obtain rapid mixing results only on graphs of constant tree-width. For brevity, we will not define tree-width here, but merely say that it is an essential concept in structural graph theory and parameterised complexity — see modern surveys on the topic by Bodlaender [8] and Hliněný et al. [23]. The restriction of tree-width is commonly used in graph algorithms

[2] A Metropolis-Hastings filter is applied in order to ensure that the resulting process is a reversible Markov chain and thus guaranteed to converge to a unique stationary distribution with state probabilities proportional to the weights.

to reduce the complexity of a computationally difficult problem, usually by way of dynamic programming. For example, it is already known that many of the polynomials covered here can be evaluated efficiently for graphs of bounded tree-width. Independently, Andrzejak [2] and Noble [35] exhibited polynomial-time algorithms to compute the Tutte polynomial of graphs with bounded tree-width. Works of Makowsky and Mariño [32] and Noble [36] have significantly generalised this, in the former case, to a wide array of polynomials under the framework of monadic second order logic (MSOL), and, in the latter case, to the so-called U-polynomial [37], a polynomial that includes not only the Tutte polynomial but also a powerful type of knot invariant as a special case.

Even though many of the polynomials we refer to can be computed exactly in polynomial time for graphs of bounded tree-width, it remains of interest to show that the associated Glauber dynamics is rapidly mixing. One hope is that for some polynomials the chain mixes rapidly for a wider class of graphs. There have been significant and concerted endeavours by researchers spanning physics, computer science and probability to determine the mixing properties of Glauber dynamics on many related Markov chains. Spin systems have been of particular interest; indeed, the main thrust of the work of Jerrum and Sinclair was to tackle the partition function for the 'spins world' of the ferromagnetic Ising model (using a translation to the rapidly mixing 'subgraphs world'). Much recent work on spin systems is restricted to trees or tree-like graphs, cf. e.g. [4,11,12,20,33,41].

Our primary focus in this paper is to establish results for polynomials defined according to *edge* subset expansion, but we can also extend our methodology to polynomials defined according to *vertex* subset expansion, which may be viewed as the 'induced subgraphs world'. To our knowledge, this form of Markov chain has not been greatly examined, but it handles one important graph polynomial that was recently introduced by Arratia, Bollobás and Sorkin [3]: the *bivariate interlace polynomial* is defined for any graph $G = (V, E)$ and parameters x, y as

$$q(G; x, y) := \sum_{S \subseteq V} (x-1)^{\mathrm{rk}_2(S)} (y-1)^{|V| - \mathrm{rk}_2(S)}, \tag{5}$$

where $\mathrm{rk}_2(S)$ is the $\mathbb{F}_2$-rank of the adjacency matrix for $G[S]$. This polynomial specialises to the independence polynomial and is intimately related to Martin polynomials. Just as for the Tutte polynomial, computation of the bivariate interlace polynomial is #P-hard in almost the entire plane [7]. The multivariate interlace polynomial, a generalisation of the interlace polynomial, can be evaluated efficiently for graphs of bounded tree-width [10], cf. [6]. Subject to a condition on the weightings, which we call *vertex λ-multiplicativity*, we establish rapid mixing for vertex subset Glauber dynamics on graphs of constant tree-width.

For all of our results, we need that the weight function is strictly positive for all (induced) subgraphs. Many of the classical enumeration polynomials such as the matching, independence, clique and chromatic polynomials are captured by the general polynomials that we mention as examples throughout this work. However, these are 'hard-core models', in which some (induced) subgraphs have

a zero weighting, and hence are not included in our approach. Many of these are evaluations that fall at the boundary of the regions that we can handle. For example, the Tutte polynomial evaluated at the point $(2, 1)$ counts the number of forests of the graph. We have shown rapid mixing at all fixed points $(2, 1+\delta)$, for $\delta > 0$, with a mixing time that depends on δ. It would be interesting to consider whether the chains associated with these boundary points mix rapidly for graphs of bounded tree-width.

Outline of the paper. In the next section, we give the definitions that are necessary for a detailed description of the main theorem. We give the main theorem in Section 3 and then indicate some of its consequences. We present an outline of the proof in Section 4. In Section 5, we state how our results extend to Glauber dynamics on vertex subsets.

2 Definitions

2.1 λ-Multiplicative Weight Functions

In this subsection, we describe the condition we require on our graph functions $\mathcal{P}$. This condition prescribes that the weight function is multiplicative with respect to the operation of disjoint graph union as well as "nearly multiplicative" with respect to the operation of composition via small vertex cuts.

We use the notation $\hat{\lambda} := \max\{\lambda, 1/\lambda\}$. For a graph $G = (V, E)$, a vertex cut K is said to separate sets V_1 and V_2 if (V_1, K, V_2) is a partition of V and there is no edge of E that is incident to both a vertex of V_1 and a vertex of V_2. A partition (E_1, E_2) of E is appropriate (for K) if E_1 has no edge adjacent to a vertex in V_2 and E_2 has no edge adjacent to a vertex in V_1.

For fixed $\lambda > 0$, we say that the weight function w is λ-*multiplicative*, if for any $G = (V, E)$, any vertex cut K that separates sets V_1 and V_2, and any appropriate partition (E_1, E_2), we have

$$\hat{\lambda}^{-|K|} \leq \frac{w((V_1 \cup K, E_1))w((V_2 \cup K, E_2))}{w(G)} \leq \hat{\lambda}^{|K|}. \tag{6}$$

As mentioned above, if w is λ-multiplicative, then w is multiplicative with respect to disjoint union (by taking $K = \emptyset$); furthermore, taking $V_2 = \emptyset$ implies that the addition or deletion of a few edges in the graph does not change w wildly.

2.2 Examples of Valid Polynomials

In this subsection, we emphasise specific examples with weight functions that are λ-multiplicative. Let $G = (V, E)$ be any graph, K be any vertex cut that separates vertex subsets V_1 and V_2, and (E_1, E_2) be any appropriate partition. We define G' to be the disjoint union of graphs $(V_1 \cup K, E_1)$ and $(V_2 \cup K, E_2)$. We could imagine forming G' from G by splitting each vertex in K, taking incident edges in E_1 with one copy of the vertex and those in E_2 with the other. It is trivial

to verify multiplicativity with respect to disjoint union for each of the weight functions considered below. Therefore, to establish λ-multiplicativity for these weight functions w, it will suffice to verify that $\hat{\lambda}^{-|K|} \leq w(G')/w(G) \leq \hat{\lambda}^{|K|}$.

First, we observe that the partition function of the *random cluster model* for $q, \mu > 0$ satisfies the condition. Recalling (2), the relevant weight function is $w((V,S)) := q^{\kappa(S)}\mu^{|S|}$. To handle the $\mu^{|S|}$ factor, note that the graphs G and G' have the same number of edges. For the $q^{\kappa(S)}$ factor, the number of components in G' can be at most $\kappa(G)+|K|$ since G' can be obtained by splitting $|K|$ vertices of G. Thus, w is λ-multiplicative if we take $\lambda := q$.

This can also be seen in the context of the *Tutte polynomial* when $x, y > 1$. Recalling (3), the relevant weight function is $w((V,S)) := (x-1)^{r(E)-r(S)}(y-1)^{|S|-r(S)}$. As before, it is easy to take care of the $(x-1)^{r(E)}(y-1)^{|S|}$ factor. For the remaining $((x-1)(y-1))^{-r(S)}$ factor, it is enough to observe that the incidence matrix of G may be obtained from the incidence matrix of G' as follows. The matrix for G' has two rows for each of the vertices in K, one from $(V_1 \cup K, E_1)$ and one from $(V_2 \cup K, E_2)$. If we replace one of these two rows with the sum of the two rows, we do not alter the rank; if we then delete the other of the two rows, we change the rank by at most 1. Repeating this for each vertex in K, we obtain the incidence matrix for G, at a total change in the rank r of the incidence matrix of at most $|K|$. Thus, w is λ-multiplicative if we take $\lambda := (x-1)(y-1)$.

Next, we see that the *adjacency-rank polynomial* of Ge and Štefankovič satisfies the condition if $q, \mu > 0$. Recalling (4), the relevant weight function is $w((V,S)) := q^{\mathrm{rk}_2(S)}\mu^{|S|}$. As before, it is simple to handle the $\mu^{|S|}$ factor. For the $q^{\mathrm{rk}_2(S)}$ factor, we note that the adjacency matrix of G may be formed from the adjacency matrix of G' by $|K|$ row additions, followed by $|K|$ column additions and finally the deletion of $|K|$ rows and $|K|$ columns. Since we must delete both rows and columns, the rank rk_2 of the adjacency matrix may change by up to $2|K|$. Thus, in this case, w is λ-multiplicative when taking $\lambda := q^2$.

Now, consider the *multivariate Tutte polynomial* as formulated by Sokal [40], defined for any graph $G = (V,E)$ and parameters $q, \boldsymbol{v} = \{v_e\}_{e\in E}$ by

$$Z_{Tutte}(G; q, \boldsymbol{v}) := \sum_{S\subseteq E} q^{\kappa(S)} \prod_{e\in S} v_e. \tag{7}$$

Under this expansion, $w := q^{\kappa(S)}\prod_{e\in S} v_e$ is an edge subset weight function if $q > 0$ and $v_e > 0$ for any $e \in E$ are fixed. We can handle the $q^{\kappa(S)}$ factor as we did for the random cluster model partition function. For the $\prod_{e\in S} v_e$ factor, observe that G and G' have the same set of edges. Thus, w is λ-multiplicative when taking $\lambda := q$.

Last, we discuss the *U-polynomial* of Noble and Welsh [37], defined for any graph $G = (V,E)$ and parameters $y, \boldsymbol{x} = \{x_i\}_{i=1}^{|V|}$ by

$$U(G; \boldsymbol{x}, y) := \sum_{S\subseteq E} (y-1)^{|S|-r(S)} \prod_{i=1}^{|V|} {x_i}^{\kappa(i,S)}, \tag{8}$$

where $\kappa(i,S)$ denotes the number of components of order i in (V,S). If $y > 1$ and $x_i > 0$ for all i, then $w((V,S)) := (y-1)^{|S|-r(S)} \prod_{i=1}^{|V|} {x_i}^{\kappa(i,S)}$ gives an edge subset weighting. The $(y-1)^{|S|-r(S)}$ factor can be handled as above. For the $\prod_{i=1}^{|V|} {x_i}^{\kappa(i,S)}$ factor, observe that $\sum_i |\kappa(i,G) - \kappa(i,G')|$ is at most $3|K|$, since, if we obtain G' by splitting the vertices in K, each time we split a vertex we either change the size of a single component or split a single component into two smaller components. Thus, taking $x' := \max_i \max\{x_i, x_i^{-1}\}$ and $y' := \max\{y-1, (y-1)^{-1}\}$, we see that w is λ-multiplicative when taking $\lambda := y' {x'}^3$.

2.3 Glauber Dynamics for Edge Subsets

In this subsection, we define the Markov chain associated with the edge subset expansion formula for $\mathcal{P}$. From the formulation in (1), the *single bond flip chain* $\mathcal{M}$ on a given graph $G = (V,E)$ is defined as follows. We start with an arbitrary subset $X_0 \subseteq E$ and repeatedly generate X_{t+1} from X_t by running the following experiment.

1. Pick an edge $e \in E$ uniformly at random and let $S = X_t \oplus \{e\}$.
2. Set $X_{t+1} = S$ with probability $\frac{1}{2}\min\{1, w((V,S))/w((V,X_t))\}$ and $X_{t+1} = X_t$ with the remaining probability.

By convention, we denote the state space of $\mathcal{M}$ by Ω (i.e. $\Omega = 2^E$) and its transition probability matrix by P.

The term *rapidly mixing* applies to a Markov chain that quickly converges to its stationary distribution. We make this precise here. The *total variation distance* $\|\nu - \nu'\|_{TV}$ between two probability distributions ν and ν' is defined by $\|\nu - \nu'\|_{TV} = \frac{1}{2}\sum_{H\in\Omega} |\nu(H) - \nu'(H)|$. For $\varepsilon > 0$, the *mixing time* of a Markov chain $\mathcal{M}$ (with state space Ω, transition matrix P and stationary distribution π) is defined as

$$\tau(\varepsilon) := \max_{H\in\Omega}\{\min\{t \mid \|P^t(H,\cdot) - \pi(\cdot)\|_{TV} \le \varepsilon\}\}.$$

In this paper, we shall say that a chain $\mathcal{M}$ *mixes rapidly* if, for any fixed ε, $\tau(\varepsilon)$ is (upper) bounded by a polynomial in the number of vertices of the input graph.

3 Results

We are now prepared for a precise statement of the main theorem.

Theorem 1. *Let $G = (V,E)$ where $|V| = n$. If w is λ-multiplicative for some $\lambda > 0$, then the mixing time of $\mathcal{M}$ on G satisfies*

$$\tau(\varepsilon) = O\left(n^{4+4(\mathrm{tw}(G)+1)|\log\lambda|}\log(1/\varepsilon)\right)$$

(where $\mathrm{tw}(G)$ denotes the tree-width of G).

In Subsection 2.2, we noted some examples of polynomials with λ-multiplicative weight functions; thus, Theorem 1 implies the following.

Corollary 1. *Let $G = (V, E)$ where $|V| = n$. In the following list, we state conditions on the parameters which guarantee rapid mixing of the single bond flip chain on G associated with the stated polynomial and weighting. We also state the mixing time bound.*

1. *For fixed $q, \mu > 0$ and the weighting (2) of $Z_{RC}(G; q, \mu)$, the mixing time satisfies*

$$\tau(\varepsilon) = O\left(n^{4+4(\mathrm{tw}(G)+1)|\log q|} \log(1/\varepsilon)\right).$$

   Equivalently, for fixed $x, y > 1$ and the weighting (3) of $T(G; x, y)$, the mixing time satisfies

$$\tau(\varepsilon) = O\left(n^{4+4(\mathrm{tw}(G)+1)|\log((x-1)(y-1))|} \log(1/\varepsilon)\right).$$

2. *For fixed $q, \mu > 0$ and the weighting (4) of $R_2(G; q, \mu)$, the mixing time satisfies*

$$\tau(\varepsilon) = O\left(n^{4+8(\mathrm{tw}(G)+1)|\log q|} \log(1/\varepsilon)\right).$$

3. *For fixed $q > 0$ and $v_e > 0$ for all e and the weighting (7) of $Z(G; q, \boldsymbol{v})$, the mixing time satisfies*

$$\tau(\varepsilon) = O\left(n^{4+4(\mathrm{tw}(G)+1)|\log q|} \log(1/\varepsilon)\right).$$

4. *For fixed $y > 1$ and $x_i > 0$ for all i and the weighting (8) of $U(G; \boldsymbol{x}, \mu)$, the mixing time satisfies*

$$\tau(\varepsilon) = O\left(n^{4+4(\mathrm{tw}(G)+1)\left|\log\left(y' x'^3\right)\right|} \log(1/\varepsilon)\right),$$

   where $x' = \max_i \max\{x_i, x_i^{-1}\}$ and $y' = \max\{y - 1, (y-1)^{-1}\}$.

Here, we remark that Ge and Štefankovič obtained part 1 above and showed part 2 above in the special case of trees. Parts 2–4 directly extend these findings, and our main theorem considerably broadens the scope of mixing time bounds for subset Glauber dynamics on graphs of bounded tree-width.

4 Proof Outline

Due to page restrictions, the detailed proof of Theorem 1 has been postponed to a full journal version and can be found on arXiv, but we give a brief outline.

Although our main result is stated in terms of tree-width, we do not treat tree-width directly but instead use linear-width, a more restrictive width parameter

introduced by Thomas [42], which is nearly equal to path-width pw [15]. This strategy was also employed by Ge and Štefankovič in the two specific cases mentioned above. For any graph $G = (V, E)$, an ordering $(e_1, \ldots, e_m)$ of E has linear-width at most ℓ, if, for each $i \in \{1, \ldots, m\}$, there are at most ℓ vertices that are incident to both an edge in $\{e_1, \ldots, e_{i-1}\}$ and an edge in $\{e_i, \ldots, e_m\}$. The *linear-width* $\mathrm{lw}(G)$ of $G = (V, E)$ is the smallest integer ℓ such that there is an ordering of E with linear-width at most ℓ. The motive for using linear-width is that it implies an ordering of the edges which we can then use to define canonical paths between pairs of edge subsets. Then we show that λ-multiplicativity is the general condition under which we can bound the congestion of these canonical paths. A key relationship we rely on is that $\mathrm{pw}(G) \leq (\mathrm{tw}(G) + 1)(\lfloor \log_2 n \rfloor + 1)$, cf. [16]. The use of canonical paths is a standard technique for obtaining a bound on the mixing time of MCMC methods — see the lecture notes of Jerrum [26] for an expository account of this approach.

5 Vertex Subset Mixing for Bounded Tree-Width

Until now, we have considered edge subsets (subgraphs) and Glauber transitions which change one edge at a time. In this section, we consider vertex subsets (induced subgraphs) and transitions that involve one vertex at a time — each such transition can affect many edges, up to the maximum degree of G.

A *vertex subset expansion formula for* $\mathcal{P}$ is written as follows: for any simple graph $G = (V, E)$,

$$\mathcal{P}(G) = \sum_{S \subseteq V} w(G[S]) \tag{9}$$

for some graph function w, where $G[S]$ denotes the subgraph of G induced by S. If the function w is non-negative, we refer to (9) as a *vertex subset weighting for* $\mathcal{P}$ and to w as its *weight function*. From such a weighting, we define the *single site flip chain* $\mathcal{M}'$ on a given graph $G = (V, E)$ as follows. We start with an arbitrary subset $X_0 \subseteq V$ and repeatedly generate X_{t+1} from X_t by running the following experiment.

1. Pick a vertex $v \in V$ uniformly at random and let $S = X_t \oplus \{v\}$.
2. Set $X_{t+1} = S$ with probability $\frac{1}{2} \min\{1, w(G[S])/w(G[X_t])\}$ and $X_{t+1} = X_t$ with the remaining probability.

For fixed $\lambda > 0$, we say that the weight function w in (9) is *vertex* λ*-multiplicative*, if for any $G = (V, E)$ and K a vertex cut that separates sets V_1 and V_2 with respect to G, we have

$$\hat{\lambda}^{-|K|} \leq \frac{w(G[V_1])w(G[V_2 \cup K])}{w(G)} \leq \hat{\lambda}^{|K|}. \tag{10}$$

The main result of this section is the following.

Theorem 2. *Let $G = (V, E)$ where $|V| = n$. If w is vertex λ-multiplicative for some $\lambda > 0$, then the mixing time of $\mathcal{M}'$ on G satisfies*

$$\tau(\varepsilon) = O\left(n^{2+4(\mathrm{tw}(G)+1)|\log\lambda|}\log(1/\varepsilon)\right).$$

Again, due to space limitations, we have omitted the proof, but note that it follows a pattern similar to what is described in Section 4, with the exception that instead of linear-width it is convenient to work with *vertex-separation* (a closely related width parameter, shown by Kinnersley [30] to be equal to path-width).

Recalling (5), for fixed $x, y > 1$, $w(G[S]) := (x-1)^{\mathrm{rk}_2(S)}(y-1)^{|V|-\mathrm{rk}_2(S)}$ gives a vertex subset weighting for $q(G; x, y)$. With arguments similar to those given in Subsection 2.2, it can be verified that this weight function is vertex λ-multiplicative. By Theorem 2, it follows that a natural Markov chain derived from the bivariate interlace polynomial — a chain that has not been studied extensively, as far as we are aware — mixes rapidly on tree-width-bounded graphs.

Corollary 2. *Let $G = (V, E)$ where $|V| = n$. If $x, y > 1$ are fixed, then for the single site flip chain on G associated with the weighting (5) of $q(G; q, \mu)$, the mixing time satisfies*

$$\tau(\varepsilon) = O\left(n^{2+8(\mathrm{tw}(G)+1)|\log((x-1)/(y-1))|}\log(1/\varepsilon)\right).$$

6 Conclusion

In this work, we have developed a new general framework of graph polynomials and Markov chains defined via subset expansion formulae for these polynomials, and demonstrated that their dynamics mix rapidly for graphs of bounded tree-width. On a graph G with n vertices, we have shown a mixing time of order $n^{O(1)}e^{O(\mathrm{pw}(G))} = n^{O(\mathrm{tw}(G))}$. Our results apply to many of the most prominent and well-known polynomials in the field. The mixing times of our processes have, respectively, exponential and super-exponential dependencies upon path-width and tree-width. We ask if this could be improved, in particular, to achieve something akin to fixed-parameter tractability in terms of tree-width.

Acknowledgements. This research was supported by the Engineering and Physical Sciences Research Council (EPSRC), grant EP/G066604/1.

References

1. Alon, N., Frieze, A., Welsh, D.: Polynomial time randomized approximation schemes for Tutte-Gröthendieck invariants: the dense case. Random Structures Algorithms 6(4), 459–478 (1995)
2. Andrzejak, A.: An algorithm for the Tutte polynomials of graphs of bounded treewidth. Discrete Math. 190(1-3), 39–54 (1998)

3. Arratia, R., Bollobás, B., Sorkin, G.B.: A two-variable interlace polynomial. Combinatorica 24(4), 567–584 (2004)
4. Berger, N., Kenyon, C., Mossel, E., Peres, Y.: Glauber dynamics on trees and hyperbolic graphs. Probab. Theory Related Fields 131(3), 311–340 (2005)
5. Birkhoff, G.D.: A determinant formula for the number of ways of coloring a map. Ann. of Math. (2) 14(1-4), 42–46 (1912/1913)
6. Bläser, M., Hoffmann, C.: Fast evaluation of interlace polynomials on graphs of bounded treewidth. To appear in Algorithmica, doi:10.1007/s00453-010-9439-4
7. Bläser, M., Hoffmann, C.: On the complexity of the interlace polynomial. In: Albers, S., Weil, P. (eds.) 25th International Symposium on Theoretical Aspects of Computer Science (STACS 2008), Dagstuhl, Germany. Leibniz International Proceedings in Informatics (LIPIcs), vol. 1, pp. 97–108. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik (2008)
8. Bodlaender, H.L.: Treewidth: Characterizations, applications, and computations. In: Fomin, F.V. (ed.) WG 2006. LNCS, vol. 4271, pp. 1–14. Springer, Heidelberg (2006)
9. Bordewich, M.: Approximating the number of acyclic orientations for a class of sparse graphs. Combin. Probab. Comput. 13(1), 1–16 (2004)
10. Courcelle, B.: A multivariate interlace polynomial and its computation for graphs of bounded clique-width. Electron. J. Combin. 15(1): Research Paper 69, 36 (2008)
11. Dembo, A., Montanari, A.: Ising models on locally tree-like graphs. Ann. Appl. Probab. 20(2), 565–592 (2010)
12. Ding, J., Lubetzky, E., Peres, Y.: Mixing time of critical Ising model on trees is polynomial in the height. Comm. Math. Phys. 295(1), 161–207 (2010)
13. Ellis-Monaghan, J.A., Merino, C.: Graph polynomials and their applications I: The Tutte polynomial. In: Dehmer, M. (ed.) Structural Analysis of Complex Networks, pp. 219–255. Birkhäuser, Boston (2011)
14. Ellis-Monaghan, J.A., Merino, C.: Graph polynomials and their applications II: Interrelations and interpretations. In: Dehmer, M. (ed.) Structural Analysis of Complex Networks, pp. 257–292. Birkhäuser, Boston (2011)
15. Fomin, F.V., Thilikos, D.M.: A 3-approximation for the pathwidth of Halin graphs. J. Discrete Algorithms 4(4), 499–510 (2006)
16. Ge, Q., Štefankovič, D.: A graph polynomial for independent sets of bipartite graphs. CoRR, abs/0911.4732 (2009)
17. Glauber, R.J.: Time-dependent statistics of the Ising model. J. Mathematical Phys. 4, 294–307 (1963)
18. Goldberg, L.A., Jerrum, M.: Personal communication (2010)
19. Goldberg, L.A., Jerrum, M.: Approximating the partition function of the ferromagnetic potts model. In: Abramsky, S., Gavoille, C., Kirchner, C., Meyer auf der Heide, F., Spirakis, P.G. (eds.) ICALP 2010. LNCS, vol. 6198, pp. 396–407. Springer, Heidelberg (2010)
20. Goldberg, L.A., Jerrum, M., Karpinski, M.: The mixing time of Glauber dynamics for coloring regular trees. Random Structures Algorithms 36(4), 464–476 (2010)
21. Grimmett, G.: The random-cluster model. Grundlehren der Mathematischen Wissenschaften (Fundamental Principles of Mathematical Sciences), vol. 333. Springer, Berlin (2006)
22. Hastings, W.K.: Monte Carlo sampling methods using Markov chains and their applications. Biometrika 57(1), 97–109 (1970)
23. Hliněný, P., Oum, S.-i., Seese, D., Gottlob, G.: Width Parameters Beyond Treewidth and their Applications. The Computer Journal 51(3), 326–362 (2008)

24. Ising, E.: Beitrag zur theorie des ferromagnetismus. Zeitschrift für Physik A Hadrons and Nuclei 31, 253–258 (1925), doi:10.1007/BF02980577
25. Jaeger, F., Vertigan, D.L., Welsh, D.J.A.: On the computational complexity of the Jones and Tutte polynomials. Math. Proc. Cambridge Philos. Soc. 108(1), 35–53 (1990)
26. Jerrum, M.: Counting, sampling and integrating: algorithms and complexity, ETH Zürich. Lectures in Mathematics. Birkhäuser, Basel (2003)
27. Jerrum, M., Sinclair, A.: Polynomial-time approximation algorithms for the Ising model. SIAM J. Comput. 22(5), 1087–1116 (1993)
28. Jones, V.F.R.: A polynomial invariant for knots via von Neumann algebras. Bull. Amer. Math. Soc (N.S.) 12(1), 103–111 (1985)
29. Karger, D.R.: A randomized fully polynomial time approximation scheme for the all-terminal network reliability problem. SIAM Rev. 43(3), 499–522 (2001)
30. Kinnersley, N.G.: The vertex separation number of a graph equals its path-width. Inform. Process. Lett. 42(6), 345–350 (1992)
31. Makowsky, J.A.: From a zoo to a zoology: towards a general theory of graph polynomials. Theory Comput. Syst. 43(3-4), 542–562 (2008)
32. Makowsky, J.A., Mariño, J.P.: Farrell polynomials on graphs of bounded tree width. Adv. in Appl. Math. 30(1-2), 160–176 (2003), Formal power series and algebraic combinatorics, Scottsdale, AZ (2001)
33. Martinelli, F., Sinclair, A., Weitz, D.: Fast mixing for independent sets, colorings, and other models on trees. Random Structures Algorithms 31(2), 134–172 (2007)
34. Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., Teller, E.: Equation of state calculations by fast computing machines. The Journal of Chemical Physics 21(6), 1087–1092 (1953)
35. Noble, S.D.: Evaluating the Tutte polynomial for graphs of bounded tree-width. Combin. Probab. Comput. 7(3), 307–321 (1998)
36. Noble, S.D.: Evaluating a weighted graph polynomial for graphs of bounded tree-width. Electron. J. Combin. 16(1): research Paper 64, 14 (2009)
37. Noble, S.D., Welsh, D.J.A.: A weighted graph polynomial from chromatic invariants of knots. Ann. Inst. Fourier 49(3), 1057–1087 (1999), Symposium à la Mémoire de François Jaeger, Grenoble (1998)
38. Potts, R.B.: Some generalized order-disorder transformations. Proc. Cambridge Philos. Soc. 48, 106–109 (1952)
39. Randall, D.: Rapidly mixing Markov chains with applications in computer science and physics. Computing in Science Engineering 8(2), 30–41 (2006)
40. Sokal, A.D.: The multivariate Tutte polynomial (alias Potts model) for graphs and matroids. In: Surveys in Combinatorics 2005. London Math. Soc. Lecture Note Ser., vol. 327, pp. 173–226. Cambridge Univ. Press, Cambridge (2005)
41. Tetali, P., Vera, J.C., Vigoda, E., Yang, L.: Phase transition for the mixing time of the Glauber dynamics for coloring regular trees. In: Charikar, M. (ed.) SODA, pp. 1646–1656. SIAM, Philadelphia (2010)
42. Thomas, R.: Tree-decompositions of graphs (1996), Lecture notes http://www.math.gatech.edu/~thomas/tree.ps
43. Tutte, W.T.: A contribution to the theory of chromatic polynomials. Canadian J. Math. 6, 80–91 (1954)
44. Welsh, D.J.A.: Complexity: knots, colourings and counting. London Mathematical Society Lecture Note Series, vol. 186. Cambridge University Press, Cambridge (1993)

Efficient Sample Extractors for Juntas with Applications*

Sourav Chakraborty[1], David García-Soriano[2], and Arie Matsliah[3]

[1] Chennai Mathematical Institute, India
[2] CWI Amsterdam, The Netherlands
[3] IBM Research and Technion, Haifa, Israel

Abstract. We develop a query-efficient sample extractor for juntas, that is, a probabilistic algorithm that can simulate random samples from the core of a k-junta $f : \{0,1\}^n \to \{0,1\}$ given oracle access to a function $f' : \{0,1\}^n \to \{0,1\}$ that is only close to f. After a preprocessing step, which takes $\widetilde{O}(k)$ queries, generating each sample to the core of f takes only one query to f'.

We then plug in our sample extractor in the "testing by implicit learning" framework of Diakonikolas et al. [DLM+07], improving the query complexity of testers for various Boolean function classes. In particular, for some of the classes considered in [DLM+07], such as s-term DNF formulas, size-s decision trees, size-s Boolean formulas, s-sparse polynomials over $\mathbb{F}_2$, and size-s branching programs, the query complexity is reduced from $\widetilde{O}(s^4/\epsilon^2)$ to $\widetilde{O}(s/\epsilon^2)$. This shows that using the new sample extractor, testing by implicit learning can lead to testers having better query complexity than those tailored to a specific problem, such as the tester of Parnas et al. [PRS02] for the class of monotone s-term DNF formulas.

In terms of techniques, we extend the tools used in [CGM11] for testing function isomorphism to juntas. Specifically, while the original analysis in [CGM11] allowed query-efficient noisy sampling from the core of any k-junta f, the one presented here allows similar sampling from the core of the *closest* k-junta to f, even if f is not a k-junta but just close to being one. One of the observations leading to this extension is that the junta tester of Blais [Bla09], based on which the aforementioned sampling is achieved, enjoys a certain weak form of tolerance.

Keywords: property testing, sample extractors, implicit learning.

1 Introduction

Suppose we wish to test for the property defined by a class $\mathcal{C}$ of Boolean functions over $\{0,1\}^n$; that is, we aim to distinguish the case $f \in \mathcal{C}$ from the case $\mathrm{dist}(f,\mathcal{C}) \geq \epsilon$. The class is parameterized by a "size" parameter s (e.g. the class of DNFs with s terms, or circuits of size s) and, as usual, our goal is to minimize

* Research supported in part by an ERC-2007-StG grant number 202405.

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 545–556, 2011.

the number of queries made to f. In particular we strive for query complexity independent of n whenever possible.

The main observation underlying the "testing by implicit learning" paradigm of Diakonikolas et al. [DLM+07] (see also [Ser10], [DLM+08], [GOS+09]) is that a large number of interesting classes $\mathcal{C}$ can be well approximated by (relatively) small juntas also belonging to $\mathcal{C}$.

The prototypical example is obtained by taking for $\mathcal{C}$ the class of s-term DNFs. Let $\tau > 0$ be an approximation parameter (which for our purpose should be thought of as polynomial in ϵ/s). Any DNF term involving more than $\log(s/\tau)$ variables may be removed from f while affecting only a τ/s fraction of its values; hence, removing all of them results in an s-term DNF f' that is τ-close to f and *depends on only* $s\log(s/\tau)$ *variables* (equivalently, f' is a $s\log(s/\tau)$-junta). Let $\mathsf{Jun}_{[k]}$ denote the subset of (k-junta) functions $\{0,1\}^n \to \{0,1\}$ that depend only on the *first* k variables. Since the class $\mathcal{C}$ is isomorphism-invariant (closed under permutations of the variables), the foregoing observation can be rephrased as follows: for any $k \geq s\log(s/\tau)$, the subclass $\mathcal{C}_{[k]} \triangleq \mathcal{C} \cap \mathsf{Jun}_{[k]}$ is such that every $f \in \mathcal{C}$ is τ-close to being isomorphic to some $g \in \mathcal{C}_{[k]}$ (in short, $\mathrm{distiso}(f, \mathcal{C}_{[k]}) \leq \tau$).

On the other hand, for every f such that $\mathrm{dist}(f,\mathcal{C}) = \mathrm{distiso}(f,\mathcal{C}) \geq \epsilon$ it also holds that $\mathrm{distiso}(f, \mathcal{C}_{[k]}) \geq \epsilon$, since $\mathcal{C}_{[k]} \subseteq \mathcal{C}$. Hence, to solve the original problem, all we need is to differentiate between the two cases (i) $\mathrm{distiso}(f, \mathcal{C}_{[k]}) \leq \tau$ and (ii) $\mathrm{distiso}(f, \mathcal{C}_{[k]}) \geq \epsilon$.

Let us denote by f^* the k-junta that is closest to f; f^* can be identified with its *core*, i.e. the Boolean function $\mathsf{core}_k(f^*) : \{0,1\}^k \to \{0,1\}$ obtained from f^* by dropping its irrelevant variables. If we could somehow manage to get random samples of the form $(x, \mathsf{core}_k(f^*)(x)) \in \{0,1\}^k \times \{0,1\}$, we could use standard learning algorithms to identify an element $g \in \mathcal{C}_{[k]}$ which is close to being isomorphic to f^* (if any), which would essentially allow us to differentiate between the aforementioned cases. The number of such samples required for this is roughly logarithmic in $|\mathcal{C}_{[k]}|$; we elaborate on this later.[1] An important observation is that the size of $\mathcal{C}_{[k]} \triangleq \mathcal{C} \cap \mathsf{Jun}_{[k]}$ is usually very small, even compared to the size of $\mathsf{Jun}_{[k]}$, which is 2^{2^k}. For instance, it is not hard to see that for the case of s-term DNFs, the size of $\mathcal{C}_{[k]}$ is bounded by $(2k)^k$, which is exponential in k, rather than doubly exponential.

It is a surprising fact that such samples from the core of f^* can indeed be efficiently obtained (with some noise), even though f is the only function we have access to. Even having query access to f^* itself would not seem to help much at first glance, since the location of the relevant variables of f^* is unknown to us, and cannot be found without introducing a dependence of n in the query complexity. It is in this step that our approach departs from that of [DLM+07]. We mention next the two main differences that, when combined together, lead to better query complexity bounds.

[1] Issues of computational efficiency are usually disregarded here; however see [DLM+08].

The first difference is in the junta-testing part; both algorithms start with a junta tester to identify k disjoint subsets of variables (blocks), such that every "influential" variable of the function f being tested lies in one of these blocks. While [DLM+07] use the tolerant version of the junta-tester of Fischer et al. [FKR+02], we switch to the query-efficient junta tester of Blais [Bla09]. To make this step possible, we have to show that the tester from [Bla09] is sufficiently tolerant (the level of tolerance of the tester determines how large τ can be, which in turn determines how small k can be). The second (and the main) difference is in sample extraction - the actual process that obtains samples from the core of f^*. While in [DLM+07] sampling is achieved via independence tests[2], applied to each of the identified blocks separately (which requires $\Omega(k)$ queries to f per sample), we use ideas from [CGM11] instead. The algorithm presented in [CGM11, Section 7] accomplishes this task in the (strict) case $f = f^*$ by making just *one* query to f. The bulk of this work is a proof that, when f is close enough to f^*, it is still possible to obtain each such sample using only one query to f (an overview of the proof is given in Section 4.1).

Organization. In Section 2 we give the notation necessary for the formal statement of our results, which is done in Section 3. In Section 4 some of the proofs are presented. For reasons of lack of space, many proofs have been omitted; the reader can find them in the full version of the paper at
`http://homepages.cwi.nl/~david/downloads/implicit.pdf`.

2 Notation

For any permutation $\pi : [n] \to [n]$ and $x \in \{0,1\}^n$, we define $\pi(x)$ as the map on n-bit strings that sends $x = x_1 \dots x_n \in \{0,1\}^n$ to $\pi(x) \triangleq x_{\pi(1)} \dots x_{\pi(n)} \in \{0,1\}^n$. If $f : \{0,1\}^n \to \{0,1\}$, we also denote by f^π the function $f^\pi(x) \equiv f(\pi(x))$.

Given $x \in \{0,1\}^n$, $A \subseteq [n]$ and $y \in \{0,1\}^{|A|}$, we denote by $x_{A \leftarrow y}$ an input obtained by taking x and substituting its values in A with y (according to the natural ordering of $[n]$).

For a function $f : \{0,1\}^n \to \{0,1\}$ and a set $A \subseteq [n]$, the *influence*[3] of f on A is

$$Inf_f(A) \triangleq \Pr_{x \in \{0,1\}^n,\ y \in \{0,1\}^{|A|}} \left[f(x) \neq f(x_{A \leftarrow y}) \right].$$

Here and throughout this paper, $x \in S$ under the probability symbol means that an element x is chosen uniformly at random from a set S.

[2] Loosely speaking, these tests try to extract the values of the relevant variables of f^* by querying f on several inputs that are slightly perturbed (see [FKR+02] for details).

[3] When $|A| = 1$, this value is half that of the most common definition of influence of one variable; for consistency we stick to the previous definition instead in this case as well. It also appears in the literature under the alternate name of *variation*.

A set $S \subseteq [n]$ is *relevant* with respect to f if $Inf_f(S) \neq 0$; an index (variable) $i \in [n]$ is relevant if $\{i\}$ is. A *k-junta* is a function g that has at most k relevant variables; equivalently, there is $S \in \binom{[n]}{k}$ such that $Inf_g([n] \setminus S) = 0$.

Jun_k denotes the class of k-juntas (on n variables), and for $A \subseteq [n]$, Jun_A denotes the class of juntas with all relevant variables in A. In addition, given a function $f : \{0,1\}^n \to \{0,1\}$, we denote by $f^* : \{0,1\}^n \to \{0,1\}$ the k-junta that is closest to f (if there are several k-juntas that are equally close, break ties using some arbitrarily fixed scheme). Clearly, if f is itself a k-junta then $f^* = f$.

Given a k-junta $f : \{0,1\}^n \to \{0,1\}$ we define $\mathsf{core}_k(f) : \{0,1\}^k \to \{0,1\}$ to be the restriction of f to its relevant variables (where the variables are placed according to the natural order). In case f has fewer than k relevant variables, $\mathsf{core}_k(f)$ is extended to a function $\{0,1\}^k \to \{0,1\}$ arbitrarily (by adding dummy variables).

Unless explicitly mentioned otherwise, $\mathcal{C}$ will always denote a class of functions $f : \{0,1\}^n \to \{0,1\}$ that is closed under permutation of variables; that is, for any f and permutation π of $[n]$, $f \in \mathcal{C}$ if and only if $f^\pi \in \mathcal{C}$. For any $k \in \mathbb{N}$, let $\mathcal{C}_{[k]}$ denote the subclass $\mathcal{C} \cap \mathsf{Jun}_{[k]}$. Note that since $\mathcal{C}$ is closed under permutations of variables, $\mathcal{C}_{[k]}$ is closed under permutations of the first k variables. With a slight abuse of notation, we may use $\mathsf{core}_k(\mathcal{C}_{[k]})$ to denote the class $\{\mathsf{core}_k(f) : f \in \mathcal{C}_{[k]}\}$ of k-variable functions.

3 Results

3.1 Upper Bounds

The main tool we develop here is the following:

Theorem 1. *Let $\epsilon > 0, k \in \mathbb{N}$ and let $\mathcal{C}_{[k]} \subseteq \mathsf{Jun}_{[k]}$ be a class closed under permutations of the first k variables. Let $\theta_1(k,\epsilon) = (\epsilon/2400)^6/(10^{26}k^{10}) = \mathrm{poly}(\epsilon/k)$. There is a randomized algorithm A_1 that given ϵ, k and oracle access to a function $f : \{0,1\}^n \to \{0,1\}$ does the following:*

- *if* $\mathrm{distiso}(f, \mathcal{C}_{[k]}) \leq \theta_1(k,\epsilon)$, *$A_1$ accepts with probability at least $7/10$;*
- *if* $\mathrm{distiso}(f, \mathcal{C}_{[k]}) \geq \epsilon$, *$A_1$ rejects with probability at least $7/10$;*
- *A_1 makes $O\left(\frac{k}{\epsilon} + k\log k + \frac{1+\log|\mathcal{C}_{[k]}|}{\epsilon^2}\right)$ queries to f.*

Coupled with the prior discussion on testing by implicit learning, Theorem 1 also implies:

Corollary 1. *Let $\epsilon > 0$ and let $\mathcal{C}$ be an isomorphism-invariant class of Boolean functions. In addition, let $k \in \mathbb{N}$ be such that for every $f \in \mathcal{C}$, $\mathrm{distiso}(f, \mathcal{C}_{[k]}) \leq \theta_1(k,\epsilon)$. Then there is an algorithm that makes*

$$O\left(\frac{k}{\epsilon} + k\log k + \frac{1+\log|\mathcal{C}_{[k]}|}{\epsilon^2}\right) = O\left(\frac{k\log k + \log|\mathcal{C}_{[k]}|}{\epsilon^2}\right)$$

queries and satisfies:

– *if* $f \in \mathcal{C}$*, it accepts with probability at least* $7/10$*;*
– *if* $\mathrm{dist}(f, \mathcal{C}) \geq \epsilon$*, it rejects with probability at least* $7/10$*.*

To minimize the query complexity, we would like to pick k as small as possible, subject to the requirement of the theorem. Let $k^\star(\mathcal{C}, \tau)$ be the smallest $k \in \mathbb{N}$ such that for every $f \in \mathcal{C}$, $\mathrm{distiso}(f, \mathcal{C}_{[k]}) \leq \tau$; intuitively, this condition means that $\mathcal{C}$ is τ-approximated by $\mathcal{C}_{[k]}$. We take from [DLM+07] the bounds on $k^\star = k^\star(\mathcal{C}, \tau)$ and $|\mathcal{C}_{[k^\star]}|$ for the following classes of functions:

	$\mathcal{C}$ (class)	$k^\star \triangleq k^\star(\mathcal{C}, \tau) \leq$	$\lvert\mathcal{C}_{[k^\star]}\rvert \leq$
1	s-term DNFs	$s \log(s/\tau)$	$(2s \log(s/\tau))^{s \log(s/\tau))}$
2	size-s Boolean formulae	$s \log(s/\tau)$	$(2s \log(s/\tau))^{s \log(s/\tau)+s}$
3	size-s Boolean circuits	$s \log(s/\tau)$	2^{2s^2+4s}
4	s-sparse polynomials over $\mathbb{F}_2$	$s \log(s/\tau)$	$(2s \log(s/\tau))^{s \log(s/\tau))}$
5	size-s decision trees	s	$(8s)^s$
6	size-s branching programs	s	$s^s(s+1)^{2s}$
7	functions with Fourier degree at most d	$d2^d$	$2^{d^2 2^{2d}}$

These bounds hold for any approximation parameter $\tau \geq 0$. But to make Corollary 1 applicable, we need to pick τ and k such that the (circular) inequalities $\tau \leq \theta_1(k, \epsilon)$ and $k \geq k^\star(\mathcal{C}, \tau)$ are satisfied.

For items $5, 6, 7$ setting $\tau = 0$ does the job; the reason these bounds are independent of τ is the fact that the corresponding classes contain only functions that actually are $k^\star$-juntas (rather than functions that can be well approximated by $k^\star$-juntas).

For the first 4 items we can set $\tau = \theta_1(s, \epsilon)^2$. It is easy to verify that this satisfies the foregoing pair of inequalities. Furthermore, since $\theta_1(s, \epsilon)$ is polynomial in ϵ/s, we get $k = O(s(\log s + \log 1/\epsilon))$. Plugging in the resulting values into Corollary 1, we obtain the following query-complexity bounds:

Class	This work	[DLM+07], [PRS02] (*)
s-term DNFs, size-s Boolean formulae, s-sparse polynomials over $\mathbb{F}_2$, size-s decision trees, size-s branching programs	$\widetilde{O}(s/\epsilon^2)$	$\widetilde{O}(s^4/\epsilon^2)$
size-s Boolean circuits	$\widetilde{O}(s^2/\epsilon^2)$	$\widetilde{O}(s^6/\epsilon^2)$
functions with Fourier degree at most d	$\widetilde{O}(2^{2d}/\epsilon^2)$	$\widetilde{O}(2^{6d}/\epsilon^2)$
s-term monotone DNFs	$\widetilde{O}(s/\epsilon^2)$	$\widetilde{O}(s^2/\epsilon)^*$

3.2 Lower Bounds

In order to analyze how close to optimal our testers are, we need lower bounds on the problems studied here. By using constructions of k-wise independent generators in restricted computational models (in particular the ones of Healy and Viola [HV06]), we improve some of the existing lower bounds and rederive others (refer to the full version of this paper for details):

1. size-s boolean formulae, branching programs and boolean circuits: poly(s).
2. functions with Fourier degree d: $\Omega(d)$.
3. s-sparse polynomials over $GF(2)$: $\Omega(\sqrt{s})$.
4. s-term DNFs, size-s decision trees: $\Omega(\log s)$.

We remark that from independent work of Blais, Brody and Matulef [BBM11] follows a stronger lower bound of $\Omega(s)$ queries for s-sparse polynomials. They also obtain the $\Omega(\log s)$ lower bounds for DNFS and decision trees.

4 Proof of Theorem 1

4.1 Overview

A key component of our algorithm is the nearly optimal junta tester of [Bla09]. This is a test to distinguish k-juntas from functions that are ϵ-far from being one, and has perfect completeness, i.e., never rejects a k-junta (see Section 4.4 for a more detailed description). The tester is not guaranteed to accept functions that are, say, $\epsilon/10$ close to juntas. We observe, however, that it enjoys a certain weak form of *tolerance*; roughly speaking, $\theta_{\boxed{1}}(k,\epsilon)$ is a measure of the amount of tolerance of said tester, i.e. how close f must be to a k-junta in order to guarantee it will be accepted with high probability. This is Lemma 7 in Section 4.4.

Our algorithm begins by calling the junta tester with parameter k. If f is $\theta_{\boxed{1}}(k,\epsilon)$-close to being a k-junta, the aforementioned tolerance implies that f is not rejected. (Note however that f may be $\theta_{\boxed{1}}(k,\epsilon)$-far from any k-junta and still be accepted with high probability, as long as it is ϵ-close to some k-junta.) The tester also returns a set of k blocks (disjoint subsets of indices of the n variables) such that there is a k-junta h that is $O(\epsilon)$-close to f and has all its relevant variables in one of the k blocks, with no block containing more than one relevant variable. Such an h must be itself $O(\epsilon)$ close to f^* as well. Using these properties, we then obtain a noisy sampler for the core of f^*, which on each execution makes one query to f and outputs a pair $(x,a) \in \{0,1\}^k \times \{0,1\}$ such that $\mathsf{core}_k(f^*) = a$ with high probability.

Intuitively, the idea is that such samples may be obtained by making queries to f on certain strings $y \in \{0,1\}^n$ that are constant inside each of the blocks, so that we know the values that y sets on the (unknown) relevant variables of h (which is sufficiently close to both f and f^*). While such y's are far from being uniformly distributed, the approach can be shown to work most of the time. These samples are then used to test isomorphism between $\mathsf{core}_k(f^*)$ and the functions in $\mathcal{C}_{[k]}$; in this final step we allow a small, possibly correlated, fraction of the samples to be incorrectly labelled.

4.2 Main Lemmas and Proof of Theorem 1

We start with the notion of a noisy sampler.

Definition 1. *Let $g : \{0,1\}^k \to \{0,1\}$ be a function, and let $\eta, \mu \in [0,1)$. An (η,μ)*-noisy sampler for g *is a probabilistic algorithm $\widetilde{g}$ that on each execution outputs $(x,a) \in \{0,1\}^k \times \{0,1\}$ such that*

- *for all $\alpha \in \{0,1\}^k$, $\Pr[x = \alpha] = \frac{1}{2^k}(1 \pm \mu)$;*
- *$\Pr[a = g(x)] \geq 1 - \eta$;*
- *the pairs output on each execution are mutually independent.*

An η-noisy sampler is an $(\eta, 0)$-noisy sampler, i.e. one that on each execution picks a uniformly random x. [4]

Now assume that f is very close to a k-junta $g : \{0,1\}^n \to \{0,1\}$, and we have been given an η-noisy sampler for $\mathsf{core}_k(g) : \{0,1\}^k \to \{0,1\}$. Then we can use a variant of Occam's razor to test (tolerantly) whether g is close to some function from a given class $\mathcal{S}$:

Lemma 1. *There is an algorithm that given $\epsilon \in \mathbb{R}^+$, $k \in \mathbb{N}$, a set $\mathcal{S}$ of Boolean functions on $\{0,1\}^k$, and an η-noisy sampler $\widetilde{g}$ for some $g : \{0,1\}^k \to \{0,1\}$, where $\eta \leq \epsilon/100$, satisfies the following:*

- *if $\mathrm{dist}(g, \mathcal{S}) < \epsilon/10$, it accepts with probability at least $9/10$;*
- *if $\mathrm{dist}(g, \mathcal{S}) > 9\epsilon/10$, it rejects with probability at least $9/10$;*
- *it draws $O\left(\frac{1+\log|\mathcal{S}|}{\epsilon^2}\right)$ samples from $\widetilde{g}$.*

Now is the time to state the main technical lemma.

Lemma 2 (Construction of efficient noisy samplers)
There are algorithms A_P, A_S (resp. preprocessor and sampler), both of which having oracle access to a function $f : \{0,1\}^n \to \{0,1\}$, and satisfying the following properties:

The preprocessor A_P takes $\epsilon > 0, k \in \mathbb{N}$ as inputs, makes $O(k/\epsilon + k \log k)$ queries to f and can either reject or accept and return a state $\alpha \in \{0,1\}^{\mathrm{poly}(n)}$. Assuming A_P accepted, the sampler A_S can be called on demand, with state α as an argument; in each call, A_S makes only one query to f and outputs a pair $(x,a) \in \{0,1\}^k \times \{0,1\}$.

On termination of the preprocessing stage A_P, all the following conditions are fulfilled with probability at least $4/5$:

- *If f is $\theta_{\mathrm{J}}(k,\epsilon)$-close to a k-junta, A_P has accepted f;*
- *If f is $\epsilon/2400$-far from a k-junta, A_P has rejected f;*
- *If A_P has accepted, state α is such that, for some permutation $\pi : [k] \to [k]$, $A_S(\alpha)$ is an $\epsilon/100$-noisy sampler for $\mathsf{core}_k(f^*)^\pi$.*

The statement is somewhat technical and calls for careful reading. It is crucial that the last condition be satisfied with high probability for *any* f. When

[4] The reader familiar with [CGM11] should beware that the usage of the parameter μ here is slightly different from that of the similar definition thereof.

$\theta_{\boxed{1}}(k,\epsilon) < \mathrm{dist}(f,\mathsf{Jun}_k) < \epsilon/2400$, it might be the case that A_P always accepts f, always rejects f, or anything in between, but with high probability either f has been rejected *or* an $\epsilon/100$-noisy sampler for (a permutation of) $\mathsf{core}_k(f^*)$ has been constructed.

Assuming Lemmas 2 and 1 we can prove our main theorem.

Proof (of Theorem 1). Let $\tau \triangleq \theta_{\boxed{1}}(k,\epsilon)$. Suppose first that $\mathrm{distiso}(f,\mathcal{C}_{[k]}) \le \tau$. Then Lemma 2 says that, with probability at least $4/5$, we can construct an $\epsilon/100$-noisy sampler for $\mathsf{core}_k(f^*)$. Since $\mathrm{dist}(f,f^*) \le \tau$ and $\mathrm{dist}(f,\mathcal{C}_{[k]}) \le \tau$, we actually obtain an $\epsilon/100$-noisy sampler for a function that is $2\tau < \epsilon/10$-close to the core of some $g \in \mathcal{C}_{[k]}$. Using this noisy sampler we may apply the algorithm from Lemma 1 with $\mathcal{S} = \mathsf{core}_k(\mathcal{C}_{[k]})$, which in turn will accept with probability at least $9/10$. The overall acceptance probability in this case is at least $7/10$ by the union bound.

Now consider the case $\mathrm{distiso}(f,\mathcal{C}_{[k]}) \ge \epsilon$. There are two possible sub cases:

$\mathrm{dist}(f,\mathsf{Jun}_k) > \epsilon/2400$**:** In this case f is rejected with probability at least $4/5 > 7/10$.

$\mathrm{dist}(f,\mathsf{Jun}_k) \le \epsilon/2400$**:** In this case, with probability at least $4/5$, either f is rejected (in which case we are done), or an $\epsilon/100$-noisy sampler has been constructed for $\mathsf{core}_k(f^*)$. Since f^* is $\epsilon/2400$-close to f, by the triangle inequality we have $\mathrm{dist}(\mathsf{core}_k(f^*),\mathsf{core}_k(\mathcal{C}_{[k]})) \ge \mathrm{distiso}(f,\mathcal{C}_{[k]}) - \mathrm{dist}(f,f^*) > 9\epsilon/10$, and hence with probability at least $9/10$ the algorithm from Lemma 1 rejects. Thus the overall rejection probability in this case is at least $7/10$ too.

The assertion about the number of queries is easily seen to be correct, as it is the sum of the number of queries made in the preprocessing stage by A_P, and the number of executions of the sampler A_S.

The rest of this section is devoted to the proof of Lemma 2.

4.3 Additional Definitions and Lemmas

Our first observation is that, using rejection sampling, one can easily obtain an exactly uniform sampler (as required in Lemma 1) from a slightly non-uniform sampler at the cost of a small increase in the error probability:

Lemma 3. *Let $\widetilde{g}$ be an (η,μ)-noisy sampler for $g : \{0,1\}^k \to \{0,1\}$, that on each execution picks x according to some fixed distribution D. Then $\widetilde{g}$ can be turned into an $(\eta+\mu)$-noisy sampler $\widetilde{g}_{uniform}$ for g.*

We remark that the conversion made in Lemma 3 is only possible when the distribution D is known. However, this will be the case for the sampler that we construct here.

Throughout the rest of this section, a random partition $\mathcal{I} = I_1,\ldots,I_\ell$ of $[n]$ into ℓ sets is constructed by starting with ℓ empty sets, and then putting each coordinate $i \in [n]$ into one of the ℓ sets picked uniformly at random. Unless

explicitly mentioned otherwise, $\mathcal{I}$ will always denote a random partition $\mathcal{I} = I_1, \ldots, I_\ell$ of $[n]$ into ℓ subsets, where ℓ is even; and $\mathcal{J} = J_1, \ldots, J_k$ will denote an (ordered) k-subset of $\mathcal{I}$ (meaning that there are $a_1, \ldots, a_k$ such that $J_i = I_{a_i}$ for all $i \in [k]$).

Definition 2 (Operators replicate and extract). *We call $y \in \{0,1\}^n$ $\mathcal{I}$-blockwise constant if the restriction of y on every set of $\mathcal{I}$ is constant; that is, if for all $i \in [\ell]$ and $j, j' \in I_i$, $y_j = y_{j'}$.*

- *Given $z \in \{0,1\}^\ell$, define $\mathsf{replicate}_{\mathcal{I}}(z)$ to be the $\mathcal{I}$-blockwise constant string $y \in \{0,1\}^n$ obtained by setting $y_j \leftarrow z_i$ for all $i \in \ell$ and $j \in I_i$.*
- *Given an $\mathcal{I}$-blockwise constant $y \in \{0,1\}^n$ and an ordered subset $\mathcal{J} = (J_1, \ldots, J_k)$ of $\mathcal{I}$ define $\mathsf{extract}_{\mathcal{I},\mathcal{J}}(y)$ to be the string $x \in \{0,1\}^k$ where for every $i \in [k]$: $x_i = y_j$ if $j \in J_i$; and x_i is a uniformly random bit if $J_i = \emptyset$.*

Definition 3 (Distributions $\mathcal{D}_{\mathcal{I}}$ and $\mathcal{D}_{\mathcal{J}}$). *For any $\mathcal{I}$ and $\mathcal{J} \subseteq \mathcal{I}$ as above, we define a pair of distributions:*

- *The distribution $\mathcal{D}_{\mathcal{I}}$ on $\{0,1\}^n$: A random $y \sim \mathcal{D}_{\mathcal{I}}$ is obtained by*
  1. *picking $z \in \{0,1\}^\ell$ uniformly at random among all $\binom{\ell}{\ell/2}$ strings of weight $\ell/2$;*
  2. *setting $y \leftarrow \mathsf{replicate}_{\mathcal{I}}(z)$.*
- *The distribution $\mathcal{D}_{\mathcal{J}}$ on $\{0,1\}^{|\mathcal{J}|}$: A random $x \sim \mathcal{D}_{\mathcal{J}}$ is obtained by*
  1. *picking $y \in \{0,1\}^n$ at random, according to $\mathcal{D}_{\mathcal{I}}$;*
  2. *setting $x \leftarrow \mathsf{extract}_{\mathcal{I},\mathcal{J}}(y)$.*

Lemma 4 (Properties of $\mathcal{D}_{\mathcal{I}}$ and $\mathcal{D}_{\mathcal{J}}$)

1. *For all $\alpha \in \{0,1\}^n$, $\Pr_{\mathcal{I}, y \sim \mathcal{D}_{\mathcal{I}}}[y = \alpha] = 1/2^n$;*
2. *Assume $\ell > 4|\mathcal{J}|^2$. For every $\mathcal{I}$ and $\mathcal{J} \subseteq \mathcal{I}$, the total variation distance between $\mathcal{D}_{\mathcal{J}}$ and the uniform distribution on $\{0,1\}^{|\mathcal{J}|}$ is bounded by $2|\mathcal{J}|^2/\ell$. Moreover, the total variation distance between the two distributions is at most $4|\mathcal{J}|^2/(\ell 2^{|\mathcal{J}|})$.*

Definition 4 (Algorithm $\mathsf{sampler}_{\mathcal{I},\mathcal{J}}(f)$). *Given $\mathcal{I}, \mathcal{J}$ as above and oracle access to $f : \{0,1\}^n \to \{0,1\}$, we define a probabilistic algorithm $\mathsf{sampler}_{\mathcal{I},\mathcal{J}}(f)$, that on each execution produces a pair $(x, a) \in \{0,1\}^{|\mathcal{J}|} \times \{0,1\}$ as follows: first it picks a random $y \sim \mathcal{D}_{\mathcal{I}}$, then it queries f on y, and outputs the pair $(\mathsf{extract}_{\mathcal{I},\mathcal{J}}(y), f(y))$.*

Jumping ahead, we remark that the pair $\mathcal{I}, \mathcal{J}$ (along with the values of k, ϵ) will be the information encoded in state α referred to in Lemma 2. In order to ensure that the last condition there is satisfied, we need to impose certain conditions on $\mathcal{I}$ and $\mathcal{J}$.

Definition 5. *Given $\delta > 0$, a function $f : \{0,1\}^n \to \{0,1\}$, a partition $\mathcal{I} = I_1, \ldots, I_\ell$ of $[n]$ and a k-subset $\mathcal{J}$ of $\mathcal{I}$, we call the pair $(\mathcal{I}, \mathcal{J})$ δ-good (with respect to f) if there exists a k-junta $h : \{0,1\}^n \to \{0,1\}$ such that the following conditions are satisfied:*

1. *Conditions on h:*
   (a) *Every relevant variable of h is also a relevant variable of f^* (recall that f^* denotes the k-junta closest to f);*
   (b) $\mathrm{dist}(f^*, h) < \delta$.
2. *Conditions on $\mathcal{I}$:*
   (a) *For all $j \in [\ell]$, I_j contains at most one variable of $\mathsf{core}_k(f^*)$;* [5]
   (b) $\Pr_{y \sim \mathcal{D}_{\mathcal{I}}}[f(y) \neq f^*(y)] \leq 10 \cdot \mathrm{dist}(f, f^*)$;
3. *Conditions on $\mathcal{J}$:*
   (a) *The set $\bigcup_{I_j \in \mathcal{J}} I_j$ contains* all *relevant variables of h;*

Lemma 5. *Let $\delta, f, \mathcal{I}, \mathcal{J}$ be as in the preceding definition. If the pair $(\mathcal{I}, \mathcal{J})$ is δ-good (with respect to f), then $\mathsf{sampler}_{\mathcal{I},\mathcal{J}}(f)$ is an (η, μ)-noisy sampler for some permutation of $\mathsf{core}_k(f^*)$, with $\eta \leq 2\delta + 4k^2/\ell + 10 \cdot \mathrm{dist}(f, f^*)$ and $\mu \leq 4k^2/\ell$.*

This is essentially Lemma 8.3 in [CGM11].

As the lemma suggests, our next goal is to obtain a good pair $(\mathcal{I}, \mathcal{J})$. For this we need to prove that (a slight variation of) the junta tester from [Bla09] satisfies certain properties.

4.4 Junta Testers, Smoothness, and Tolerance

Consider a property $\mathcal{P}$ of Boolean functions on $\{0,1\}^n$ and an ϵ-tester T for it that makes q queries and has success probability $1-\delta$. Let r denote a random seed (so that we can view the tester as a deterministic algorithm with an additional input r) and let $Q(f, r) \subseteq \{0,1\}^n$ be the set of queries it makes on input f and seed r. Define $Q(r) \triangleq \bigcup_f Q(f, r)$; this is the set of all possible queries T may make as f ranges over all possible functions, once r is fixed. We call $p \triangleq \max_r |Q(r)|$ the *non-adaptive complexity* of the tester. If $q = p$ then the tester is essentially non-adaptive; and clearly $p \leq 2^q$ holds for any tester of Boolean properties. We observe that for the junta tester of Blais [Bla09], p is in fact polynomially bounded in q. (Without loss of generality we assume that $Q(r)$ is never empty.)

Definition 6. *A tester is* p-smooth *if its non-adaptive complexity is at most p and for all $\alpha \in \{0,1\}^n$,*

$$\Pr_{\substack{r \\ y \in Q(r)}}[y = \alpha] = \frac{1}{2^n}.$$

Notice that y is picked uniformly at random from the set $Q(r)$, regardless of the probability y would be queried by T on any particular f. In other words, we are picking one random query of the non-adaptive version of T that queries all of $Q(r)$ in bulk, and requiring that the resulting string be uniformly distributed.

[5] Note that this with 1a implies that every block I_j contains at most one relevant variable of h, since the variables of $\mathsf{core}_k(f^*)$ contain all relevant variables of f^*.

Lemma 6. *Let T be a p-smooth tester for $\mathcal{P}$ that accepts every $f \in \mathcal{P}$ with probability at least $1-\delta$. Then for every $f : \{0,1\}^n \to \{0,1\}$, $\Pr[T \text{ accepts } f] \geq 1 - \delta - p \cdot \mathrm{dist}(f, \mathcal{P})$.*

Proof. Choose any $f' \in \mathcal{P}$ and let $\Delta \triangleq \{y \in \{0,1\}^n : f(y) \neq f'(y)\}$. By the union bound, the probability (over r) that $Q(r)$ intersects Δ is at most $\mu \triangleq p \cdot \mathrm{dist}(f, f')$, and hence the probability is at least $1-\mu$ that the tester reaches the same decision about f as it does about f'. But the probability that f' is rejected is at most δ, hence the claim follows.

Lemma 7. *The one-sided error junta tester $\mathrm{T}_{\text{Bla09}}$ from [Bla09] is $p_{\text{T}_{\text{Bla09}}}(k, 1/\epsilon)$-smooth, where $p_{\text{T}_{\text{Bla09}}}(k, 1/\epsilon) \triangleq (10^{25}k^{10})/\epsilon^6$. Thus, by Lemma 6, it accepts functions that are $\theta_{\text{T}_{\text{Bla09}}}(k,\epsilon)$-close to Jun_k with probability at least $9/10$ (since $10 \cdot \theta_{\text{T}_{\text{Bla09}}}(k,\epsilon) \leq 1/p_{\text{T}_{\text{Bla09}}}(k, 1/\epsilon)$.) It also rejects functions that are ϵ-far from Jun_k with probability at least $2/3$, as proved in [Bla09].*

4.5 Obtaining a Good Pair $(\mathcal{I}, \mathcal{J})$

In the following proposition we claim that the tester $\mathrm{T}_{\text{Bla09}}$ satisfies several conditions that we need for obtaining the aforementioned sampler.

Proposition 1. *There is a tester $\mathrm{T}_{\text{Bla09}}$ for Jun_k that makes $O(k \log k + k/\epsilon)$ queries, takes a (random) partition $\mathcal{I} = I_1, \ldots, I_\ell$ of $[n]$ as input, where $\ell = \Theta(k^9/\epsilon^5)$ is even, and outputs (in case of acceptance) a k-subset $\mathcal{J}$ of $\mathcal{I}$ such that for any f the following conditions hold (the probabilities below are taken over the randomness of the tester and the construction of $\mathcal{I}$):*

- *if f is $\theta_{\text{T}_{\text{Bla09}}}(k,\epsilon)$ close to Jun_k, $\mathrm{T}_{\text{Bla09}}$ accepts with probability at least $9/10$;*
- *if f is $\epsilon/2400$-far from Jun_k, $\mathrm{T}_{\text{Bla09}}$ rejects with probability at least $9/10$;*
- *for any f, with probability at least $4/5$ either $\mathrm{T}_{\text{Bla09}}$ rejects, or it outputs $\mathcal{J}$ such that the pair $(\mathcal{I}, \mathcal{J})$ is $\epsilon/600$-good (as per Definition 5).*

In particular, if $\mathrm{dist}(f, \mathsf{Jun}_k) \leq \theta_{\text{T}_{\text{Bla09}}}(k,\epsilon)$, then with probability at least $4/5$ $\mathrm{T}_{\text{Bla09}}$ outputs a set $\mathcal{J}$ such that $(\mathcal{I}, \mathcal{J})$ is $\epsilon/600$-good.

We are finally ready to complete the proof of Lemma 2.

4.6 Proof of Lemma 2

We start by describing how A_P and A_S operate: The preprocessor A_P starts by constructing a random partition $\mathcal{I}$ and calling the junta tester $\mathrm{T}_{\text{Bla09}}$. Then, in case $\mathrm{T}_{\text{Bla09}}$ accepted, A_P encodes in the state α the partition $\mathcal{I}$ and the subset $\mathcal{J} \subseteq \mathcal{I}$ output by $\mathrm{T}_{\text{Bla09}}$ (see Proposition 1), along with the values of k and ϵ. The sampler A_S, given α, obtains a pair $(x, a) \in \{0,1\}^k \times \{0,1\}$ by executing $\mathsf{sampler}_{\mathcal{I},\mathcal{J}}(f)$ (once).

Now we show how Lemma 2 follows from Proposition 1. The first two items are immediate. As for the third one, notice that we only have to analyze the case where $\mathrm{dist}(f, f^*) \leq \epsilon/2400$ and T [Bla09] accepted; all other cases are taken care of by the first two items. By the third item in Proposition 1, with probability at least $4/5$ the pair $(\mathcal{I}, \mathcal{J})$ is $\epsilon/600$-good. If so, by Lemma 5 $\mathsf{sampler}_{\mathcal{I},\mathcal{J}}(f)$ is an (η, μ)-noisy sampler for some permutation of $\mathsf{core}_k(f^*)$, with $\eta \leq \epsilon/300 + 4k^2/\ell + 10 \cdot \mathrm{dist}(f, f^*) \leq \epsilon/120 + 4k^2/\ell$ and $\mu \leq 4k^2/\ell$. The final step we apply is the conversion from Lemma 3, with which we obtain a $(\epsilon/120 + 4k^2/\ell + 4k^2/\ell) \leq (\epsilon/100)$-noisy sampler for some permutation of $\mathsf{core}_k(f^*)$. □

Acknowledgement

We are grateful to Noga Alon, Eric Blais and Eldar Fischer for very useful discussions, and to Bruno Loff for bringing the paper [HV06] to our attention.

References

[BBM11] Blais, E., Brody, J., Matulef, K.: Property testing lower bounds via communication complexity. Personal communication (2011)

[Bla09] Blais, E.: Testing juntas nearly optimally. In: Proc. ACM Symposium on the Theory of Computing, pp. 151–158. ACM, New York (2009)

[CGM11] Chakraborty, S., García-Soriano, D., Matsliah, A.: Nearly tight bounds for testing function isomorphism. In: Proc. of the ACM-SIAM Symposium on Discrete Algorithms, SODA (2011)

[DLM+07] Diakonikolas, I., Lee, H.K., Matulef, K., Onak, K., Rubinfeld, R., Servedio, R.A., Wan, A.: Testing for concise representations. In: Proc. IEEE Symposium on Foundations of Computer Science, pp. 549–558 (2007)

[DLM+08] Diakonikolas, I., Lee, H.K., Matulef, K., Servedio, R.A., Wan, A.: Efficiently testing sparse $GF(2)$ polynomials. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part I. LNCS, vol. 5125, pp. 502–514. Springer, Heidelberg (2008)

[FKR+02] Fischer, E., Kindler, G., Ron, D., Safra, S., Samorodnitsky, A.: Testing juntas. In: FOCS, pp. 103–112 (2002)

[GOS+09] Gopalan, P., O'Donnell, R., Servedio, R.A., Shpilka, A., Wimmer, K.: Testing fourier dimensionality and sparsity. In: Albers, S., Marchetti-Spaccamela, A., Matias, Y., Nikoletseas, S., Thomas, W. (eds.) ICALP 2009. LNCS, vol. 5555, pp. 500–512. Springer, Heidelberg (2009)

[HV06] Healy, A., Viola, E.: Constant-depth circuits for arithmetic in finite fields of characteristic two. In: Durand, B., Thomas, W. (eds.) STACS 2006. LNCS, vol. 3884, pp. 672–683. Springer, Heidelberg (2006)

[PRS02] Parnas, M., Ron, D., Samorodnitsky, A.: Testing basic boolean formulae. SIAM J. Discrete Math. 16(1), 20–46 (2002)

[Ser10] Servedio, R.A.: Testing by implicit learning: a brief survey (2010)

Efficiently Decodable Error-Correcting List Disjunct Matrices and Applications
(Extended Abstract)

Hung Q. Ngo[1], Ely Porat[2], and Atri Rudra[1,⋆]

[1] Department of CSE, University at Buffalo, SUNY, Buffalo, NY, 14260, USA
[2] Department of Computer Science, Bar-Ilan University, Ramat Gan 52900, Israel

Abstract. A (d, ℓ)-list disjunct matrix is a non-adaptive group testing primitive which, given a set of items with at most d "defectives," outputs a superset of the defectives containing less than ℓ non-defective items. The primitive has found many applications as stand alone objects and as building blocks in the construction of other combinatorial objects.

This paper studies error-tolerant list disjunct matrices which can correct up to e_0 false positive and e_1 false negative tests in sub-linear time. We then use list-disjunct matrices to prove new results in three different applications.

Our major contributions are as follows. (1) We prove several (almost)-matching lower and upper bounds for the optimal number of tests, including the fact that $\Theta(d \log(n/d) + e_0 + de_1)$ tests is necessary and sufficient when $\ell = \Theta(d)$. Similar results are also derived for the disjunct matrix case (i.e. $\ell = 1$). (2) We present two methods that convert error-tolerant list disjunct matrices in a *black-box* manner into error-tolerant list disjunct matrices that are also *efficiently decodable*. The methods help us derive a family of (strongly) explicit constructions of list-disjunct matrices which are either optimal or near optimal, and which are also efficiently decodable. (3) We show how to use error-correcting efficiently decodable list-disjunct matrices in three different applications: (i) explicit constructions of d-disjunct matrices with $t = O(d^2 \log n + rd)$ tests which are decodable in poly(t) time, where r is the maximum number of test errors. This result is optimal for $r = \Omega(d \log n)$, and even for $r = 0$ this result improves upon known results; (ii) (explicit) constructions of (near)-optimal, error-correcting, and efficiently decodable monotone encodings; and (iii) (explicit) constructions of (near)-optimal, error-correcting, and efficiently decodable multiple user tracing families.

1 Introduction

The basic objective of *group testing* is to figure out a subset of "defective items" in a large item population by performing tests on subsets of items. The manifestation of "defective" and "tests" depends on the application. For most of this paper we will consider the basic interpretation where we have a universe $[n]$ of

⋆ Supported by NSF CAREER grant CCF-0844796.

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 557–568, 2011.

items and some subset $S \subset [n]$ of at most d *defectives* (also interchangeably called *positives*). Every (group) test is a subset $T \subseteq [n]$, which results in a *positive outcome* if some defective is in T and a *negative outcome* when T contains no defectives. In many applications, *non-adaptive* group testing is required, where one cannot use one test's outcome to design another test. Non-adaptive group testing ($NAGT$) has found applications in drug and DNA library screening [18], live baiting of DoS attackers [16], data forensics [12] and data streams [4], among others. See the standard monograph on group testing for more details [6].

The first objective in the design of such NAGT primitives is to minimize the number of tests necessary to identify (or *decode*) all the defectives. A NAGT strategy with t tests on n items can be represented by a $t \times n$ binary matrix $\mathbf{M}$ where each row is the incidence vector of the corresponding test. For unique decoding of up to d defectives, it is necessary that all the unions of up to d columns of $\mathbf{M}$ have to be distinct. Such a matrix is said to be *d-separable*. It has been known for a long time that the optimal number of rows of a d-separable matrix is between $\Omega(d^2 \log n / \log d)$ [8] and $O(d^2 \log(n/d))$ [6].

The second objective is to explicitly construct disjunct matrices with as few tests as possible. Recently, a $O(nt)$-time explicit construction attaining the $t = O(d^2 \log(n))$-bound has also been found [20]. No strongly explicit construction matching the bound is known.[1]

The third objective is to decode efficiently. The brute-force algorithm is too slow as it goes through all possible $\binom{n}{\leq d} = O(n^d)$ choices for the defective set. Some NAGT strategies, however, allow a very simple $O(nt)$-time decoding algorithm to work: the decoder simply eliminates items belonging to negative tests and returns the remaining items. We shall refer to this decoder as the *naive decoder*. A NAGT matrix is said to be *d-disjunct* iff the naive decoder works on all possible inputs of up to d defectives. While disjunct matrices are a stronger notion than separable matrices, they have asymptotically the same number of tests [6]. Thus, we went from $O(n^d)$ down to $O(nt)$-decoding time "for free."

The time complexity of $O(nt)$ is reasonable for most of the "traditional" algorithmic applications. However with the proliferation of massive data sets and their numerous applications, the decoding time of $O(nt)$ is no longer good enough because the number of items n is prohibitively large. For example, in typical data stream applications a running time of poly(t) (with $t = O(d^2 \log n)$ tests in the best case) for moderate values of d would imply an exponential improvement in the running time. The question of constructing efficiently decodable disjunct matrices was first explicitly raised by Cormode and Muthukrishnan [4]. Recently, Indyk, Ngo and Rudra [14] presented a randomized construction of d-disjunct matrices with $t = O(d^2 \log(n))$ tests that could be decoded in time poly(t). They also derandomized their construction for $d \leq O(\log n / \log \log n)$. Our construction in this paper removes the above constraint on d. We thus can get further

[1] Throughout this paper we will call a $t \times n$ matrix strongly explicit if any column of the matrix can be constructed in time poly(t). A matrix will be called explicit if it can be constructed in time poly(t, n).

down to poly(t) decoding time "for free." Henceforth, "efficient decoding" means decoding in poly(t)-time.

The fourth objective is to correct errors in test outcomes. In many applications such as drug discovery and DNA library screening, faulty test outcomes are unavoidable [15]. Or, when heavy-hitters in a data-stream are identified using group testing, non-heavy hitter elements might generate false positive tests if the small-tail property is not satisfied [4]. This paper essentially obtains the above "for free" result even *with* the additional error-correcting requirement.

Our NAGT results crucially use as a building block a weaker notion of disjunct matrices called list disjunct matrices, which turns out to have many other useful applications as well.

1.1 List Disjunct Matrices and Our Main Results

Similar to list-decoding, if we relax the exact decoding requirement and only require the NAGT primitive to output a bounded super-set of all the defectives, then separable/disjunct matrices become list-separable/disjunct matrices. Roughly, a (d, ℓ)-list disjunct matrix is one where the naive decoder always outputs a super-set of the defectives containing less than ℓ other non-defective items. The name "list disjunct" was coined in [14], though it was previously studied under the different names: (d, n, ℓ)-*super-imposed codes* in [7,5], *list-decoding super-imposed codes* of strength d and list-size ℓ in [21].[2] We stick with the term "list disjunct matrices" in this paper. Shortly prior to [14], Cheraghchi [3] studied the notion of *error-correcting measurement matrices* for d-sparse vectors, which are slightly more general than the notion of *list-separable matrices*. It can be shown that list-separable matrices are equivalent to list-disjunct matrices with a slight loss in parameters. Hence, the results in [3], though shown for list-separable matrices, apply for list-disjunct matrices as well. Conversely, our bounds also apply to error-correcting measurement matrices. Our lower bounds slightly improve lower bounds in [3].

List-disjunct matrices were used in [14] to construct efficiently decodable disjunct matrices. They also presented a "stand alone" application of list disjunct matrices in constructing sparsity separators, which were used by Ganguly [11] to design data stream algorithms for the sparsity problem. Rudra and Uurtamo [22] used these objects to design data stream algorithms for tolerant testing Reed-Solomon codes. As observed in De Bonis et. al. [5], these objects can also be used to construct optimal two stage group testing algorithms.

List disjunct matrices are also similar to other combinatorial objects such as selectors [13] and multi-user tracing families [2]. Selectors have found numerous applications such as broadcasting in unknown directed networks and designing tests for coin-weighting problems [13] as well as designing optimal two stage group testing schemes [5]. Multi-user tracing families were used to construct monotone encodings in [2]. Monotone encodings can be used for designing secure vote storage systems [17].

[2] The authors of [14] were not aware of these previous works.

Given the various applications of list disjunct matrices, it is very natural to ask if one could (explicitly) construct list disjunct matrices that are also efficiently decodable. Indyk, Ngo and Rudra [14] constructed efficiently decodable $(d, O(d^{1+\epsilon}))$-list-disjunct matrices with $O(d^{1+\epsilon}\log^{1+1/\epsilon} n)$ tests $(\forall \epsilon > 0)$. Cheraghchi [3] constructed efficiently decodable $(d, O(d))$-list-separable matrices (and thus, due to their almost-equivalence, $(d, O(d))$-list-disjunct matrices) with $O(d^{3+\alpha+1/\alpha}\log^{1+1/\alpha} n)$ tests $(\forall \alpha > 0)$.

This paper improves upon both [14] and [3] by presenting efficiently decodable $(d, O(d))$-list-disjunct matrices with $O(d^{1+o(1)}\log n \log\log_d n)$ tests as well as $(d, O(d^{1+\epsilon}))$-list disjunct matrices with $O(d^{1+\epsilon}\log n)$ tests (for any $\epsilon > 0$). In addition, our matrices are also error-correcting. To state the results more precisely we briefly discuss the error-correcting matrices next.

Error-correcting versions of disjunct matrices have been studied a fair bit [6] but to the best of our knowledge the only existing work that considers error tolerant list-separable matrices is [3]. This paper studies the general notion of (d, ℓ, e_0, e_1)-list disjunct/separable matrices which are (d, ℓ)-disjunct/separable matrices capable of correcting up to e_0 false positives and e_1 false negatives.

We prove upper and lower bounds on the optimal number of rows of a (d, ℓ, e_0, e_1)-list disjunct matrix. The bounds are tight for sufficiently large $e_0 + de_1$. (Our lower bounds are slightly better than those in [3].)

We then show how to construct error-tolerant and efficiently decodable list-disjunct matrices by presenting two general procedures which – in a *black-box* fashion – convert any (d, ℓ, e_0, e_1)-list disjunct matrices into (d, ℓ, e_0', e_1')-list disjunct matrices that are also efficiently decodable with a mild blow-up in the number of tests. Note that we essentially show how to convert a combinatorial guarantee into an algorithmic guarantee, which for combinatorial objects (e.g. codes) is generally not obvious.

One of our conversion procedures provides a tradeoff between the blow-up in the number of tests vs. the gain in decoding time (from the simple linear time algorithm). Unfortunately, this procedure can only give $e_0' = e_0$ and $e_1' = e_1$ (even though the number of tests has gone up). Our other conversion procedure does not provide such a nice tradeoff but it does lead to efficient decoding with a mild blow-up in the number of tests. More importantly, the quantities e_0'/e_0 and e_1'/e_1 scale up linearly with the blow-up in the number of tests. This allows us to design error-tolerant $(d, d^{1+\epsilon}, e_0, e_1)$-list disjunct matrices that for large values of $e_0 + de_1$ have essentially all the nice properties one can wish for: (i) Optimal number of tests; (ii) Strongly explicit construction and (iii) Efficiently decodable.

1.2 Applications and Other Results

(Near) Optimal, explicit, and error-correcting disjunct matrices. Constructions of efficiently decodable list disjunct matrices lead to constructions of efficiently decodable d-disjunct matrices with the best known $O(d^2 \log n)$ number of tests. This result settles an open question from [14]. The black-box conversion procedures also work for disjunct matrices. We prove a similar optimal result as in the

list-disjunct case where a disjunct matrix is explicit, error-correcting, efficiently decodable, and has the best known number of tests. In fact, when the number of errors is sufficiently large, our error-tolerant disjunct matrices also have the optimal number of tests. This result points out the following somewhat surprising fact: our understanding of error-tolerant (list) disjunct matrices is essentially limited by our understanding of the traditional no error case. In other words, adding more errors only makes the problem "easier," which is not the case for related combinatorial objects such as codes.

Near Optimal, Efficiently Computable, and Efficiently Decodable Monotone Encodings. With a construction similar to that in Alon-Hod [2], we show that $(d, d/2)$-list disjunct $t \times n$ matrices imply a (n, d)-monotone encoding of length t, i.e. a monotone injective functions from subsets of size up to d of $[n]$ to t bits. By contrast, the Alon-Hod's construction used multi-user tracing families which are, in a sense, duals of list-disjunct matrices. Alon and Hod showed that optimal (n, d)-monotone encodings have length $t = \Theta(d \log (n/d))$, and presented a probabilistic construction with encoding and decoding time of $O(nd \log n)$. From our list-disjunct matrix constructions, we construct (n, d)-monotone encodings with length $t = O((d \log n)^{1+o(1)})$ that are both explicitly computable and decodable in poly(t)-time. Our result also hold for the error-tolerant monotone encodings. We also prove an almost matching lower-bound for the length of error-correcting monotone-encoding.

Near-Optimal Efficiently Decodable Multiple User Tracing Families. Given positive integers $u \leq d$, a (d, u)-multiuser tracing (MUT) family is a NAGT strategy (or matrix) which, given the test outcomes imposed by an arbitrary set of $v \leq d$ defectives, there is a decoding algorithm that outputs at least $\min(u, v)$ out of the v defectives. Thus, in a sense MUT is the dual of list-disjunct matrices. Alon-Asodi [1] proved that, given n, the smallest t for which a $t \times n$ (d, u)-MUT matrix exists must satisfy $\Omega\left(\left(d + \frac{u^2}{\log u}\right) \cdot \log n\right) \leq t \leq O\left((d + u^2) \log n\right)$. From our results on list-disjunct matrices, we show how to construct (d, u)-MUT matrix of size $t \times n$ with $t = O((d^{1+o(1)} + u^2) \log n + (e_0 + e_1)d)$ which are explicit, efficiently decodable and error tolerant. By removing the explicitness requirement, we can retain the other two properties and reduce t to essentially the optimal $O((d + u^2) \log n + (e_0 + e_1)d \log \log n)$.

2 Error-Correcting List-Disjunct/Separable Matrices

Let $\mathbf{M} = (m_{ij})$ be any binary matrix with t rows and n columns. Let $\mathbf{M}^j$ denote the jth column of $\mathbf{M}$. We can also think of the columns $\mathbf{M}^j$ as characteristic vectors of subsets of $[t]$, i.e. $\mathbf{M}^j = \{i \mid m_{ij} = 1\}$. Thus, overloading notation we will apply set operations on the columns of $\mathbf{M}$.

The key combinatorial structure we study in this paper is the error-tolerant version of the notion of *list-disjunct matrix*. Using a (d, ℓ)-list-disjunct $t \times n$ matrix $\mathbf{M}$ we can design a NAGT procedure for n items with at most d defectives.

The decoding algorithm simply eliminates any item which is present in any negative test. When $\mathbf{M}$ is (d, ℓ)-list-disjunct, this "naive decoder" returns a set R of (remaining) items containing all defectives and less than ℓ extra (non-negative) items. In many applications, the test outcomes might have errors. The following combinatorial structure is a generalization of list-disjunct matrices which can correct up to e_0 *false positives* and e_1 *false negatives* in test outcomes.

Definition 1. *Let $d \geq 1, \ell \geq 1, e_0 \geq 0, e_1 \geq 0$, and $n \geq d+\ell$ be given integers. A $t \times n$ binary matrix $\mathbf{M}$ is called a (d, ℓ, e_0, e_1)-list-disjunct matrix if $\mathbf{M}$ satisfies the following conditions. For any disjoint subsets $S, T \subset [n]$ such that $|S| = d, |T| = \ell$, and an arbitrary subset $X \subseteq \left(\bigcup_{j \in T} \mathbf{M}^j\right) \setminus \left(\bigcup_{j \in S} \mathbf{M}^j\right)$ of size $|X| \leq e_0$, there exists a column $\bar{j} \in T \setminus S$ such that $\left|\mathbf{M}^{\bar{j}} \setminus \left(X \cup \bigcup_{j \in S} \mathbf{M}^j\right)\right| \geq e_1 + 1$.*

Proposition 2. *Define the "naive decoder" be the algorithm which eliminates all items belonging to at least $e_1 + 1$ negative tests and returns the remaining items. If $\mathbf{M}$ is (d, ℓ, e_0, e_1)-list-disjunct, then the naive decoder returns a set R of items containing all the (at most d) defectives and at most $\ell - 1$ negative items, even if the test outcomes have up to e_0 false positives and e_1 false negatives.*

3 Intuition behind the Blackbox Conversion Procedures

The first conversion uses ideas similar to those used in [14]: given an error tolerant list disjunct matrix, we concatenate Parvaresh-Vardy codes [19] with it. PV codes have excellent list recoverability properties, which can be exploited to design efficient decoding procedure for the resulting matrix.

Our second conversion procedure is perhaps technically more interesting. The main idea behind the construction is as follows. Say we are trying to construct a (d, ℓ, e_0, e_1)-list-disjunct matrix $\mathbf{M}^*$ with n columns that is efficiently decodable from a family of matrices, where for any $i \geq d$, there is a (d, ℓ, e_0, e_1)-list-disjunct $t(i) \times i$ matrix (not necessarily efficiently decodable). Towards efficient decoding, say we somehow knew that all the positive items are contained in a subset $\mathcal{S} \subseteq [n]$. Then the naive decoder would run in time $O(t(n) \cdot |\mathcal{S}|)$, which would be sublinear if $|\mathcal{S}|$ and $t(n)$ are sufficiently small. The idea is to construct this small set $\mathcal{S}$ recursively.

Fix $n \geq d \geq 1$. Assume there exists a (d, ℓ, e_0, e_1)-list disjunct $t_1 \times \sqrt{n}$ matrix $\mathbf{M}^{(1)}$ that is efficiently decodable and let $\mathbf{M}^{(2)}$ be a (d, ℓ, e_0, e_1)-list disjunct $t_2 \times n$ matrix (that is not necessarily efficiently decodable). Let $\mathbf{M}^L$ be the $t_1 \times n$ matrix where the ith column (for $i \in [n]$) is identical to the jth column of $\mathbf{M}^{(1)}$ such that the *first* $\frac{1}{2} \log n$ bits of i is j (where we think of i and j as their respective binary representations). Similarly, let $\mathbf{M}^R$ be the $t_1 \times n$ matrix where the *last* $\frac{1}{2} \log n$ bits of i is j.

Let $S \subseteq [n], |S| \leq d$, be an arbitrary set of positives. Let the vector $\mathbf{r}^L$ ($\mathbf{r}^R$, resp.) be the vector that results from applying $\mathbf{M}^L$ ($\mathbf{M}^R$, resp.) on S. Note that $\mathbf{r}^L$ and $\mathbf{r}^R$ might be different from the unions $\cup_{j \in S} (\mathbf{M}^L)^j$ and $\cup_{j \in S} (\mathbf{M}^R)^j$,

respectively, due to the $(e_0, e_1$-bounded) errors in test outcomes. Apply the decoding algorithm for $\mathbf{M}^{(1)}$ to $\mathbf{r}^L$ ($\mathbf{r}^R$, resp.) and obtain the set S_L (S_R, resp.) of $\frac{1}{2}\log n$-bit vectors such that, for every $i \in S$, the first (last, resp.) $\frac{1}{2}\log n$ bits of i belongs to S_L (S_R, resp.). In other words, $\mathcal{S} = S_L \times S_R$ contains all the indices $i \in S$. Further, note that both $|S_L|$ and $|S_R|$ have less than $d + \ell$ elements.

Now, our final matrix $\mathbf{M}^*$ is simple: just vertically stack $\mathbf{M}^L$, $\mathbf{M}^R$ and $\mathbf{M}^{(2)}$ together. Note that $\mathbf{M}^*$ is (d, ℓ, e_0, e_1)-list disjunct because $\mathbf{M}^{(2)}$ is (d, ℓ, e_0, e_1)-list disjunct. Finally, decoding $\mathbf{M}^*$ can be done efficiently: first decode the part of the result matrix corresponding to $\mathbf{M}^L$ and $\mathbf{M}^R$ to obtain S_L and S_R respectively – this is efficient as $\mathbf{M}^{(1)}$ is efficiently decodable. Finally computing the output item set (containing S) can be done with an additional $O(t_2 \cdot (d + \ell)^2)$-time as we only need to run the naive decoder for $\mathbf{M}^{(2)}$ over $\mathcal{S} = S_L \times S_R$. To achieve a tradeoff between the number of tests and the decoding time, we choose the parameters of the recursion more carefully.

Our conversion of a disjunct matrix into an efficiently decodable one is very simple: stack an efficiently decodable $(d, \text{poly}(d))$-list disjunct matrix on a d-disjunct matrix. Decoding the result vector from the list disjunct matrix gives a subset of size $\text{poly}(d)$ that contains all the defective items. We then run the naive decoder for the disjunct matrix on this set of possibilities leading to an overall efficient decoding algorithm. Since there is an $\Omega(d/\log d)$ gap in the number of tests needed in a list disjunct matrix and a disjunct matrix, as long as we have an efficiently decodable list disjunct matrix with $o(d^2 \log n/\log d)$ tests, we are fine. Indeed, we get such matrices from our construction mentioned earlier. To obtain the efficiently decodable matrix with $O(d^2 \log n)$ tests we use the d-disjunct matrix construction of Porat and Rothschild [20]. The same idea works for the error-correcting versions.

4 Bounds

Given d, ℓ, e_0, e_1, and n, let $t(d, \ell, e_0, e_1, n)$ denote the minimum number of rows t of a (d, ℓ, e_0, e_1)-list-disjunct matrix with t rows and n columns. It is not hard to see that every $(d, 1, e_0, e_1)$-list-disjunct matrix is the same as a $(d, 1, e_0 + e_1, 0)$-list-disjunct matrix, which is the same as a $(d, 1, 0, e_0 + e_1)$-list-disjunct matrix. Let $r = e_0 + e_1 + 1$. It is customary in the literature to call a $(d, 1, r - 1, 0)$-list-disjunct matrix a *d^r-disjunct matrix* [6]. To shorten the notations, let $t(d, r, n)$ denote $t(d, 1, e_0, e_1, n)$ for the disjunct case, where $r = e_0 + e_1 + 1$.

The following lower bound for (d, ℓ)-list-disjunct matrices is better than the similar bound proved in [5] in two ways: (1) the actual bounds are slightly better, and (2) the bound in [5] requires a precondition that $n > d^2/(4\ell)$ while ours does not. We make use of the argument from Erdős-Frankl-Füredi [9,10], while [5] uses the argument from Ruszinkó [23]. The bound helps prove Theorem 4, which is tight when $\ell = \Theta(d)$.

Lemma 3. *For any n, d, ℓ with $n \geq d+\ell$, we have $t(d, \ell, 0, 0, n) > d \log\left(\frac{n}{d+\ell-1}\right)$. When $d \geq 2\ell$, we have $t(d, \ell, 0, 0, n) > \frac{\lfloor d/\ell \rfloor (d+2-\ell)}{2\log(e\lfloor d/\ell \rfloor (d+2-\ell)/2)} \log\left(\frac{n-d-2\ell+2}{\ell}\right)$.*

Theorem 4. *For any non-negative integers d, ℓ, e_0, e_1, n where $n \geq d + \ell$, we have $t(d, \ell, e_0, e_1, n) = \Omega\left(d \log \frac{n}{d+\ell-1} + e_0 + de_1\right)$. In particular, $t(d, \Theta(d), e_0, e_1, n) = \Omega\left(d \log(n/d) + e_0 + de_1\right)$. Furthermore, when $d \geq 2\ell$ we have $t(d, \ell, e_0, e_1, n) = \Omega\left(\frac{d^2/\ell}{\log(d^2/\ell)} \log \frac{n-d}{\ell} + e_0 + de_1\right)$.*

Theorem 5. *Let n, d, ℓ, e_0, e_1 be given non-negative integers. If $\ell = \Omega(d)$, then $t(d, \ell, e_0, e_1, n) = O\left(d \log(n/d) + e_0 + de_1\right)$. In particular, when $l = \Theta(d)$ we have a precise characterization $t(d, \ell, e_0, e_1, n) = \Theta\left(d \log(n/d) + e_0 + de_1\right)$.*

Theorem 6. *For the d^r-disjunct matrices, we have $t(d, r, n) = O\left(d^2 \log \frac{n}{d} + rd\right)$.*

5 Constructions

We will need the following existing results from the literature.

Theorem 7 ([3]). *Let $1 \leq d \leq n$ be integers. Then there exists a strongly-explicit $t \times n$ matrix that is $(d, O(d), \alpha t, \Omega(t/d))$-list disjunct (for any constant $\alpha \in (0, 1)$) with $t = O(d^{1+o(1)} \log n)$ rows.*

Theorem 8 ([14]). *Let $1 \leq d \leq n$ be integers. Then there exists a strongly-explicit $t \times n$ matrix that is $(d, \delta d, \Omega(t), \Omega(t/d))$-list disjunct (for any constant $\delta > 0$) with $t = O((d \log n)^{1+o(1)})$ rows.*

5.1 Black-Box Conversion Using List Recoverability

Our first black-box procedure converting any error tolerant list disjunct matrix into one that is also efficiently decodable (with a mild sacrifice in some parameters) is based on concatenating PV codes [19] with the given list disjunct matrix.

Theorem 9. *Let $\ell, d \geq 1$, $e_0, e_1 \geq 0$ be integers. Assume that for every $Q \geq d$, there exists a (d, ℓ, e_0, e_1)-list-disjunct $\bar{t}(d, \ell, e_0, e_1, Q) \times Q$ matrix. For every $s \geq 1$ and every $n \geq d$, define $A(d, l, s) = 3(d+l)^{1/s}(s+1)^2$. Let k be minimum integer such that $2k \log(kA(d, l, s)) \geq \log n$, and q be the minimum power of 2 such that $q \geq kA(d, l, s)$. Then, there exists a $(d-1, L, e_0', e_1')$-list disjunct $t' \times n$ matrix with the following properties:*

(i) $t' = O\left(s^2 \cdot (d+\ell)^{1/s} \cdot \left(\frac{\log n}{\log q}\right) \cdot \bar{t}(s)\right)$, *where $\bar{t}(s)$ is shorthand for $\bar{t}(d, \ell, e_0, e_1, q^s)$*
(ii) $e_0' = \gamma e_0$ *and* $e_1' = \gamma e_1$ *where* $\gamma = \Theta(t'/\bar{t}(s))$.
(iii) $L = s^{O(s)} \cdot (d+\ell)^{1+1/s}$.
(iv) It is decodable in time $(t')^{O(s)}$.

Combining Theorem 9 and Theorem 7, we get the following result.

Corollary 10. *Let $\epsilon > 0$ be a real number and let $1 \leq d \leq n$ be integers. Then there exists a strongly-explicit $t \times n$ matrix that is $(d, (1/\epsilon)^{O(1/\epsilon)} \cdot d^{1+\epsilon}, \Omega(t), \Omega(t/d))$-list disjunct with $t = (1/\epsilon)^{O(1/\epsilon)} \cdot d^{1+\epsilon} \cdot \log n$ rows that can be decoded in time $t^{O(1/\epsilon)}$.*

5.2 Black-Box Conversion Using Recursion

Unlike the previous construction, the second procedure gives a more general tradeoff between the blow-up in the number of tests and the resulting decoding time. On the other hand, this conversion uses multiple matrices from a given family of error-tolerant matrices unlike the previous procedure, which only used one error-tolerant list disjunct matrix.

Theorem 11. *Let $n \geq d \geq 1$ be integers. Assume for every $i \geq d$, there is a (d, ℓ, e_0, e_1)-list disjunct $t(i) \times i$ matrix $\mathbf{M}_i$ for integers $1 \leq \ell \leq n - d$ and $e_0, e_1 \geq 0$. Let $1 \leq a \leq \log n$ and $1 \leq b \leq \log n/a$ be integers. Then there exists a $t_{a,b} \times n$ matrix $\mathbf{M}_{a,b}$ that is (d, ℓ, e_0, e_1)-list disjunct that can be decoded in time $D_{a,b}$ where*

$$t_{a,b} = \sum_{j=0}^{\lceil \log_b(\frac{\log n}{a}) \rceil - 1} b^j \cdot t\left(\sqrt[b^j]{n}\right) \tag{1}$$

and

$$D_{a,b} = O\left(t_{a,b} \cdot \left(\frac{\log n \cdot 2^a}{a} + (d + \ell)^b\right)\right). \tag{2}$$

Finally, if the family of matrices $\{\mathbf{M}_i\}_{i \geq d}$ is (strongly) explicit then so is $\mathbf{M}_{a,b}$.

The bound in (1) is somewhat unwieldy. We note in Corollary 12 that when $t(i) = d^x \log^y i$ for some reals $x, y \geq 1$, we can achieve efficient decoding with only a log-log factor increase in number of tests. Theorem 11 with $b = 2$ and $a = \log d$ implies the following:

Corollary 12. *Let $n \geq d \geq 1$ be integers and $x, y \geq 1$ be reals. Assume for every $i \geq d$, there is a (d, ℓ, e_0, e_1)-list disjunct $O(d^x \log^y i) \times i$ matrix for integers $1 \leq \ell \leq n - d$ and $e_0, e_1 \geq 0$. Then there exists a $t \times n$ matrix that is (d, ℓ, e_0, e_1)-list disjunct that can be decoded in $\mathrm{poly}(t, \ell)$ time, where*

$$t \leq O\left(d^x \cdot \log^y n \cdot \log \log_d n\right).$$

Finally, if the original matrices are (strongly) explicit then so is the new one.

Corollary 12 along with Theorems 8 and 7 imply the followings:

Corollary 13. *Let $1 \leq d \leq n$ be integers. For any constant $\delta > 0$ there exists a strongly-explicit $t \times n$ matrix that is $(d, \delta d, \Omega(t/\log\log n), \Omega(t/(d \log\log n)))$-list-disjunct with $t = O((d \log n)^{1+o(1)})$ rows and can be decoded in $\mathrm{poly}(t)$ time.*

Corollary 14. *Let $1 \leq d \leq n$ be integers. For any constant $\alpha \in (0, 1)$ there exists a strongly-explicit $t \times n$ matrix that is $(d, O(d), \alpha t/\log\log n, \Omega(t/(d \log\log n)))$-list disjunct with $t = O(d^{1+o(1)} \log n \log\log n)$ rows and can be decoded in $\mathrm{poly}(t)$ time.*

6 Applications

Efficiently decodable disjunct matrices. The following proposition along with Corollary 10 (with say $\epsilon = 1/2$) lead to the next theorem, a significant improvement over the similar result obtained in [14] that only worked for $d \leq O(\log n / \log\log n)$.

Proposition 15. *Let $n \geq d \geq 1$ be integers. Let M_1 be a (d, ℓ)-list disjunct $t_1 \times n$ matrix that can be decoded in time D. Let M_2 be a d-disjunct $t_2 \times n$ matrix. Then there exists a $(t_1 + t_2) \times n$ matrix that is d-disjunct and can be decoded in time $D + O((d+\ell) \cdot t_2)$. If both M_1 and M_2 are polynomial time constructible then so is the final matrix.*

Theorem 16. *Let $1 \leq d \leq n$. Then there exists a $t \times n$ d-disjunct matrix with $t = O(d^2 \log n)$ that can be decoded in $\mathrm{poly}(t)$ time. Further, the matrix can be computed in time $\tilde{O}(nt)$.*

(Near) Optimal Error-Tolerant, Efficiently Constructible, and Efficiently Decodable (List) Disjunct Matrices. We begin with simple observation that stacking i copies of a list disjunct matrix increases the error-tolerance parameters by a factor of i.

Proposition 17. *If there exists a (d, ℓ, e_0, e_1)-list disjunct matrix with t rows then there exists another $(d, \ell, \alpha \cdot e_0, \alpha \cdot e_1)$-list disjunct matrix with αt rows for any integer $\alpha \geq 1$.*

The surprising thing is that the result above leads to optimal construction of (list) disjunct matrices. In particular, applying Proposition 17 with Corollary 10 implies the following:

Corollary 18. *For every $\epsilon > 0$, there exists a strongly explicit $(d, (1/\epsilon)^{O(1/\epsilon)} \cdot d^{1+\epsilon}, r, r/d)$-list disjunct matrix with $O(t + r)$ rows, where $t = O((1/\epsilon)^{O(1/\epsilon)} \cdot d^{1+\epsilon} \cdot \log n)$ that can be decoded in time $r \cdot t^{O(1/\epsilon)}$.*

Note that Theorem 4 shows that the number of tests in the result above is optimal for $r \geq \Omega(t)$. We also get the following result with the construction in [20], Corollary 18, and Proposition 17. The result is optimal for $r = \Omega(d \log n)$, by Theorem 4.

Corollary 19. *Let $1 \leq d \leq n$ and $r \geq 1$ be integers. Then there exists a $t \times n$ d^r-disjunct matrix with $t = O(d^2 \log n + rd)$ decodable in $r \cdot \mathrm{poly}(t)$ time. Further, the matrix can be computed in time $\tilde{O}(nt)$.*

Error tolerant monotone encodings. A (one-sided) *r-error-correcting (n, d)-monotone encoding* is a monotone injective mapping from subsets of size up to d of $[n]$ to t bits, such that we can recover the correct subset even when up to r bits are flipped from 0 to 1. The number t is called the *length* of the encoding. This type of one-sided error holds true for the read-once memory environment where monotone encoding is applicable [17,2].

Theorem 20. *Let $n \geq d$ be given positive integers. For each integer i, $0 \leq i \leq \log_2 d - 1$, let $\mathbf{A}_i$ be a $(d/2^i, d/2^{i+1}, r, 0)$-list-disjunct $t_i \times n$ matrix. From the matrices $\mathbf{A}_i$, we can construct a r-error-correcting (n, d)-monotone encoding with length $t = \sum_{i=0}^{\log_2 d - 1} t_i$ which can be encoded and decoded in time $O(nt)$. Furthermore, if the matrices $\mathbf{A}_i$ are strongly explicit then the monotone encoding is strongly explicit. If the list-disjunct matrices $\mathbf{A}_i$ can be decoded in time $T_i(n, d)$, then the monotone encoding can be computed in time $\sum_i T_i(n, d)$.*

Corollary 21. *There exist r-error-correcting (n, d)-monotone-encodings of length $t = O\left(d \log(n/d) + r \log d\right)$. Note that, this bound is best possible when $r = 0$ because the information theoretic bound is $\Omega(d \log(n/d))$.*

Finally, apply the list-disjunct matrices in Corollary 13 to Theorem 20, we obtain the following.

Corollary 22. *Given integers $n \geq d \geq 1$ and r, there exists a $\mathrm{poly}(t)$-time algorithm computing an r-error-correcting (n, d)-monotone-encoding with length $t = O\left((d \log n)^{1+o(1)} + r \log d \cdot \log \log n\right)$, which can be decoded in $\mathrm{poly}(t)$-time also.*

The following lower bound is off from the upper bound by a factor of about $\tilde{O}(\log d)$.

Proposition 23. *Suppose there exists a r-error-correcting (n, d)-monotone encoding, then the code length t has to satisfy $t = \Omega\left(d \log(n/d) + r \log\left(\frac{d \log(n/d)}{r}\right)\right)$.*

Efficiently Decodable Multiple User Tracing Family. The following results answer two open questions left in [1], concerning the explicit constructions of MUT families and the fast-decodability of such families. Furthermore, our results are error-correcting.

Theorem 24. *Given non-negative integers $e_0, e_1, u \leq d < n$, there is a randomized construction of $t \times n$ (d, u)-MUT matrix, which can correct e_0 false positives and e_1 false negatives, where $t = O((d + u^2) \log n + (e_0 + e_1)d)$. If we also want explicit construction along with efficient decoding, then t is slightly increased to $t = O((d^{1+o(1)} + u^2) \log n + (e_0 + e_1)d \log \log n)$.*

References

1. Alon, N., Asodi, V.: Tracing many users with almost no rate penalty. IEEE Trans. Inform. Theory 53(1), 437–439 (2007)
2. Alon, N., Hod, R.: Optimal monotone encodings. IEEE Transactions on Information Theory 55(3), 1343–1353 (2009)
3. Cheraghchi, M.: Noise-resilient group testing: Limitations and constructions. In: Kutyłowski, M., Charatonik, W., Gębala, M. (eds.) FCT 2009. LNCS, vol. 5699, pp. 62–73. Springer, Heidelberg (2009)

4. Cormode, G., Muthukrishnan, S.: What's hot and what's not: tracking most frequent items dynamically. ACM Trans. Database Syst. 30(1), 249–278 (2005)
5. De Bonis, A., Gąsieniec, L., Vaccaro, U.: Optimal two-stage algorithms for group testing problems. SIAM J. Comput. 34(5), 1253–1270 (electronic) (2005)
6. Du, D.Z., Hwang, F.K.: Combinatorial group testing and its applications, 2nd edn. Series on Applied Mathematics, vol. 12. World Scientific Publishing Co. Inc., River Edge (2000)
7. D'yachkov, A.G., Rykov, V.V.: A survey of superimposed code theory. Problems Control Inform. Theory/Problemy Upravlen. Teor. Inform. 12(4), 229–242 (1983)
8. D'yachkov, A.G., Rykov, V.V., Rashad, A.M.: Superimposed distance codes. Problems Control Inform. Theory 18(4), 237–250 (1989)
9. Erdős, P., Frankl, P., Füredi, Z.: Families of finite sets in which no set is covered by the union of r others. Israel J. Math. 51(1-2), 79–89 (1985)
10. Füredi, Z.: On r-cover-free families. J. Combin. Theory Ser. A 73(1), 172–173 (1996)
11. Ganguly, S.: Data stream algorithms via expander graphs. In: Hong, S.-H., Nagamochi, H., Fukunaga, T. (eds.) ISAAC 2008. LNCS, vol. 5369, pp. 52–63. Springer, Heidelberg (2008)
12. Goodrich, M.T., Atallah, M.J., Tamassia, R.: Indexing information for data forensics. In: Third International Conference on Applied Cryptography and Network Security (ANCS), pp. 206–221 (2005)
13. Indyk, P.: Explicit constructions of selectors and related combinatorial structures, with applications. In: SODA, pp. 697–704 (2002)
14. Indyk, P., Ngo, H.Q., Rudra, A.: Efficiently decodable non-adaptive group testing. In: Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 1126–1142 (2010)
15. Kainkaryam: Pooling in high-throughput drug screening. Current Opinion in Drug Discovery & Development 12(3), 339–350 (2009)
16. Khattab, S.M., Gobriel, S., Melhem, R.G., Mossé, D.: Live baiting for service-level dos attackers. In: INFOCOM, pp. 171–175 (2008)
17. Moran, T., Naor, M., Segev, G.: Deterministic history-independent strategies for storing information on write-once memories. Theory of Computing 5(1), 43–67 (2009)
18. Ngo, H.Q., Du, D.Z.: A survey on combinatorial group testing algorithms with applications to DNA library screening. In: Discrete Mathematical Problems with Medical Applications. DIMACS Ser. Discrete Math. Theoret. Comput. Sci., vol. 55, pp. 171–182. Amer. Math. Soc., Providence (2000)
19. Parvaresh, F., Vardy, A.: Correcting errors beyond the Guruswami-Sudan radius in polynomial time. In: Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS), pp. 285–294 (2005)
20. Porat, E., Rothschild, A.: Explicit non-adaptive combinatorial group testing schemes. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part I. LNCS, vol. 5125, pp. 748–759. Springer, Heidelberg (2008)
21. Rashad, A.M.: Random coding bounds on the rate for list-decoding superimposed codes. Problems Control Inform. Theory/Problemy Upravlen. Teor. Inform. 19(2), 141–149 (1990)
22. Rudra, A., Uurtamo, S.: Data stream algorithms for codeword testing. In: Abramsky, S., Gavoille, C., Kirchner, C., Meyer auf der Heide, F., Spirakis, P.G. (eds.) ICALP 2010. LNCS, vol. 6198, pp. 629–640. Springer, Heidelberg (2010)
23. Ruszinkó, M.: On the upper bound of the size of the r-cover-free families. J. Combin. Theory Ser. A 66(2), 302–310 (1994)

Robust Simulations and Significant Separations

Lance Fortnow[1,⋆] and Rahul Santhanam[2,⋆⋆]

[1] Northwestern University
[2] University of Edinburgh

Abstract. We define and study a new notion of "robust simulations" between complexity classes which is intermediate between the traditional notions of infinitely-often and almost-everywhere, as well as a corresponding notion of "significant separations". A language L has a robust simulation in a complexity class C if there is a language in C which agrees with L on arbitrarily large polynomial stretches of input lengths. There is a significant separation of L from C if there is no robust simulation of $L \in \mathsf{C}$.

The new notion of simulation is a cleaner and more natural notion of simulation than the infinitely-often notion. We show that various implications in complexity theory such as the collapse of PH if $\mathsf{NP} = \mathsf{P}$ and the Karp-Lipton theorem have analogues for robust simulations. We then use these results to prove that most known separations in complexity theory, such as hierarchy theorems, fixed polynomial circuit lower bounds, time-space tradeoffs, and the recent theorem of Williams, can be strengthened to significant separations, though in each case, an almost everywhere separation is unknown.

Proving our results requires several new ideas, including a completely different proof of the hierarchy theorem for non-deterministic polynomial time than the ones previously known.

1 Introduction

What does the statement "$\mathsf{P} \neq \mathsf{NP}$" really tell us? All is says is that for any polynomial-time algorithm A, A fails to solve SAT on an infinite number of inputs. These hard-to-solve inputs could be exponentially (or much worse) far from each other. Thus even a proof of $\mathsf{P} \neq \mathsf{NP}$ could leave open the possibility that SAT or any other NP-complete problem is still solvable on all inputs encountered in practice. This is unsatisfactory if we consider that one of the main motivations of proving lower bounds is to understand the limitations of algorithms.

Another important motivation for proving lower bounds is that hardness is *algorithmically useful* in the context of cryptography or derandomization. Again, if the hardness only holds for inputs or input lengths that are very far apart, this usefulness is called into question. For this reason, theorists have studied a notion

⋆ Supported in part by NSF grants CCF-0829754 and DMS-0652521.
⋆⋆ Supported in part by EPSRC grant H05068X/1.

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 569–580, 2011.

of almost-everywhere (a.e.) separations, and a corresponding notion of infinitely-often (i.o.) simulations. A language L is in $\mathsf{i.o.C}$ for a complexity class C if there is some $A \in \mathsf{C}$ such that A and L agree on infinitely many input lengths. A class D is almost everywhere not in C if for some language L in D, $L \notin \mathsf{i.o.C}$, that is any C-algorithm fails to solve L on all but a finite number of input lengths. As an example of applying these notions, Impagliazzo and Wigderson [IW97] show that if $\mathsf{E} \not\subseteq \mathsf{SIZE}(2^{o(n)})$ then BPP is in $\mathsf{i.o.P}$, and that if $\mathsf{E} \not\subseteq \mathsf{i.o.SIZE}(2^{o(n)})$, then $\mathsf{BPP} = \mathsf{P}$.

However, the infinitely often notion has its own issues. Ideally, we would like a notion of simulation to capture "easiness" in some non-trivial sense. Unfortunately, many problems that we consider hard have trivial infinitely often simulations. For example, consider any NP-hard problem on graphs or square matrices. The natural representation of inputs for such problems yields non-trivial instances only for input lengths that are perfect squares. In such a case, the problem has a trivial infinitely often simulation on the set of all input lengths which are not perfect squares. On the other hand, the problem could be "padded" so that it remains non-trivial on input lengths which are not perfect squares. It's rather unsatisfactory to have a notion of simulation which is so sensitive to the choice of input representation.

Not unrelated to this point is that analogues of many classical complexity results fail to hold in the infinitely often setting. For example, we do not know if $SAT \in \mathsf{i.o.P}$ implies that the entire Polynomial Hierarchy has simulations infinitely-often in polynomial time. Also it's not true in general that if a complete language for a class is easy infinitely-often, then the entire class is easy infinitely-often. This is true for SAT and NP because SAT is paddable and downward self-reducible, but it's unclear in which situations the implication holds. Given that even these basic analogues are not known, it's not surprising that more involved results such as the Karp-Lipton theorem [KL82] and the theorem of Impagliazzo, Kabanets and Wigderson [IKW02] that $\mathsf{NEXP} \subseteq \mathsf{SIZE}(poly)$ implies $\mathsf{NEXP} = \mathsf{MA}$ don't have known infinitely often analogues either.

In an ideal world, we would like all our algorithms to work on all input lengths, and all our separations to be almost-everywhere separations. While algorithm design does typically focus on algorithms that work on all input lengths, many of the complexity separations we know do not work in the almost everywhere setting. Separations proved using combinatorial or algebraic methods, such as Hastad's lower bound for Parity [Hås86] or Razborov's monotone circuit lower bound for Clique [Raz85] tend to be almost everywhere (in an appropriate input representation). However, such techniques typically have intrinsic limitations, as they run into the natural proofs barrier [RR97]. Many of the lower bounds proved recently have come from the use of indirect diagonalization. A contrary upper bound is assumed and this assumption is used together with various other ideas to derive a contradiction to a hierarchy theorem. These newer results include hierarchy theorems [Bar02, FS04, vMP06], time-space tradeoffs [For00, FLvMV05], and circuit lower bounds [BFT98, Vin05, San07, Wil10a, Wil10b]. Unfortunately, *none* of these results give almost everywhere separations, and so the question

immediately arises what we can say quantitatively about these separations, in terms of the frequency with which they hold.

To address all these issues, we describe a new notion of "robust simulation" and a corresponding notion of "significant separation". A language L is in $\mathsf{r.o.C}$ (robustly-often in C) if there is a language A in C such that for every k there are infinitely many m such that A and L agree on all input lengths between m and m^k. A class D has a significant separation from C if there is some L in D such that $L \notin \mathsf{r.o.C}$. This implies that for each $L' \in \mathsf{C}$, there is a constant k such that for each m, L and L' differ on at least one input length between m and m^k. Intuitively, this means that if the separation holds at some input length, there is another input length at most polynomially larger at which the separation also holds, i.e., the hardness is not too "sparsely" distributed.

Our definition of robust simulations extends the notion of uniform hardness of Downey and Fortnow [DF03]. A set A is uniformly hard in the sense of Downey and Fortnow if $A \notin \mathsf{r.o.P}$.

The notion of robust simulation is just slightly stronger than the notion of infinitely often simulation, and correspondingly the notion of significant separation is slightly weaker than that of almost everywhere separations. By making this tradeoff, however, we show that we can often achieve the best of both worlds.

We give robustly often analogues of many classical complexity results, where infinitely often analogues remain open, including

- $\mathsf{NP} \subseteq \mathsf{r.o.P}$ implies $\mathsf{PH} \subseteq \mathsf{r.o.P}$
- $\mathsf{NP} \subseteq \mathsf{r.o.SIZE}(poly)$ implies $\mathsf{PH} \subseteq \mathsf{r.o.SIZE}(poly)$
- $\mathsf{NEXP} \subseteq \mathsf{r.o.SIZE}(poly)$ implies $\mathsf{NEXP} \subseteq \mathsf{r.o.MA}$

We then use these robustly often analogues together with other ideas to give several significant separations where almost everywhere separations remain open, including

- $\mathsf{NTIME}(n^r) \not\subseteq \mathsf{r.o.NTIME}(n^s)$, when $r > s \geqslant 1$
- For each constant k, $\Sigma_2\mathsf{P} \not\subseteq \mathsf{r.o.SIZE}(n^k)$
- $SAT \not\subseteq \mathsf{r.o.DTISP}(n^\alpha, \mathsf{polylog}(n))$ when $\alpha < \sqrt{2}$
- $\mathsf{NEXP} \not\subseteq \mathsf{r.o.ACC}^0$

The robustly often notion gives us a cleaner and more powerful theory than the infinitely often notion.

1.1 Intuition and Techniques

To illustrate the advantages of the robustly often notion over the infinitely often notion, let's look at a simple example: trying to show that if SAT is easy, then all of NP is easy. Let $L \in \mathsf{NTIME}(n^k)$ be any NP language, where $k > 0$ is a constant. $\mathsf{SAT} \in \mathsf{i.o.P}$ doesn't immediately imply $L \in \mathsf{i.o.P}$, as the range of the reduction from L to SAT might only intersect input lengths where the polynomial-time algorithm for SAT is incorrect. In this case, the problem can be fixed by padding the reduced instance to polynomially many different input

lengths and using downward self-reducibility to check YES answers for any of these inputs. However, this fix depends on specific properties of SAT.

Showing that SAT $\in$ r.o.P implies $L \in$ r.o.P is an easier, more generic argument. Define a robust set of natural numbers to be any set S such that for each $k > 0$ there is an m for which S contains all numbers between m and m^k for some m. SAT $\in$ r.o.P means that there is a robust set S on which the simulation works. Now the reduction from L to SAT creates instances of length $n^k \mathsf{polylog}(n)$, and it's not hard to see that this automatically implies that composing the reduction with the algorithm for SAT gives an algorithm for L which works on some robust subset S' of S. This implies $L \in$ r.o.P. We call a robust subset of a set a *robust refinement*. Many of our arguments will involve defining a series of robust refinements of a robust set such that the desired simulation holds on the final refinement in the series, thus implying that the simulation goes through in the robustly often setting.

Thus far, using robustly often seems easier but hasn't given us any additional power. The situation changes if we consider implications where the assumption is used *two or more* times. An example is the proof that NP $\subseteq$ P implies PH $\subseteq$ P - this is an inductive proof where the assumption is used several times. Trying to carry an infinitely often simulation through fails miserably in this setting because two infinitely often simulations do not compose - they might work on two completely different infinite sets of input lengths.

Now two robustly often simulations do not in general compose either. It is not in general true that for complexity classes B, C and D, if B $\subseteq$ r.o.C and $C \subseteq$ r.o.D, then B $\subseteq$ r.o.D. However, we *can* get these two robustly often simulations to compose when they are *both* consequences of a single robustly often assumption. The robustly often assumption gives us some robust set to work with. If we're careful we can define a single robust refinement of this set on which B $\subseteq$ C holds and so too does C $\subseteq$ D, which implies B $\subseteq$ D holds on this refinement as well.

This is an idea that we will use again and again in our proofs. However, in order to use this idea, we need to be careful with the steps in our proof, as there are only some kinds of implications for which the idea is useful. For example, it works well with fixed polynomial-time reductions or translations with fixed polynomial advice, but not with exponential padding. More importantly, the idea only works when all the steps in the proof follow from a *single* assumption, so we need to re-formulate proofs so that they conform to this pattern. In some cases, eg. the proofs of Theorem 4 and Theorem 6, the re-formulation is non-trivial and leads to proofs that are quite a bit more involved than the originals [IKW02, Kan82].

In the case of hierarchies for non-deterministic time where the lower bound is against robust simulations, the known techniques break down entirely. The traditional argument is a "chaining argument" [Coo72, SFM78, Ž83] which uses a chain of exponentially many input lengths and cannot possibly give a hierarchy against robustly often simulations. Here, we come up with a novel idea of chaining using witnesses to get such a hierarchy for polynomial time.

Our most technically involved result is that the recent breakthrough lower bound of Williams [Wil10a, Wil10b] can be strengthened to a significant separation. The proof of this result uses almost all of the techniques we develop, including the sophisticated use of robust refinements involved in proving Theorem 4, and a variant of the significant hierarchy for non-deterministic polynomial time.

Implicit in our paper is a certain proof system for proving complexity class separations, such that any separation proved in this system automatically yields a significant separation. It's an interesting open problem to make this system explicit, and to study its power and its limitations more formally.

2 Preliminaries

2.1 Complexity Classes, Promise Problems and Advice

We assume a basic familiarity with complexity classes such as P, RP, BPP, NP, MA, AM, Σ_2^p,PP and their exponential-time versions.The Complexity Zoo[1] is an excellent resource for basic definitions and statements of results.

Given a complexity class C, coC is the class of languages L such that $\bar{L} \in \mathsf{C}$. Given a function $s : \mathbb{N} \to \mathbb{N}$, SIZE($s$) is the class of Boolean functions $f = \{f_n\}$ such that for each n, f_n has Boolean circuits of size $O(s(n))$. Given a language L and an integer n, $L_n = L \cap \{0,1\}^n$. Given a class C, i.o.C is the class of languages L for which there is a language $L' \in \mathsf{C}$ such that $L_n = L'_n$ for infinitely many length n.

In order to deal with promise classes in a general way, we take as fundamental the notion of a complexity measure. A complexity measure CTIME is a mapping which assigns to each pair (M, x), where M is a time-bounded machine (here a time function $t_M(x)$ is implicit) and x an input, one of three values "0" (accept), "1" (reject) and "?" (failure of CTIME promise). We distinguish between *syntactic* and *semantic* complexity measures. Syntactic measures have as their range $\{0,1\}$ while semantic measures may map some machine-input pairs to "?". The complexity measures DTIME and NTIME are syntactic (each halting deterministic or non-deterministic machine either accepts or rejects on each input), while complexity measures such as BPTIME and MATIME are semantic (a probabilistic machine may accept on an input with probability 1/2, thus failing the bounded-error promise). For syntactic measures, any halting machine defines a language, while for semantic measures, only a subset of halting machines define languages.

A promise problem is a pair (Y, N), where $Y, N \subseteq \{0,1\}^*$ and $Y \cap N = \varnothing$. We say that a promise problem (Y, N) belongs to a class CTIME(t) if there is a machine M halting in time t on all inputs of length n such that M fulfils the CTIME promise on inputs in $Y \cup N$, accepting on inputs in Y and rejecting on inputs in N.

[1] `http://qwiki.caltech.edu/wiki/ComplexityZoo`

A language L is in $\mathsf{CTIME}(t)/a$ if there is a machine M halting in time $t(\cdot)$ taking an auxiliary *advice* string of length $a(\cdot)$ such that for each n, there is some advice string $b_n, |b_n| = a(n)$ such that M fulfils the CTIME promise for each input x with advice string b_n and accepts x iff $x \in L$. Note that this is a weaker requirement than in the original Karp-Lipton notion where the promise must be satisfied on all advice strings.

For syntactic classes, a lower bound for the class with small advice or for the promise version of the class translates to a lower bound for the class itself. For eg., if there is a promise problem in P which doesn't have polynomial-size circuits, then $\mathsf{P} \not\subseteq \mathsf{SIZE}(poly)$ and similarly, if $\mathsf{P}/O(n) \not\subseteq \mathsf{SIZE}(poly)$, then $\mathsf{P} \not\subseteq \mathsf{SIZE}(poly)$.

Definition 1. *Let S be a subset of positive integers. S is robust if for each positive integer k, there is a positive integer $m \geqslant 2$ such that $n \in S$ for all $m \leqslant n \leqslant m^k$.*

Note that any robust set is infinite. We now define what it means to simulate a language in a complexity class on a subset of the positive integers.

Definition 2. *Let L be a language, C a complexity class, and S a subset of the positive integers. We say $L \in \mathsf{C}$ on S if there is a language $L' \in \mathsf{C}$ such that $L_n = L'_n$ for any $n \in S$.*

Using the terminology of Definition 2, $L \in \mathsf{i.o.C}$ for a language L and complexity class C if there is some infinite set $S \subseteq \mathbb{N}$ such that $L \in \mathsf{C}$ on S. We now define our main notion of robustly-often simulations.

Definition 3. *Given a language L and complexity class C, $L \in \mathsf{r.o.C}$ if there is a robust S such that $L \in C$ on S. In such a case, we say that there is a robustly-often ($\mathsf{r.o.}$) simulation of L in C. We extend this notion to complexity classes in the obvious way - given complexity classes B and C, $\mathsf{B} \subseteq \mathsf{r.o.C}$ if there for each language $L \in \mathsf{B}$, $L \in \mathsf{r.o.C}$. If $\mathsf{B} \not\subseteq \mathsf{r.o.C}$, we say that there is a significant separation of B from C.*

Clearly $\mathsf{B} \subseteq \mathsf{r.o.C}$ implies $\mathsf{B} \subseteq \mathsf{i.o.C}$. Conversely, $\mathsf{B} \not\subseteq \mathsf{i.o.C}$ gives a very strong separation of B and C, i.e., an almost-everywhere separation, while a significant separation is somewhat weaker but still much more significant than simply a separation of B and C. Intuitively, a significant separation means that input lengths witnessing the separation are at most polynomially far apart.

We now define a sequence of *canonical refinements* for any given set S, which will play an important part in many of our proofs.

Definition 4. *Let S be a robust set. The canonical refinement S_d of S at level d is defined as follows for any integer $d > 0$: $m \in S_d$ iff $m \in S$ and $n \in S$ for all $m \leqslant n \leqslant m^d$.*

It is easy to see S_d is robust if S is robust and that $S_d \subseteq S_{d'}$ for $d \geqslant d'$.

Due to space constraints, we omit most proofs in this version of the paper.

3 Robust Simulations

For any NP-complete language L the language

$$L' = \{x10^i \mid x \in L, |x| + 1 + i \text{ is even}\}$$

remains NP-complete but sits in i.o.P. In contrast if any NP-complete set under honest m-reductions sits in r.o.P then NP $\subseteq$ r.o.P.

Lemma 1. *Let L and L' be languages such that L' reduces to L via an honest polynomial-time m-reduction. Let* C *be a complexity class closed under poly-time m-reductions. If there is a robust S such that $L \in$ C on S, then there is a robust refinement S' of S such that $L' \in$ C on S'.*

The proof ideas of Lemma 1 can be used to show that robustly often analogues of various useful implications hold. The first analogue essentially says that we can take a complete language to be representative of a complexity class, even in the context of robustly often simulations. It is an immediate consequence of Lemma 1.

Proposition 1. *If* SAT $\in$ r.o.P*, then* NP $\subseteq$ r.o.P*.*

The next proposition says that translation arguments using a fixed polynomial amount of padding carry through in the robustly often setting, for any "reasonable" complexity measure.

Proposition 2. *Let* BTIME *and* CTIME *be any complexity measures closed under efficient deterministic transductions. Let g and h be time-constructable functions, and p a polynomial. If* BTIME$(g(n)) \subseteq$ r.o.CTIME$(h(n))$*, then* BTIME$(g(p(n))) \subseteq$ r.o.CTIME$(h(p(n)))$*.*

As a consequence of Proposition 2, we get for example that if NTIME$(n) \subseteq$ r.o.P, then NP $\subseteq$ r.o.P.

The proposition below says that simulations of a syntactic class in another class can be translated to a simulation with fixed polynomial advice, even in the robustly often setting.

Proposition 3. *Let* BTIME *be a syntactic complexity measure and* CTIME *a complexity measure, such that both* BTIME *and* CTIME *are closed under efficient deterministic transductions. Let f and g be time-constructable measures and p a polynomial. If* BTIME$(f(n)) \subseteq$ r.o.CTIME$(g(n))$*, then* BTIME$(f(n))/p(n) \subseteq$ r.o.CTIME$(g(n + p(n)))/p(n)$*.*

Theorem 1. *If* NP $\subseteq$ r.o.P*, then* PH $\subseteq$ r.o.P

The proof is by induction, where the inductive hypothesis states that the k'th level of PH is contained in a suitably chosen refinement of the robust set on which the original polynomial-time simulation of SAT works.

Theorem 2. *If* NP $\subseteq$ r.o.SIZE$(poly)$*, then* PH $\subseteq$ r.o.SIZE$(poly)$

Theorem 3. *If* $\mathsf{NP} \subseteq \mathsf{r.o.BPP}$*, then* $\mathsf{PH} \subseteq \mathsf{r.o.BPP}$.

We omit the proofs of Theorems 2 and 3, which closely resemble the proof of Theorem 1.

Next we state a robust analogue of the Karp-Lipton theorem [KL82]. We formulate a stronger statement which will be useful when we show significant fixed-polynomial circuit size lower bounds for Σ_2^p.

Lemma 2. *If there is a constant* k *and a robust set* S *such that* $\mathsf{SAT} \in \mathsf{SIZE}(n^k)$ *on* S*, then there is a robust refinement* S' *of* S *such that* $\Pi_2 SAT \in \Sigma_2 - \mathsf{TIME}(n^{k+1+o(1)})$ *on* S'.

The following is an immediate corollary.

Corollary 1. *If* $\mathsf{NP} \subseteq \mathsf{r.o.SIZE}(poly)$*, then* $\Sigma_2^p \subseteq \mathsf{r.o.}\Pi_2^p$.

The following can be shown using the easy witness method of Kabanets [Kab01, IKW02] and known results on pseudo-random generators [NW94, KvM99].

Lemma 3. *Let* R *be any robust set and let* $k > 1$ *be any constant. Then there is a robust refinement* R' *of* R *such that either* $\mathsf{NE} \subseteq \mathsf{DTIME}(2^{n^{16k^4}})$ *on* R *or* $\mathsf{MATIME}(n^{4k^2}) \subseteq \mathsf{NE}/O(n)$ *on* R'.

Lemma 3 can be used to prove the following robustly-often analogue of the main theorem of Impagliazzo, Kabanets and Wigderson [IKW02].

Theorem 4. $\mathsf{NEXP} \subseteq \mathsf{r.o.SIZE}(poly)$ *iff* $\mathsf{NEXP} \subseteq \mathsf{r.o.MA}$.

4 Significant Separations

4.1 Hierarchies

The proofs of the hierarchies for deterministic time and space actually give almost-everywhere separations and therefore significant separations.

For nondeterministic time the situation is quite different. Cook [Coo72] showed that $\mathsf{NTIME}(n^r) \subsetneq \mathsf{NTIME}(n^s)$ for any reals $r < s$. Seiferas, Fischer and Meyer [SFM78] generalize this result to show that $\mathsf{NTIME}(t_1(n)) \subsetneq \mathsf{NTIME}(t_2(n))$ for $t_1(n+1) = o(t_2(n))$. Zak [Ž83] gives a simpler proof of the same result. All these proofs require building an exponential (or worse) chain of equalities to get a contradiction. Their proofs do not give almost everywhere separations or significant separations. No relativizable proof can give an i.o. hierarchy as Buhrman, Fortnow and Santhanam give a relativized world that $\mathsf{NEXP} \subseteq \mathsf{i.o.NP}$.

In this section we give a relativizing proof that $\mathsf{NTIME}(n^r) \not\subseteq \mathsf{r.o.NTIME}(n^s)$ for $r > s \geqslant 1$. This also gives a new proof of the traditional nondeterministic time hierarchy.

Theorem 5. *If* t_1 *and* t_2 *are time-constructable functions such that*

– $t_1(n) = o(t_2(n))$, *and*
– $n \leqslant t_1(n) \leqslant n^c$ *for some constant* c

then $\mathsf{NTIME}(t_2(n)) \not\subseteq \mathsf{r.o.NTIME}(t_1(n))$.

Corollary 2. *For any reals* $1 \leqslant r < s$, $\mathsf{NTIME}(n^s) \not\subseteq \mathsf{r.o.NTIME}(n^r)$.

Proof (Proof of Theorem 5). Let $M_1, M_2, \ldots$ be an enumeration of multitape nondeterministic machines that run in time $t_1(n)$.

Define a nondeterministic Turing machine M that on input 1^i01^m0w does as follows:

– If $|w| < t_1(i+m+2)$ accept if both $M_i(1^i01^m0w0)$ and $M_i(1^i01^m0w1)$ accept.
– If $|w| \geqslant t_1(i+m+2)$ accept if $M_i(1^i01^m0)$ rejects on the path specified by the bits of w.

Since we can universally simulate $t(n)$-time nondeterministic multitape Turing machines on an $O(t(n))$-time 2-tape nondeterministic Turing machine, $L(M) \in \mathsf{NTIME}(O(t_1(n+1))) \subseteq \mathsf{NTIME}(t_2(n))$. Note $(n+1)^c = O(n^c)$ for any c.

Suppose $\mathsf{NTIME}(t_2(n)) \subseteq \mathsf{r.o.NTIME}(t_1(n))$. Pick a c such that $t_1(n) \ll n^c$. By the definition of r.o. there is some n_0 and a language $L \in \mathsf{NTIME}(t_1(n))$ such that $L(M) = L$ on all inputs of length between n_0 and n_0^c. Fix i such that $L = L(M_i)$. Then $z \in L(M_i) \Leftrightarrow z \in L(M)$ for all $z = 1^i01^{n_0}0w$ for $w \leqslant t_1(i+n_0+2)$.

By induction we have $M_i(1^i01^{n_0}0)$ accepts if $M_i(1^i01^{n_0}0w)$ accepts for all $w \leqslant t_1(i+n_0+2)$. So $M_i(1^i01^{n_0}0)$ accepts if and only $M_i(1^i01^{n_0}0)$ rejects on every computation path, contradicting the definition of nondeterministic time.

4.2 Circuit Lower Bounds

We first state a stronger version of Kannan's [Kan82] lower bound for Σ_2^p against fixed polynomial size, where the separation is significant.

Theorem 6. *For each integer* $k > 1$, $\Sigma_2^p \not\subseteq \mathsf{r.o.SIZE}(n^k)$.

Using similar ideas we can get robustly often analogues of the lower bound of Cai and Sengupta [Cai01] for $\mathsf{S_2P}$ and Vinodchandran [Vin05] for PP:

Theorem 7. *For any* $k > 0$, $\mathsf{S_2P} \not\subseteq \mathsf{r.o.SIZE}(n^k)$.

Theorem 8. *For any* $k > 0$, $\mathsf{PP} \not\subseteq \mathsf{r.o.SIZE}(n^k)$.

Our most technically involved result is that the recent lower bound of Williams [Wil10b] that $\mathsf{NEXP} \not\subseteq \mathsf{ACC}^0$ extends to the robustly often setting. His proof uses the nondeterministic time hierarchy and the proof of Impagliazzo, Kabanets and Wigderson [IKW02], neither of which may hold in the infinitely-often setting. So to get a robustly-often result we require variants of our Theorems 5 and 4. To save space, we will focus on the new ingredients, and abstract out what we need from Williams' paper.

We first need the following simultaneous resource-bounded complexity class.

Definition 5. $\mathsf{NTIMEGUESS}(T(n), g(n))$ *is the class of languages accepted by NTMs running in time* $O(T(n))$ *and using at most* $O(g(n))$ *non-deterministic bits.*

We have the following variant of Theorem 5, which has a similar proof.

Lemma 4. *For any constant* k, $\mathsf{NTIME}(2^n) \not\subseteq \mathsf{r.o.NTIMEGUESS}(2^n/n, n^k)$.

We also need the following robustly often analogue of a theorem of Williams [Wil10a], which uses the proof idea of Theorem 4. The problem SUCCINCT3SAT is complete for NEXP under polynomial-time m-reductions.

Lemma 5. *If* $\mathsf{NE} \subseteq \mathsf{r.o.ACC}^0$ *on* S *for some robust set* S*, then there is a constant* c *and a refinement* S' *of* S *such that* SUCCINCT3SAT *has succinct satisfying assignments that are* ACC^0 *circuits of size* n^c *on* S'.

Proof. The proof of Theorem 4 gives that if $\mathsf{NE} \subseteq \mathsf{ACC}^0$ on S, then there is a constant d and a robust refinement R of S such that SUCCINCT3SAT has succinct satisfying assignments that are circuits of size n^d on R. Since $\mathsf{P} \subseteq \mathsf{ACC}^0$ on S and using Proposition 3, we get that there is a constant c and a robust refinement S' of R such that SUCCINCT3SAT has succinct satisfying assignments that are ACC^0 circuits of size n^c on S'.

Now we are ready to prove the robustly often analogue of Williams' main result [Wil10b].

Theorem 9. $\mathsf{NEXP} \not\subseteq \mathsf{r.o.ACC}^0$.

Proof Sketch. Assume, to the contrary, that $\mathsf{SUCCINCT3SAT} \in \mathsf{ACC}^0$ on R for some robust R. By completeness of SUCCINCT3SAT, it follows that there is a robust refinement S of R and a constant $k' > 1$ such that NE has ACC^0 circuits of size $n^{k'}$. Let $L \in \mathsf{NTIME}(2^n)$ but not in $\mathsf{r.o.NTIMEGUESS}(2^n/n, n^k)$, where k will be chosen large enough as a function of k'. Existence of L is guaranteed by Lemma 4. We will show $L \in \mathsf{r.o.NTIMEGUESS}(2^n/n, n^k)$ and obtain a contradiction.

The proof of Theorem 3.2 in Williams' paper gives an algorithm for determining if $x \in L$. The algorithm non-deterministically guesses and verifies a "small" (size $n^{O(c)}$) ACC^0 circuit which is equivalent to the SUCCINCT3SAT instance to which x reduces, within time $2^n/\omega(n)$ by using Williams' new algorithm for ACC^0-SAT together with the assumption that NEXP and hence P in ACC^0 on S. This guess-and-verification procedure works correctly on some robust refinement of S. Then, the algorithm uses the existence guarantee of Lemma 5 to guess and verify a succinct witness, again using Williams' algorithm for ACC^0-SAT. This further guess-and-verification procedure works correctly on some further robust refinement S'' of S. In total, the algorithm uses at most $n^{dk'}$ non-deterministic bits for some constant d, runs in time at most $2^n/n$ and decides L correctly on S''. By choosing $k > dk'$, we get the desired contradiction. □

Williams' work still leaves open whether $\mathsf{NEXP} \subseteq \mathsf{SIZE}(poly)$. Using the same ideas as in the proof of Theorem 9, we can show that an algorithm for CircuitSAT that improves slightly on brute force search robustly often would suffice to get such a separation.

Theorem 10. *If for each polynomial p,* CircuitSAT *can be solved in time $2^{n-\omega(\log(n))}$ robustly often on instances where the circuit size is at most $p(n)$, then* $\mathsf{NEXP} \not\subseteq \mathsf{SIZE}(poly)$.

4.3 Time-Space Tradeoffs

Proposition 4. *Let t and T be time-constructible functions such that $t = o(T)$. Then* $\mathsf{NTIME}(T) \not\subseteq \mathsf{i.o.coNTIME}(t)$.

Proposition 4 can be used to show the following r.o.analogue of a time-space tradeoff for SAT [FLvMV05].

Theorem 11. *Let $\alpha < \sqrt{2}$ be any constant.* $\mathsf{SAT} \notin \mathsf{r.o.DTISP}(n^{\alpha}, \mathsf{polylog}(n))$.

References

[Bar02] Barak, B.: A probabilistic-time hierarchy theorem for "Slightly Non-uniform" algorithms. In: Rolim, J.D.P., Vadhan, S.P. (eds.) RANDOM 2002. LNCS, vol. 2483, pp. 194–208. Springer, Heidelberg (2002)

[BFL91] Babai, L., Fortnow, L., Lund, C.: Non-deterministic exponential time has two-prover interactive protocols. Computational Complexity 1, 3–40 (1991)

[BFS09] Buhrman, H., Fortnow, L., Santhanam, R.: Unconditional lower bounds against advice. In: Albers, S., Marchetti-Spaccamela, A., Matias, Y., Nikoletseas, S., Thomas, W. (eds.) ICALP 2009. LNCS, vol. 5555, pp. 195–209. Springer, Heidelberg (2009)

[BFT98] Buhrman, H., Fortnow, L., Thierauf, T.: Nonrelativizing separations. In: Proceedings of 13th Annual IEEE Conference on Computational Complexity, pp. 8–12 (1998)

[Cai01] Cai, J.-Y.: $S_2^P \subseteq ZPP^{NP}$. In: Proceedings of the 42nd Annual Symposium on Foundations of Computer Science, pp. 620–629 (2001)

[Coo72] Cook, S.: A hierarchy for nondeterministic time complexity. In: Fourth Annual ACM Symposium on Theory of Computing, Conference Record, Denver, Colorado, May 1-3, pp. 187–192 (1972)

[Coo88] Cook, S.: Short propositional formulas represent nondeterministic computations. Informations Processing Letters 26(5), 269–270 (1988)

[DF03] Downey, R., Fortnow, L.: Uniformly hard languages. Theoretical Computer Science 298(2), 303–315 (2003)

[FLvMV05] Fortnow, L., Lipton, R., van Melkebeek, D., Viglas, A.: Time-space lower bounds for satisfiability. Journal of the ACM 52(6), 833–865 (2005)

[For00] Fortnow, L.: Time-space tradeoffs for satisfiability. Journal of Computer and System Sciences 60(2), 337–353 (2000)

[FS04] Fortnow, L., Santhanam, R.: Hierarchy theorems for probabilistic polynomial time. In: Proceedings of the 45th IEEE Symposium on Foundations of Computer Science, pp. 316–324 (2004)

[Hås86] Håstad, J.: Almost optimal lower bounds for small depth circuits. In: Proceedings of the 18th Annual ACM Symposium on Theory of Computing, pp. 6–20 (1986)

[IKW02] Impagliazzo, R., Kabanets, V., Wigderson, A.: In search of an easy witness: Exponential time vs. probabilistic polynomial time. Journal of Computer and System Sciences 65(4), 672–694 (2002)

[IW97] Impagliazzo, R., Wigderson, A.: P = BPP if E requires exponential circuits: Derandomizing the XOR lemma. In: Proceedings of the 29th Annual ACM Symposium on the Theory of Computing, pp. 220–229 (1997)

[Kab01] Kabanets, V.: Easiness assumptions and hardness tests: Trading time for zero error. Journal of Computer and System Sciences 63(2), 236–252 (2001)

[Kan82] Kannan, R.: Circuit-size lower bounds and non-reducibility to sparse sets. Information and Control 55(1), 40–56 (1982)

[KL82] Karp, R., Lipton, R.: Turing machines that take advice. L'Enseignement Mathématique 28(2), 191–209 (1982)

[KvM99] Klivans, A., van Melkebeek, D.: Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses. In: Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing, pp. 659–667 (1999)

[NW94] Nisan, N., Wigderson, A.: Hardness vs randomness. Journal of Computer and System Sciences 49(2), 149–167 (1994)

[Raz85] Razborov, A.: Lower bounds for the monotone complexity of some boolean functions. Soviet Mathematics Doklady 31, 354–357 (1985)

[RR97] Razborov, A., Rudich, S.: Natural proofs. Journal of Computer and System Sciences 55(1), 24–35 (1997)

[San07] Santhanam, R.: Circuit lower bounds for Merlin-Arthur classes. In: Proceedings of 39th Annual Symposium on Theory of Computing, pp. 275–283 (2007)

[SFM78] Seiferas, J., Fischer, M., Meyer, A.: Separating nondeterministic time complexity classes. Journal of the ACM 25(1), 146–167 (1978)

[Vin05] Vinodchandran, V.: A note on the circuit complexity of PP. Theoretical Computer Science 347(1-2), 415–418 (2005)

[vMP06] van Melkebeek, D., Pervyshev, K.: A generic time hierarchy for semantic models with one bit of advice. In: Proceedings of 21st Annual IEEE Conference on Computational Complexity, pp. 129–144 (2006)

[Wil10a] Williams, R.: Improving exhaustive search implies superpolynomial lower bounds. In: Proceedings of the 42nd Annual ACM Symposium on Theory of Computing, pp. 231–240 (2010)

[Wil10b] Williams, R.: Non-uniform ACC circuit lower bounds (2010) (manuscript)

[Žák83] Žák, S.: A Turing machine time hierarchy. Theoretical Computer Science 26(3), 327–333 (1983)

A PCP Characterization of AM

Andrew Drucker*

Computer Science and Artificial Intelligence Laboratory
Massachusetts Institute of Technology
Cambridge, MA 02139, USA
adrucker@mit.edu
http://www.springer.com/lncs

Abstract. We introduce a 2-round stochastic constraint-satisfaction problem, and show that its approximation version is complete for (the promise version of) the complexity class AM. This gives a "PCP characterization" of AM analogous to the PCP Theorem for NP. Similar characterizations have been given for higher levels of the Polynomial Hierarchy, and for PSPACE; however, we suggest that the result for AM might be of particular significance for attempts to derandomize this class.

To test this notion, we pose some hypotheses related to our stochastic CSPs that (in light of our result) would imply collapse results for AM. Unfortunately, the hypotheses may be over-strong, and we present evidence against them. In the process we show that, if some language in NP is hard-on-average against circuits of size $2^{\Omega(n)}$, then there exist "inapproximable-on-average" optimization problems of a particularly elegant form.

All our proofs use a powerful form of PCPs known as Probabilistically Checkable Proofs of Proximity, and demonstrate their versatility. We also use known results on randomness-efficient soundness- and hardness-amplification. In particular, we make essential use of the Impagliazzo-Wigderson generator; our analysis relies on a recent Chernoff-type theorem for expander walks.

Keywords: Arthur-Merlin games, PCPs, average-case complexity.

1 Introduction

1.1 CSPs, PCPs, and Complexity Classes

A *Constraint Satisfaction Problem (CSP)* is a collection of Boolean-valued constraints over variables from a finite alphabet Σ. A CSP in which each constraint depends on at most k variables is called a *k-CSP*. A natural computational task is to determine the maximum fraction of constraints that can be satisfied by any assignment. The landmark PCP Theorem of [1] asserts that, for a sufficiently

* Supported by a DARPA YFA grant. Supported during part of this work by an Akamai Presidential Graduate Fellowship. This research formed part of the author's Master's thesis [8].

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 581–592, 2011.

small constant $\varepsilon > 0$, it is NP-hard to distinguish between 3-CSPs in which one can satisfy all constraints, and those for which every assignment violates an ε fraction of the constraints (here one can take $\Sigma = \{0, 1\}$).

Choosing an assignment to a CSP can be viewed as a game of *solitaire*; the PCP Theorem implies that this game's value is NP-hard to approximate. Now, it is equally possible to study games played on a CSP in which 2 players *alternate* in setting values to designated blocks of variables, with one player trying to maximize the fraction of satisfied clauses and the other trying to minimize this fraction. Inspired by and building on the PCP Theorem, several works explored the complexity of approximating the value of such games. Ko and Lin [15] showed that approximating the value of such a game is hard for the j^{th} level of the Polynomial Hierarchy, if the game lasts for j moves. In more recent work of Haviv, Regev, and Ta-Shma [9] this result was shown to hold even if each variable is allowed to appear in at most a constant number of constraints. If the game is allowed to last polynomially many rounds, the approximation problem becomes PSPACE-hard, as shown by Condon, Feigenbaum, Lund, and Shor [4]. The same authors showed in [5] that the approximation problem for poly(n) rounds is also PSPACE-hard if a maximizing player plays against a *random* player, where the game's value is now defined as the *expected* fraction of satisfied clauses under optimum play by the maximizer. Moreover, all of the hardness-of-approximation results mentioned so far are in fact completeness results for the corresponding promise classes, so they can be viewed as giving "PCP characterizations" of NP, PSPACE, and the Polynomial Hierarchy.

One class that did not receive a PCP characterization based on CSP games was the "Arthur-Merlin" class AM. In fact, there are few known natural complete problems for AM (technically, for its promise version, prAM; it is unknown if AM, a semantic class, has *any* complete problems. See Sec. 2.1 for the definition of prAM.). To our knowledge, there is only one *approximation* problem previously known to be prAM-complete: Mossel and Umans [16] give a prAM-completeness result for approximating the VC dimension of set systems. This striking result does not fall within the framework of CSP games given above.

1.2 Our Results

Our first main result is a new PCP characterization of prAM. Following [5], we consider "stochastic" 2-CSPs $\psi(r, z)$, where r is a collection of Boolean variables and z a collection of variables over an alphabet Σ. Let $\mathrm{Val}_\psi(r, z)$ be the fraction of constraints of ψ satisfied by (r, z). We prove:

Theorem 1. *There is a finite alphabet Σ, a constant $\varepsilon > 0$, and a function $p(\ell) \leq 2^{-\Omega(\ell)}$, such that it is* prAM*-complete to distinguish between the following two sets of* 2*-CSPs:*

$$\Pi_{AM-CSP,YES} = \{\psi : \textit{ for all } r \textit{ there exists } z \textit{ such that } \mathrm{Val}_\psi(r, z) = 1\};$$

$$\Pi_{AM-CSP,NO} = \{\psi : \textit{ with prob. } \geq 1 - p(|r|) \textit{ over } r,\ \mathrm{Max}_z[\mathrm{Val}_\psi(r, z)] < 1 - \varepsilon\}.$$

AM is a class for which we feel such a PCP characterization might be especially important. There is compelling evidence that AM = NP, or at least that significant derandomization of AM is possible (see [18] for an overview of this line of research). One approach to try and derandomize AM is to directly attack the "easiest" AM-hard problems—those whose membership in AM is trivial to show, but whose AM-hardness is not. A problem like the one provided by Theorem 1 seems like a plausible candidate.

With this hope in mind, we initiate the study of a second, closely related computational problem for stochastic 2-round CSPs, the *randomized CSP optimization problem*. In this problem, we are given a CSP $\psi(r, z)$ along with a uniformly chosen setting to r. Subject to this setting, we wish to find a value z such that $\mathrm{Val}_\psi(r, z)$ is very nearly maximized. We are interested in efficient algorithms that perform this task well with high probability over r.

Several variations of the randomized CSP optimization problem can be explored, but one thing is clear: in any positive general solution to the problem, we must allow our optimizer to depend *nonuniformly* on the 2-CSP ψ. Such nonuniformity is necessary provided P ≠ NP, in light of the PCP Theorem for NP. This leads us to consider the following "Randomized Optimization Hypothesis", which posits that nonuniformity suffices to solve the randomized CSP optimization problem. In the statement below, $\psi(r, z)$ is a 2-CSP over ℓ Boolean variables (r) and m variables (z) over a finite alphabet Σ.

Hypothesis 1. *(Randomized Optimization Hypothesis for* P/poly*) Fix any* $\delta > 0$*. For every 2-CSP* $\psi(r, z)$*, there exists a circuit* $C_\psi(r) : \{0,1\}^\ell \to \Sigma^m$ *of size* $O(\mathrm{poly}(|\psi|))$*, such that with probability at least* $1/\mathrm{poly}(\ell)$ *over a random* $r \in \{0,1\}^\ell$*, we have* $\mathrm{Val}_\psi(r, C_\psi(r)) \geq \mathrm{Max}_z[\mathrm{Val}_\psi(r, z)] - \delta$.

Using Theorem 1, it is easy to show that any proof of this hypothesis would yield a collapse result for AM:

Claim 2. *Hypothesis 1 implies* AM = MA.

A strengthened hypothesis gives the stronger implication that AM = NP:

Hypothesis 2. *(Randomized Optimization Hypothesis for* NC^0*) For any* $\delta > 0$*, there is an integer* $t = t(\delta) > 0$ *such that the following holds. For every 2-CSP* $\psi(r, z)$*, there exists a function* $F_\psi(r) : \{0,1\}^\ell \to \Sigma^m$*, where each output coordinate of* F_ψ *depends on at most* t *bits of* r*, and such that with probability at least* $1 - \delta$ *over* r*,* $\mathrm{Val}_\psi(r, F_\psi(r)) \geq \mathrm{Max}_z[\mathrm{Val}_\psi(r, z)] - \delta$.

Claim 3. *Hypothesis 2 implies* AM = NP.

The proofs of both claims are in the full version. Given the potential consequences of our hypotheses, what chance do they have of being true? Unfortunately, it seems that each is unlikely. We prove two results to the effect that, if NP decision problems are hard on average for exponential-size circuits, then both hypotheses fail in a strong way. We state these results next. A language L is $p(n)$*-hard for size* $s(n)$ if for every circuit C of size $s(n)$, $\Pr_{x \in \{0,1\}^n}[C(x) = L(x)] \leq p(n)$. We show:

Theorem 4. *Suppose there exists a $\gamma_1 > 0$ and an $L \in \mathsf{NP} \cap \mathsf{coNP}$ that is $(1 - 1/\mathrm{poly}(n))$-hard for size $2^{\gamma_1 n}$. Then there exist constants $c, \gamma_2, \theta > 0$, and a polynomial-time constructible family $\{\psi_n(r, z)\}_{n>0}$ of 2-CSPs (with $|r| = cn, |z| = d(n) = O(\mathrm{poly}(n))$), such that for all n:*

(4.i) For all r, $\exists z$ such that $\mathrm{Val}_{\psi_n}(r, z) = 1$;
(4.ii) If $C : \{0,1\}^{cn} \to \{0,1\}^{d(n)}$ is any circuit of size at most $2^{\gamma_2 n}$, then

$$\Pr_r[\mathrm{Val}_{\psi_n}(r, C(r)) > 1 - \theta] \leq 2^{-\Omega(n)} .$$

Theorem 5. *There is an $\varepsilon_0 > 0$ such that the following holds. Suppose there exists a $\gamma_1 > 0$ and an $L \in \mathsf{NP}$ that is $(1/2 + \varepsilon_0)$-hard for size $2^{\gamma_1 n}$. Then there exist constants $c, \gamma_2, \theta > 0$, and a polynomial-time constructible family $\{\psi_n(r, z)\}_{n>0}$ of 2-CSPs (with $|r| = cn, |z| = d(n) = O(\mathrm{poly}(n))$), such that for all n:*

(5.i) With probability $\geq 1 - 2^{-\Omega(n)}$ over r, $\exists z$ such that $\mathrm{Val}_{\psi_n}(r, z) = 1$;
(5.ii) If $C : \{0,1\}^{cn} \to \{0,1\}^{d(n)}$ is any circuit of size at most $2^{\gamma_2 n}$, then

$$\Pr_r[\mathrm{Val}_{\psi_n}(r, C(r)) > 1 - \theta] \leq 2^{-\Omega(n)} .$$

Theorem 4 (proved in the full version) and Theorem 5 both say that if NP (or $\mathsf{NP} \cap \mathsf{coNP}$) has sufficiently hard-on-average problems, we get an "inapproximable-on-average" optimization problem associated with a single, uniform family of 2-CSPs. The two results offer a tradeoff: Theorem 5 gives a slightly weaker conclusion than Theorem 4, but from a presumably likelier hardness assumption. The assumptions in the above results are strong but, we feel, plausible.[1] At the very least, Theorems 4 and 5 suggest that we must consider more restricted forms of the randomized CSP optimization problem to have a reasonable chance of proving positive results. We feel, however, that the randomized CSP optimization problem is worthy of study in its own right, even if Hypotheses 1 and 2 turn out to be false. We also feel that the CSPs families produced by Theorems 4 and 5 are interesting for the study of average-case hardness in NP, and might find further applications in complexity theory.

1.3 Our Methods

For the proofs of each of our main results, we use a powerful type of PCP known as *Probabilistically Checkable Proofs of Proximity (PCPPs)*. PCPPs were introduced independently by Ben-Sasson et al. [3] and by Dinur and Reingold [7], and the PCPPs we use were developed by Dinur [6] (in [7], [6] PCPPs are referred

[1] We comment that the assumption in Theorem 5 is implied by the assumption that there exists a *balanced* function $L \in \mathsf{NP}$ that is $(1 - 1/\mathrm{poly}(n))$-hard for some size $s(n) = 2^{\Omega(n)}$ [17], [11]. We also note that the assumptions in Theorems 4 and 5 are not known to imply $\mathsf{AM} = \mathsf{NP}$ under known hardness-vs.-randomness results (surveyed in [18]).

to as *Assignment Testers*). We will define PCPPs in Section 2, and in Section 3 we give a variant form of PCPPs that is more useful for our purposes. Our PCPP variant gives a general reduction (similar to past uses of PCPPs [3], [6]) in which we start with a two-argument circuit $Q(r, w)$ and efficiently produce a 2-CSP $\psi(r, z)$. The basic hope for our reduction is as follows: first, for any r, if the restricted circuit $Q(r, \cdot)$ is satisfiable, then the restricted 2-CSP $\psi(r, \cdot)$ should be satisfiable as well. Second, if $Q(r, \cdot)$ is unsatisfiable, then any assignment to $\psi(r, \cdot)$ should violate an $\Omega(1)$-fraction of the constraints in ψ. Unfortunately, this second requirement is too strong and cannot be met. What we *can* guarantee is that, if r is "far" in Hamming distance from any r' for which $Q(r', \cdot)$ is satisfiable, then for any z, (r, z) violates an $\Omega(1)$-fraction of constraints of ψ.

How does this reduction help us prove the prAM-hardness result in Theorem 1? Any instance x of a promise problem $\Pi = (\Pi_{YES}, \Pi_{NO})$ defines a predicate $Q(r, w)$ computed by a poly-size circuit. If $x \in \Pi_{YES}$ then for all r, $Q(r, \cdot)$ is satisfiable; while if $x \in \Pi_{NO}$ then for a 2/3 fraction of r, $Q(r, \cdot)$ is unsatisfiable. In order to apply our reduction, we need a stronger condition in the second case: a random choice of r should be *far* from any r' for which $Q(r', \cdot)$ is satisfiable. Such a property would hold if we could assume an extremely low error probability in our underlying Arthur-Merlin protocol. This cannot be achieved by straightforward parallel repetition, but it is provided by a theorem of Bellare et al. [2] which gives a randomness-efficient soundness-amplification for AM. Interestingly, Mossel and Umans [16] also used a similar amplification tool for their AM-hardness-of-approximation result on VC dimension, but for rather different purposes (unrelated to PCPs).

Our prAM-hardness proof is, we feel, more straightforward than the existing proofs of the analogous results for PSPACE and the Polynomial Hierarchy, modulo our use of sophisticated tools (PCPPs and efficient soundness-amplification) which we apply in a "black-box" fashion. However, the proof of the result for PSPACE in [4] uses property-testing ideas from PCP theory that prefigure the concept of PCPPs and are similar to our applications of PCPPs.

Next we discuss our methods in Theorems 4 and 5. Our transformation in Lemma 8 from the circuit Q to the 2-CSP ψ has a further useful property: we can reduce the problem of *finding* satisfying assignments to $Q(r, \cdot)$, to the problem of finding nearly-optimal assignments to $\psi(r, \cdot)$. Roughly speaking, we show the following. Suppose there is an algorithm $P(r)$ producing an assignment z, such that with some probability p over r, the assignment $(r, P(r))$ satisfies "almost all" of the constraints of ψ; then there is a second algorithm $\tilde{P}(r)$ such that $Q(r, \tilde{P}(r)) = 1$ with probability $p' \geq 2^{-\varepsilon|r|}p$. (Here, $\varepsilon > 0$ can be chosen arbitrarily small.) This property of the reduction is more novel, although the techniques we use (involving error-correcting codes) resemble previous works.

To apply our reduction, we use the hardness assumptions in Theorems 4 and 5 to produce poly-time predicates $Q(r, w)$ such that $Q(r, \cdot)$ is satisfiable with high probability, while any "small" witness-producing circuit C fails to solve the search problem associated with Q: that is, $Q(r, C(r)) = 0$ with high probability. Due to the exponential loss factor $2^{-\varepsilon|r|}$ in our reduction, we need

the search problem associated with Q to be *extremely* hard: every "small" circuit C must succeed with probability $\leq 2^{-\Omega(|r|)}$ over r in achieving $Q(r, C(r)) = 1$.

To produce such extremely hard search problems from a more "mild" hardness assumption, we use existing hardness-amplification techniques. In particular, we use the well-known Impagliazzo-Wigderson generator [14]. This generator, on input parameter n, takes a seed r of length $O(n)$, and produces n "pseudorandom" outputs $g_1, \ldots, g_n$ each of length n. The generator has the property that if language L is mildly hard for sufficiently small (but exponential-size) circuits, then any sufficiently smaller circuit has success probability $\leq 2^{-\Omega(n)}$ in correctly guessing the n-bit string $(L(g_1), \ldots, L(g_n))$. Then, if our hard language L is in $\mathsf{NP} \cap \mathsf{coNP}$ (as in Theorem 4), defining our predicate Q is straightforward: we let $Q(r, w) = 1$ iff w contains "proofs" for the n values $(L(g_1), \ldots, L(g_n))$.

If our hard language is merely in NP (as in Theorem 5), we need to work harder. In this case, we let $Q(r, w) = 1$ iff w contains proofs that $L(g_i) = 1$, for a "sufficient number" of the strings g_i. The idea is that, if a small circuit $C(r)$ could with some noticeable probability guess such proofs for "almost all" the indices i for which $L(g_i) = 1$, then C could be modified to correctly guess $(L(g_1), \ldots, L(g_n))$ with noticeable probability, contrary to the properties of the generator. To make this idea work, we prove that the set size $|\{i \in [n] : L(g_i) = 1\}|$ is highly concentrated around its expectation. For this we rely on a recently proved concentration result called the "strong Chernoff bound for expander walks" [19], [20], [10]. This result is perfectly suited to analyze the Impagliazzo-Wigderson generator, which is partly defined in terms of walks on expander graphs.

The precise form of our assumptions in Theorems 4 and 5 are dictated by the hardness-amplification tools currently available. In particular, sufficiently strong hardness-amplification is only available if we make a hardness assumption against nonuniform, exponential-sized circuits. We believe versions of Theorems 4 and 5 should be possible for a *uniform* hardness assumption; recently Impagliazzo et al. [13] made partial progress towards the hardness-amplification tools needed.

1.4 Questions for Future Work

Are there other, perhaps weaker, complexity-theoretic assumptions that give conclusions similar to Theorems 4 and 5? Alternatively, can we disprove Hypothesis 2 *unconditionally?* Given the sharp limitations of NC^0 circuits, this might be doable. Another open question we find intriguing is whether the promise problem in Theorem 1 remains prAM-complete when all variables are restricted to appear only $O(1)$ times each in the constraints. (The challenging part is to make the stochastic "Arthur-variables" obey this restriction.) Finally, can we use PCP ideas to prove new upper bounds on the class AM?

2 Preliminaries

For a CSP ψ and an assignment x, define $\mathrm{Val}_\psi(x)$ as the fraction of constraints ψ_j of ψ for which $\psi_j(x) = 1$. We assume familiarity with complexity classes

P, NP, AM, and MA, and with the Boolean circuit model of computation (which we'll review shortly). We define promise classes and the promise class prAM in Section 2.1. For a language $L \subseteq \{0,1\}^*$, we use $L(x)$ to denote the characteristic function of L. We use $|x|$ to denote the length of a (possibly non-Boolean) string x. $d(x,S)$ denotes the Hamming distance between $x \in \Sigma^n$, and a set $S \subseteq \Sigma^n$. If $d(x,S) > c$ we say x is *c-far from* S.

$H(t) : [0,1] \to [0,1]$ denotes the binary entropy function, $H(t) = -t\log_2 t - (1-t)\log_2(1-t)$ for $t \in (0,1)$ and $H(0) = H(1) = 0$. We let $V_{n,k}$ denote the discrete volume of the Hamming sphere of radius k in $\{0,1\}^n$; that is, $V_{n,k} := \sum_{0 \le i \le k} \binom{n}{i}$. We use the known bound $V_{n,\alpha n} \le 2^{H(\alpha)n}$ (for $\alpha \in [0,1/2]$).

When we speak of circuits, we mean deterministic Boolean circuits of fanin-two, and we measure the circuit size $|C|$ as the number of gates (including inputs). A function $F : \{0,1\}^n \to \{0,1\}^m$ is $p(n)$*-hard for size* $s(n)$ if for every Boolean circuit C of size $\le s(n)$, $\Pr_{x \in \{0,1\}^n}[C(x) = F(x)] \le p(n)$. Similarly, if this holds with $F(x) := L(x)$ for language L, we say L is $p(n)$-hard for size $s(n)$.

Next we define PCPPs. Fix a circuit $C(x)$ on n Boolean input variables, a finite alphabet Σ, and a parameter $\beta > 0$. We say a k-CSP ψ is a *PCPP for* C *over* Σ *with security* β if:

1. ψ is defined on variable set (x,z), where x are the Boolean input variables to C and z are auxiliary "proof" variables taking values in Σ;
2. For any $x \in \{0,1\}^n$, if $C(x) = 1$ then $\exists z$ such that $\mathrm{Val}_\psi(x,z) = 1$;
3. For all $x \in \{0,1\}^n$ and z, $\mathrm{Val}_\psi(x,z) \le 1 - \beta \cdot d(x, C^{-1}(1))/n$.

The *proof size* of ψ is the number of variables in z. Dinur showed:

Theorem 6. *[6, Cor. 8.4] There is a constant-size alphabet* Σ_0*, a constant* $\beta > 0$*, and a polynomial-time algorithm that, given a circuit* $Q(x)$ *of size* t*, produces a 2-CSP* $\psi_Q(x,z)$ *that is a PCPP for* Q *over* Σ_0 *with security* β*. Moreover, the proof size of* ψ_Q *is* $\widetilde{O}(t)$*.*

2.1 Promise Problems and prAM

A *promise problem* is a pair $\Pi = (\Pi_{YES}, \Pi_{NO})$ of disjoint subsets of $\{0,1\}^*$. We define prAM as the promise problems for which there is a polynomial-time algorithm $M(x,r,w)$, with $|r|, |w| = O(\mathrm{poly}(|x|))$, such that: (1) If $x \in \Pi_{YES}$, then for all r, $\exists w = w(r)$ such that $M(x,r,w) = 1$; (2) If $x \in \Pi_{NO}$, then $\Pr_r[\exists w : M(x,r,w) = 1] \le 1/3$. A promise problem $\Pi_1 = (\Pi_{YES}, \Pi_{NO})$ is prAM*-hard* if for all $\Pi' = (\Pi'_{YES}, \Pi'_{NO})$ in prAM, there exists a polynomial-time computable reduction $R(x)$, such that, if $x \in \Pi'_{YES}$, then $R(x) \in \Pi_{YES}$, while if $x \in \Pi'_{NO}$, then $R(x) \in \Pi_{NO}$. We say that Π is prAM*-complete* if Π is both in prAM and prAM-hard.

For any $\Pi \in$ prAM, the *soundness parameter* in the protocol, conventionally set as 1/3, can always be made exponentially small in $|x|$, using parallel

repetition of the original protocol. However, we need a more randomness-efficient soundness-amplification, provided by a result of Bellare et al. [2].[2]

Theorem 7. *[2] Let $\Pi = (\Pi_{YES}, \Pi_{NO}) \in$ prAM, where $M(x, r, w)$ is a polynomial-time predicate defining an Arthur-Merlin protocol for Π. Let $n = |x|$, and fix a polynomial $m(n)$. Then there exists an Arthur-Merlin protocol for Π defined by a polynomial-time predicate $M'(x, r', w')$, with $|w'| \leq O(\text{poly}(n)), |r| \leq |r'| \leq O(|r| + m(n))$, and with soundness $2^{-m(n)}$.*

3 An Augmented PCPP

As a tool for proving Theorems 1, 4, and 5, we need an "augmented" form of PCPPs. Lemma 8 below can be derived from Theorem 6; the proof is in the full version. We remark that the proof of Theorem 1 uses only condition (i) of Lemma 8; this first part of the Lemma is also present in previous uses of PCPPs [6], [3]. Condition (ii) is derived by an application of efficiently encodable/decodable error-correcting codes.

Lemma 8. *There is a finite alphabet Σ_0 such that the following holds. For any $\varepsilon > 0$ there is a $\nu > 0$ and a polynomial-time algorithm A that takes as input a Boolean circuit $C = C(r, w)$. A outputs a 2-CSP $\psi(r, z)$, where $|z| = O(\text{poly}(|C|))$ and the variables of z are over Σ_0. Letting $\ell = |r|$, we have:*

(8.i) For all r, if there is a w such that $C(r, w) = 1$, then there is a z such that $\text{Val}_\psi(r, z) = 1$. On the other hand, if r is $\alpha\ell$-far from any r' for which $C(r', \cdot)$ is satisfiable, then for all z, $\text{Val}_\psi(r, z) < 1 - \Omega(\alpha)$.

(8.ii) Suppose $P(r)$ is any function such that with probability at least $p = p(\ell)$ over a uniform $r \in \{0,1\}^\ell$, $P(r)$ outputs a z such that $\text{Val}_\psi(r, z) > 1 - \nu$. Then there exists a function $\tilde{P}(r)$, such that $\Pr_r[C(r, \tilde{P}(r)) = 1] \geq p(\ell) \cdot 2^{-\varepsilon\ell}$. Also, $\tilde{P}(r)$ is computable by a nonuniform, $\text{poly}(|C|)$-sized circuit that makes a single oracle call to P on the same input length.

4 Proof of Theorem 1

First, it is easy to see that $(\Pi_{AM-CSP,YES}, \Pi_{AM-CSP,NO})$ is in prAM, for any setting $p(\ell) \leq 2^{-\Omega(|\ell|)}$. Our main task is to show that the promise problem is prAM-hard, for appropriate choice of parameters. Let $\Pi = (\Pi_{YES}, \Pi_{NO}) \in$ prAM, and let $M_1(x, r_1, w_1)$ be a polynomial-time-computable predicate defining an Arthur-Merlin protocol for Π. We use parameters $n = |x|, \ell_1(n) = |r|$; by definition of prAM we have $\ell_1(n) = O(\text{poly}(n))$ and $|w_1| = O(\text{poly}(n))$. By padding r_1 if needed, we may ensure $\ell_1(n) \geq n$. Apply Theorem 7 to M_1, with the setting $m(n) := \ell_1(n)$. Thus we get a new Arthur-Merlin protocol $M_2(x, r_2, w_2)$

[2] They state their theorem for AM, not prAM, but the proof carries over without changes to the promise setting and we state it for this setting.

for Π, with $|r_2| = \ell_2(n) \in [n, \ldots, D \cdot \ell_1(n)]$ (for some fixed $D > 0$), $|w_2| = O(\text{poly}(n))$, and with soundness $2^{-\ell_1(n)}$.

Given an input $x \in \Pi_{YES} \cup \Pi_{NO}$, we construct a poly$(n)$-sized circuit $C(r_2, w_2) = C_x(r_2, w_2)$ that accepts iff $M_2(x, r_2, w_2) = 1$. To this circuit we apply the algorithm A of Lemma 8 (with $\varepsilon > 0$ to be announced), yielding a 2-CSP $\psi = \psi(r_2, z)$ which we make the output of our reduction.

We show the correctness of the reduction. First, suppose that $x \in \Pi_{YES}$. Then for each r_2, there exists a w_2 such that $M_2(x, r_2, w_2) = 1$. By condition (8.i) of Lemma 8, there exists z such that $\text{Val}_\psi(r_2, z) = 1$. Thus $\psi \in \Pi_{AM-CSP,YES}$.

Now suppose that $x \in \Pi_{NO}$. Then by the soundness property of M_2, the number of strings r_2 for which $M_2(x, r_2, \cdot)$ is satisfiable is at most $2^{-\ell_1(n)} \cdot 2^{\ell_2(n)} \leq 2^{(1-\frac{1}{D})\ell_2(n)}$. Thus for any $\alpha \in (0, 1/2)$, the number of r_2 for which there exists an r' at distance $\leq \alpha\ell_2(n)$ from r_2, such that $M_2(x, r', \cdot)$ is satisfiable, is at most $V_{\ell_2(n),\alpha\ell_2(n)} \cdot 2^{(1-\frac{1}{D})\ell_2(n)} \leq 2^{(H(\alpha)+1-\frac{1}{D})\ell_2(n)}$.

Choosing $\alpha > 0$ such that $H(\alpha) < \frac{1}{D}$, we find that with probability $\geq 1 - 2^{-\Omega(\ell_2(n))}$ over a uniform choice of r_2, r_2 is $\alpha\ell_2(n)$-far from any r' such that $C(r', \cdot)$ is satisfiable. For such r_2, condition (8.i) of Lemma 8 tells us $\text{Val}_\psi(r_2, z) < 1 - \Omega(\alpha)$ for any z. Thus if we fix $\varepsilon > 0$ as a small enough value and choose an appropriate function $p(|r_2|) \leq 2^{-\Omega(|r_2|)}$ to define $\Pi_{AM-CSP,NO}$, we have $\psi \in \Pi_{AM-CSP,NO}$. This proves Theorem 1.

5 Proof of Theorem 5

Our key tool to prove Theorem 5 is *randomness-efficient hardness amplification*. Say we're given an integer $c \geq 1$, a parameter $k = k(n)$, and a "generator" function $G(r) : \{0,1\}^{cn} \to \{0,1\}^{k \times n}$, where the string $G(r)$ is divided into k blocks $G_1(r), \ldots, G_k(r)$, each of length n. The hope is that $G_1(r), \ldots, G_k(r)$ should behave in certain respects like a truly independent collection of random strings, while satisfying $|r| \ll k \cdot n$. In particular, if it is somewhat hard to compute $L(x)$ for random x, it should be *very* hard to compute L correctly on $G_1(r), \ldots, G_k(r)$ simultaneously. Define $L^k \circ G : \{0,1\}^{cn} \to \{0,1\}^k$ by

$$(L^k \circ G)(r) := (L(G_1(r)), L(G_2(r)), \ldots, L(G_k(r))) \ . \tag{1}$$

Impagliazzo and Wigderson gave a generator (we denote it G_{IW}) with a strong hardness-amplification property:

Theorem 9. *[14, Thm. 2.12] For any $\gamma > 0$, there are $\gamma', c > 0$, and a polynomial-time computable $G_{IW} : \{0,1\}^{cn} \to \{0,1\}^{n \times n}$, such that: if L is 2/3-hard for size $2^{\gamma n}$, then $(L^n \circ G_{IW})(r)$ is $2^{-\gamma' n}$-hard for size $2^{\gamma' n}$.*

The generator G_{IW} has a second useful property: for any language L, with very high probability over r, the fraction of the strings $G_{IW,1}(r), \ldots, G_{IW,n}(r)$ which lie in L is close to $|L \cap \{0,1\}^n|/2^n$; that is, close to the fraction we'd expect if these strings were drawn independently and uniformly. To prove this fact (not proved or used in [14]), we need to describe the generator in more detail. The input r

to G_{IW} consists of two parts, $r = (r_a, r_b)$. $G_{IW}(r_a, r_b)$ is defined blockwise for $i \in [n]$ as

$$G_{IW,i}(r_a, r_b) = K_i(r_a) + K'_i(r_b) \ , \tag{2}$$

with $K_i : \{0,1\}^{|r_a|} \to \{0,1\}^n$, $K'_i : \{0,1\}^{|r_b|} \to \{0,1\}^n$, and with $+$ denoting bitwise addition mod 2. We describe K_i; the definition of K'_i is not important to us. The string r_a defines a random walk of length n (counting the starting vertex) on an explicit expander graph $\mathcal{G}_n$ with vertex set $\{0,1\}^n$. $\mathcal{G}_n$ is 16-regular with normalized second eigenvalue λ_n at most some fixed $\lambda < 1$. The value $v_i := K_i(r_a)$ represents the i^{th} vertex visited in this walk. v_1 is uniform element of $\{0,1\}^n$, and each subsequent step v_{i+1} is picked uniformly from among the neighbors of v_i. (Note, this can be achieved with $|r_a| = O(n)$, as required.)

We will use the following powerful result, called the "strong Chernoff bound for expander walks", as proved by Healy [10].[3]

Theorem 10. *[10] Let $\mathcal{G} = (V, E)$ be a d-regular graph with normalized second eigenvalue λ, let $m > 0$, and let $f_1, \ldots, f_m : V \to [0,1]$ have expectations $\mu_1, \ldots, \mu_m$ (over a uniform choice of $v \in V$). Taking a random walk $v_1, \ldots, v_m$ on $\mathcal{G}$ with uniform starting-point, we have for all $\varepsilon > 0$,*

$$\Pr[|\textstyle\sum_{i \le m} f_i(v_i) - \sum_{i \le m} \mu_i| \ge \varepsilon m] \le 2e^{-\frac{\varepsilon^2(1-\lambda)m}{4}} \ . \tag{3}$$

For a fixed language L and $r \in \{0,1\}^{cn}$, let $\sharp(r) := |\{i : L(G_{IW,i}(r)) = 1\}|$.

Lemma 11. *Let L be an arbitrary language. Let $c_n = |L \cap \{0,1\}^n|/2^n$. Then for any fixed $\delta > 0$, $\Pr_r[|\sharp(r) - c_n \cdot n| \ge \delta n] \le 2^{-\Omega(n)}$.*

Proof. Recall that $r = (r_a, r_b)$. It is enough to show that the above inequality is true after conditioning on any value of r_b. Let $(v'_1, \ldots, v'_n) = (K'_1(r_b), \ldots, K'_n(r_b))$. Then for $i \in [n]$, $G_{IW,i}(r) \in L$ iff $K_i(r_a) + v'_i \in L$, or equivalently $K_i(r_a) \in L_n + v'_i$ (where $L_n := L \cap \{0,1\}^n$ and $L_n + v'_i = \{x + v'_i : x \in L_n\}$).

Define $f_i : \{0,1\}^n \to \{0,1\}$ to be the characteristic function of $L_n + v'_i$. Clearly $\mu_i = c_n$ for all i. The needed inequality follows by a direct application of Theorem 10, using the fact that $\mathcal{G}_n$ has second eigenvalue bounded away from 1. This proves Lemma 11.

Now we prove Theorem 5. Our choice of ε_0 will be no larger than $1/6$, so our hypothesis implies there is a $\gamma' > 0$, a $c > 0$, and a poly-time $G_{IW} : \{0,1\}^{cn} \to \{0,1\}^{n \times n}$ for which Theorem 9 holds. Let $M(x, w)$ be a polynomial-time verifier for L: $x \in L$ iff $\exists w : M(x, w) = 1$. Let $t(n) = |w| = O(\text{poly}(n))$.

If $C : \{0,1\}^{cn} \to \{0,1\}^{t(n) \times n}$ is a circuit producing n strings $w_1, \ldots, w_n$, each of length $t(n)$, define $\sharp_C(r) := |\{i \in [n] : M(G_{IW,i}(r), w_i) = 1\}|$.

Claim 12. *There exists $\gamma'' > 0$ such that, if $C(r) : \{0,1\}^{cn} \to \{0,1\}^{t(n) \times n}$ is a circuit of size at most $2^{\gamma'' n}$, then $\Pr_r[\sharp_C(r) > \sharp(r) - \gamma'' n] < 2^{-\gamma'' n}$.*

[3] A more general claim was made earlier by Wigderson and Xiao [19], but the proof contained an error, as pointed out in [20]. (A valid proof of the Theorem above, with different constants, can still be extracted from [19].)

Claim 12 is proved in the full version; the idea is that if $\sharp_C(r) \approx \sharp(r)$ with noticeable probability, we could modify C to *guess* the small set of instances $G_{IW,i}(r) \in L$ for which it failed to find witnesses—allowing us to guess $(L^n \circ G_{IW})(r)$ with noticeable probability. This contradicts Theorem 9 if $|C|$ is small.

Now set $\varepsilon_0 := \min(1/6, \gamma''/4)$. Fix any circuit $C : \{0,1\}^{cn} \to \{0,1\}^{t(n)\times n}$ of size at most $2^{\gamma'' n}$. We use Lemma 11 applied to $\delta := \varepsilon_0$, along with Claim 12, to find that, with probability $\geq 1 - 2^{-\Omega(n)}$ over r, we have the simultaneous inequalities $(c_n + \varepsilon_0)n > \sharp(r) > (c_n - \varepsilon_0)n$ and $\sharp(r) \geq \sharp_C(r) + \gamma'' n$. Call a string r with this property *C-typical*. Now, we claim that $|c_n - 1/2| \leq \varepsilon_0$. For otherwise, a size-1 circuit could guess $L(x)$ with probability greater than $1/2+\varepsilon_0$ by guessing the majority value on length n, contrary to our hardness assumption about L. Thus for a C-typical r, $\sharp_C(r) \leq (1/2 + \varepsilon_0)n - \gamma'' n \leq (1/2 - 3\gamma''/4)n$ and also $\sharp(r) \geq (1/2 - \varepsilon_0)n - \varepsilon_0 n \geq (1/2 - \gamma''/2)n$.

Fix a rational value $\eta \in (1/2 - 3\gamma''/4, 1/2 - \gamma''/2)$. Define a predicate $Q(r, w_1, \ldots, w_n) : \{0,1\}^{cn} \times \{0,1\}^{n\times t(n)} \to \{0,1\}$ as follows: $Q(r, w_1, \ldots, w_n) = 1$ iff $|\{i : M(G_{IW,i}(r), w_i) = 1\}| \geq \eta \cdot n$. Q is polynomial-time computable, computed by some uniform family $\{Q_n\}_{n>0}$ of poly-size circuits. We have the key fact that for a C-typical r, $\exists w_1, \ldots, w_n : Q(r, w_1, \ldots, w_n) = 1$, yet $Q(r, C(r)) = 0$.

Invoke Lemma 8 with $\varepsilon := \gamma''/(2c)$, yielding an algorithm A (and an associated $\nu > 0$). Then we claim $\{A(Q_n)\}_{n>0} = \{\psi_n(r, z)\}_{n>0}$ is the desired family of 2-CSPs (here $|r| = cn, |z| = d(n) = O(\text{poly}(n))$). First we verify condition (5.i). Consider any r for which there exists a $w_1, \ldots, w_n$ such that $Q_n(r, w_1, \ldots, w_n) = 1$. By condition (8.i) of Lemma 8, we find that in this case there exists z such that $\text{Val}_{\psi_n}(r, z) = 1$. Since all but a $2^{-\Omega(n)}$ fraction of r have this property, condition (5.i) is satisfied.

To establish condition (5.ii), fix γ_2 as any value in $(0, \gamma'')$ and let $\theta := \nu$. Suppose $C(r) : \{0,1\}^{cn} \to \{0,1\}^{d(n)}$ is a circuit of size at most $2^{\gamma_2 n}$, such that with some probability $q(n)$, $\text{Val}_{\psi_n}(r, C(r)) > 1-\theta$. By condition (8.ii) of Lemma 8, there exists a circuit $\tilde{C}(r) : \{0,1\}^{cn} \to \{0,1\}^{n\times t(n)}$, such that with probability at least $q(n) \cdot 2^{-\varepsilon(cn)}$ over r, $Q_n(r, \tilde{C}(r)) = 1$. Note that such an r fails to be $\tilde{C}$-typical. Moreover, $\tilde{C}$ is of size at most $|C|+O(\text{poly}(n))$, which for large enough n is less than $2^{\gamma'' n}$. So, by our previous analysis we find that $q(n) \cdot 2^{-\varepsilon cn} \leq 2^{-\gamma'' n}$, i.e., $q(n) \leq 2^{(\varepsilon c - \gamma'')n} = 2^{-\gamma'' n/2} = 2^{-\Omega(n)}$. We have proved condition (5.ii). This completes the proof of Theorem 5.

Acknowledgements. I thank Scott Aaronson and Madhu Sudan for helpful comments.

References

1. Arora, S., Lund, C., Motwani, R., Sudan, M., Szegedy, M.: Proof verification and the hardness of approximation problems. J. ACM 45(3), 501–555 (1998)
2. Bellare, M., Goldreich, O., Goldwasser, S.: Randomness in interactive proofs. Computational Complexity 3, 319–354 (1993)

3. Ben-Sasson, E., Goldreich, O., Harsha, P., Sudan, M., Vadhan, S.P.: Robust PCPs of proximity, shorter PCPs, and applications to coding. SIAM J. Comput. 36(4), 889–974 (2006)
4. Condon, A., Feigenbaum, J., Lund, C., Shor, P.S.: Probabilistically checkable debate systems and nonapproximability of PSPACE-hard functions. Chicago J. Theor. Comput. Sci (1995)
5. Condon, A., Feigenbaum, J., Lund, C., Shor, P.S.: Random debaters and the hardness of approximating stochastic functions. SIAM J. Comput. 26(2), 369–400 (1997)
6. Dinur, I.: The PCP theorem by gap amplification. J. ACM 54(3) (2007)
7. Dinur, I., Reingold, O.: Assignment testers: Towards a combinatorial proof of the PCP theorem. SIAM J. Comput. 36(4), 975–1024 (2006)
8. Drucker, A.: PCPs for Arthur-Merlin Games and Communication Protocols. Master's thesis, Massachusetts Institute of Technology, Dept. of Electrical Engineering and Computer Science (2010)
9. Haviv, A., Regev, O., Ta-Shma, A.: On the hardness of satisfiability with bounded occurrences in the polynomial-time hierarchy. Theory of Computing 3(1), 45–60 (2007)
10. Healy, A.: Randomness-efficient sampling within NC^1. Computational Complexity 17(1), 3–37 (2008)
11. Healy, A., Vadhan, S.P., Viola, E.: Using nondeterminism to amplify hardness. SIAM J. Comput. 35(4), 903–931 (2006)
12. Impagliazzo, R.: Hard-core distributions for somewhat hard problems. In: Proc. 36th IEEE FOCS, pp. 538–545 (1995)
13. Impagliazzo, R., Jaiswal, R., Kabanets, V., Wigderson, A.: Uniform direct product theorems: Simplified, optimized, and derandomized. SIAM J. Comput. 39(4), 1637–1665 (2010)
14. Impagliazzo, R., Wigderson, A.: P = BPP if E requires exponential circuits: Derandomizing the XOR lemma. In: Proc. 29th ACM STOC, pp. 220–229 (1997)
15. Ko, K.-I., Lin, C.-C.: Non-approximability in the polynomial-time hierarchy. TR 94-2, Dept. of Computer Science, SUNY at Stony Brook (1994)
16. Mossel, E., Umans, C.: On the complexity of approximating the VC dimension. J. Comput. Syst. Sci. 65(4), 660–671 (2002)
17. O'Donnell, R.: Hardness amplification within NP. In: Proc. 34th ACM STOC, pp. 751–760 (2002)
18. Shaltiel, R., Umans, C.: Low-end uniform hardness vs. randomness tradeoffs for AM. SIAM J. Comput. 39(3), 1006–1037 (2009)
19. Wigderson, A., Xiao, D.: A randomness-efficient sampler for matrix-valued functions and applications. In: Proc. 46th IEEE FOCS, pp. 397–406 (2005)
20. Wigderson, A., Xiao, D.: Derandomizing the Ahlswede-Winter matrix-valued Chernoff bound using pessimistic estimators, and applications. Theory of Computing 4(1), 53–76 (2008)

Lower Bounds for Online Integer Multiplication and Convolution in the Cell-Probe Model

Raphaël Clifford and Markus Jalsenius

Department of Computer Science, University of Bristol, Bristol, UK

Abstract. We show time lower bounds for both online integer multiplication and convolution in the cell-probe model with word size w. For the multiplication problem, one pair of digits, each from one of two n digit numbers that are to be multiplied, is given as input at step i. The online algorithm outputs a single new digit from the product of the numbers before step $i+1$. We give a lower bound of $\Omega(\frac{\delta}{w}\log n)$ time on average per output digit for this problem where 2^δ is the maximum value of a digit. In the convolution problem, we are given a fixed vector V of length n and we consider a stream in which numbers arrive one at a time. We output the inner product of V and the vector that consists of the last n numbers of the stream. We show an $\Omega(\frac{\delta}{w}\log n)$ lower bound for the time required per new number in the stream. All the bounds presented hold under randomisation and amortisation. Multiplication and convolution are central problems in the study of algorithms which also have the widest range of practical applications.

1 Introduction

We consider two related and fundamental problems: multiplying two integers and computing the convolution or cross-correlation of two vectors. We study both these problems in an online or streaming context and provide time lower bounds in the cell-probe model. The importance of these problems is hard to overstate with both the integer multiplication and convolution problems playing a central role in modern algorithms design and theory.

For notational brevity, we write $[q]$ to denote the set $\{0,\ldots,q-1\}$, where q is a positive integer.

Problem 1 (Online convolution). For a fixed vector $V \in [q]^n$ of length n, we consider a stream in which numbers from $[q]$ arrive one at a time. For each arriving number, before the next number arrives, we output the inner product (modulo q) of V and the vector that consists of the last n numbers of the stream.

We show that there are instances of this problem such that any algorithm solving it will require $\Omega(\frac{\delta}{w}\log n)$ amortised time on average per output, where $\delta = \log_2 q$ and w is the number of bits per cell in the cell-probe model. The result is formally stated in Theorem 1.

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 593–604, 2011.

Problem 2 (Online multiplication). Given two numbers $X, Y \in [q^n]$, where q is the base and n is the number of digits per number, we want to output the n least significant digits of the product of X and Y, in base q. We must do this under the constraint that the ith digit of the product (starting from the lower-order end) is outputted before the $(i+1)$th digit, and when the ith digit is outputted, we only have access to the i least significant digits of X and Y, respectively. We can think of the digits of X and Y arriving online in pairs, one digit from each of X and Y.

We show that there are instances of this problem such that any algorithm solving it takes $\Omega(\frac{\delta}{w} \log n)$ time on average per input pair, where $\delta = \log_2 q$ and w is the number of bits per cell in the cell-probe model. The result is formally stated in Theorem 4.

Our main technical innovation is to extend recently developed methods designed to give lower bounds on dynamic data structures to the seemingly distinct field of online algorithms. Where $\delta = w$, for example, we have $\Omega(\log n)$ lower bounds for both online multiplication and convolution, thereby matching the currently best known offline upper bounds. As we discuss in the Section 1.1, this may be the highest lower bound that can be formally proved for these problems.

For the convolution problem, one consequence of our results is a new separation between the time complexity of exact and inexact string matching in a stream. The convolution has played a particularly important role in the field of combinatorial pattern matching where many of the fastest algorithms rely crucially for their speed on the use of fast Fourier transforms to perform repeated convolutions. These methods have also been extended to allow searching for patterns in rapidly processed data streams [1,2]. The results we present here therefore give the first strict separation between the constant time complexity of online exact matching [7] and any convolution based online pattern matching algorithm.

Although we show only the existence of probability distributions on the inputs for which we can prove lower bounds on the expected running time of any deterministic algorithm, by Yao's minimax principle [14] this also immediately implies that for every (randomised) algorithm, there is a worst-case input such that the (expected) running time is equally high. Therefore our lower bounds hold equally for randomised algorithms as for deterministic ones.

1.1 Previous Results and Upper Bounds

Lower bounds for the time complexity of online multiplication of two n-bit numbers have been studied in [10]. However, there the focus was on the *bit complexity* of the problem with an $\Omega(\log n)$ lower bound given for a multitape Turing machine and an $\Omega(\log(n)/\log\log(n))$ lower bound for a model equivalent to the modern bit-probe model. Neither online integer multiplication nor online convolution have been considered before in the cell-probe model to our knowledge.

The best known time complexities for both offline integer multiplication and convolution in the word-RAM model are $O(n \log n)$ by the well known application of fast Fourier transforms. As a consequence our online lower bounds match

the best known upper bounds for the offline problem. The question naturally arises as to whether one could find higher lower bounds for the online problem as there remains a gap between the best known upper and lower bounds. However, any offline algorithm for convolution or multiplication can be converted to an online one with at most an $O(\log n)$ factor overhead [4,1]. As a consequence, it is likely to be hard to prove a higher lower bound than we have given, at least for the case where $\delta/w \in \Theta(1)$, as this would immediately imply a superlinear lower bound for offline convolution or multiplication. Such superlinear lower bounds are not yet known for any problem in NP except in very restricted models of computation, such as for example a single tape Turing Machine. Our only alternative route to tight bounds would be to find better upper bounds for the online problems. For the case of online multiplication at least, this was first posed as an open problem almost 40 years ago and has so far proved hard to achieve [4].

1.2 The Cell-Probe Model

The lower bounds we develop hold in perhaps the strongest model of them all, the *cell-probe model* [9,5,15]. In this model, there is a separation between the computing unit and the memory, which is external and consists of a set of cells of w bits each. The computing unit cannot remember any information between operations. Computation is free and the cost is measured only in the number of cell reads or writes (cell-probes). Typically we think of the cell size w as being at least $\log_2 n$ bits, where n is the number of cells. This allows each cell to hold the address of any location in memory.

The generality of the cell-probe model makes it particularly attractive for establishing lower bounds for data structure problems and many such results have been given in the past couple of decades. The approaches taken have until recently mainly been based on communication complexity arguments and extensions of the chronogram technique of Fredman and Saks [6]. There remains however, a number of unsatisfying gaps between the lower bounds and known upper bounds. Only a few years ago, a breakthrough lead by Demaine and Pătraşcu gave us the tools to seal the gaps for several data structure problems [13]. The new technique was based on information theoretic arguments. Demaine and Pătraşcu also presented ideas which allowed them to express more refined lower bounds such as trade-offs between updates and queries of dynamic data structures. For a list of data structure problems and their lower bounds using these and related techniques, see for example [11].

2 Online Convolution

For a vector V of length n and $i \in [n]$, we write $V[i]$ to denote the elements of V. For positive integers n and q, the *inner product* of two vectors $U, V \in [q]^n$,

denoted $\langle U, V \rangle$, is defined as

$$\langle U, V \rangle = \sum_{i \in [n]} (U[i] \cdot V[i]) .$$

Parameterised by two positive integers n and q, and a fixed vector $V \in [q]^n$, the *online convolution problem* asks to maintain a vector $U \in [q]^n$ subject to an operation next(Δ), which takes a parameter $\Delta \in [q]$, modifies U to be the vector $(U[1], U[2], \ldots, U[n-1], \Delta)$ and then returns the inner product $\langle U, V \rangle$. In other words, next(Δ) modifies U by shifting all elements one step to the left, pushing the leftmost element out, and setting the new rightmost element to Δ. We consider the online convolution problem over the ring $\mathbb{Z}/q\mathbb{Z}$, that is integer arithmetic modulo q. Let $\delta = \log_2 q$.

Theorem 1. *For any positive integers q and n, in the cell probe model with w bits per cell there exist instances of the online convolution problem such that the expected amortised time per* next*-operation is $\Omega\left(\frac{\delta}{w} \log n\right)$, where $\delta = \log_2 q$.*

In order to prove Theorem 1 we will consider a random instance that is described by n next-operations on the sequence $\Delta = (\Delta_0, \ldots, \Delta_{n-1})$, where each Δ_i is chosen independently and uniformly at random from $[q]$. We defer the choice of the fixed vector V until later. For t from 0 to $n-1$, we use t to denote the time, and we say that the operation next(Δ_t) occurs at time t.

We may assume that prior to the first update, the vector $U = \{0\}^n$, although any values are possible since they do not influence the analysis. To avoid technicalities we will from now on assume that n is a power of two.

2.1 Information Transfer

Following the overall approach of Demaine and Pătraşcu [12] we will consider adjacent time intervals and study the *information* that is transferred from the operations in one interval to the next interval. More precisely, let $t_0, t_1, t_2 \in [n]$ such that $t_0 \leqslant t_1 < t_2$ and consider any algorithm solving the online convolution problem. We would like to keep track of the memory cells that are written to during the time interval $[t_0, t_1]$ and then read during the succeeding interval $[t_1+1, t_2]$. The information from the next-operations taking place in the interval $[t_0, t_1]$ that the algorithm passes on to the interval $[t_1+1, t_2]$ must be contained in these cells. Informally one can say that there is no other way for the algorithm to determine what occurred during the interval $[t_0, t_1]$ except through these cells. Formally, the *information transfer*, denoted $IT(t_0, t_1, t_2)$, is defined to be the set of memory cells c such that c is written during $[t_0, t_1]$, read at a time $t_r \in [t_1+1, t_2]$ and not written during $[t_1+1, t_r]$. Hence a cell that is overwritten in $[t_1+1, t_2]$ before being read is not included in the information transfer. Observe that the information transfer depends on the algorithm, the vector V and the sequence Δ. The first aim is to show that for any choice of algorithm solving the online convolution problem, the number of cells in the information transfer

is bounded from below by a sufficiently large number for some choice of the vector V.

For $0 \leqslant t_0 \leqslant t_1 < n$, we write $\Delta_{[t_0,t_1]}$ to denote the subsequence $(\Delta_{t_0}, \ldots, \Delta_{t_1})$ of Δ, and $\Delta_{[t_0,t_1]^c}$ to denote the sequence $(\Delta_0, \ldots, \Delta_{t_0-1}, \Delta_{t_1+1}, \ldots, \Delta_{n-1})$ which contains all the elements of Δ except for those in $\Delta_{[t_0,t_1]}$. For $t \in [n]$, we let $P_t \in [q]$ denote the inner product returned by next(Δ_t) at time t (recall that we operate modulo q). We let $P_{[t_1+1,t_2]} = (P_{t_1+1}, \ldots, P_{t_2})$.

Since Δ is a random variable, so is $P_{[t_1+1,t_2]}$. In particular, if we condition on a fixed choice of $\Delta_{[t_0,t_1]^c}$, call it $\Delta^{\text{fix}}_{[t_0,t_1]^c}$, then $P_{[t_1+1,t_2]}$ is a random variable that depends on the random values in $\Delta_{[t_0,t_1]}$. The dependency on the next-operations in the interval $[t_0, t_1]$ is captured by the information transfer $IT(t_0, t_1, t_2)$, which must encode all the relevant information in order for the algorithm to correctly output the inner products in $[t_1 + 1, t_2]$. In other words, an encoding of the information supplied by cells in the information transfer is an upper bound on the conditional *entropy* of $P_{[t_1+1,t_2]}$. This fact is stated in Lemma 1 and was given in [11] with small notational differences.

Lemma 1 (Lemma 3.2 of [11]). *The entropy*

$$H(P_{[t_1+1,t_2]} \mid \Delta_{[t_0,t_1]^c} = \Delta^{\text{fix}}_{[t_0,t_1]^c}) \leqslant w + 2w \cdot \mathbb{E}\left[|IT(t_0, t_1, t_2)| \mid \Delta_{[t_0,t_1]^c} = \Delta^{\text{fix}}_{[t_0,t_1]^c}\right].$$

2.2 Recovering Information

In the previous section, we provided an upper bound for the entropy of the outputs from the next-operations in $[t_1 + 1, t_2]$. Next we will explore how much information *needs to be* communicated from $[t_0, t_1]$ to $[t_1+1, t_2]$. This will provide a lower bound on the entropy. As we will see, the lower bound can be expressed as a function of the length of the intervals and the vector V.

Suppose that $[t_0, t_1]$ and $[t_1 + 1, t_2]$ both have the same length ℓ. That is, $t_1 - t_0 + 1 = t_2 - t_1 = \ell$. For $i \in [\ell]$, the output at time $t_1 + 1 + i$ can be broken into two sums S_i and S'_i, such that $P_{t_1+1+i} = S_i + S'_i$, where

$$S_i = \sum_{j \in [\ell]} (V[n - 1 - (\ell + i) + j] \cdot \Delta_{t_0+j})$$

is the contribution from the alignment of V with $\Delta_{[t_0,t_1]}$, and S'_i is the contribution from the alignments that do not include $\Delta_{[t_0,t_1]}$.

We define $M_{V,\ell}$ to be the $\ell \times \ell$ matrix with entries $M_{V,\ell}(i, j) = V[n - 1 - (\ell + i) + j]$ and observe that $M_{V,\ell}$ is a *Toeplitz* matrix (or "upside down" *Hankel* matrix) since it is constant on each descending diagonal from left to right. This property will be important later. From the definitions above it follows that

$$M_{V,\ell} \times \begin{pmatrix} \Delta_{t_0} \\ \Delta_{t_0+1} \\ \vdots \\ \Delta_{t_1} \end{pmatrix} = \begin{pmatrix} S_0 \\ S_1 \\ \vdots \\ S_{\ell-1} \end{pmatrix}. \tag{1}$$

We define the *recovery number* $R_{V,\ell}$ to be the number of variables $x \in \{x_1, \ldots, x_\ell\}$ such that x can be determined uniquely by the system of linear equations

$$M_{V,\ell} \times \begin{pmatrix} x_1 \\ \vdots \\ x_\ell \end{pmatrix} = \begin{pmatrix} y_1 \\ \vdots \\ y_\ell \end{pmatrix},$$

where we operate in $\mathbb{Z}/q\mathbb{Z}$. The recovery number may be distinct from the rank of a matrix, even where we operate over a field. As an example, consider the all ones matrix. The matrix will have recovery number zero but rank one. The recovery number is however related to the conditional entropy of $P_{[t_1+1,t_2]}$ as described by the next lemma.

Lemma 2. *If the intervals $[t_0, t_1]$ and $[t_1 + 1, t_2]$ both have the same length ℓ, then the entropy*

$$H(P_{[t_1+1,t_2]} \mid \Delta_{[t_0,t_1]^c} = \Delta^{\text{fix}}_{[t_0,t_1]^c}) \geqslant \delta R_{V,\ell}\,.$$

Proof. As described above, for $i \in [\ell]$, $P_{t_1+1+i} = S_i + S'_i$, where S'_i is a constant that only depends on V and $\Delta^{\text{fix}}_{[t_0,t_1]^c}$. Hence we can compute the values $S_0, \ldots, S_{\ell-1}$ from $P_{[t_1+1,t_2]}$. From Equation (1) it follows that $S_0, \ldots, S_{\ell-1}$ uniquely specify $R_{V,\ell}$ of the parameters in $\Delta_{[t_0,t_1]}$. That is, we can recover $R_{V,\ell}$ of the parameters from the interval $[t_0, t_1]$. Each of these parameters is a random variable that is uniformly distributed in $[q]$, so it contributes δ bits of entropy. □

We now combine Lemmas 1 and 2 in the following corollary.

Corollary 1. *For any fixed vector V, two intervals $[t_0, t_1]$ and $[t_1 + 1, t_2]$ of the same length ℓ, and any algorithm solving the online convolution problem on Δ chosen uniformly at random from $[q]^n$,*

$$\mathbb{E}\left[|IT(t_0, t_1, t_2)|\right] \geqslant \frac{\delta R_{V,\ell}}{2w} - \frac{1}{2}\,.$$

Proof. For $\Delta_{[t_0,t_1]^c}$ fixed to $\Delta^{\text{fix}}_{[t_0,t_1]^c}$, comparing Lemmas 1 and 2, we see that

$$\mathbb{E}\left[|IT(t_0, t_1, t_2)| \mid \Delta_{[t_0,t_1]^c} = \Delta^{\text{fix}}_{[t_0,t_1]^c}\right] \geqslant \frac{\delta R_{V,\ell}}{2w} - \frac{1}{2}\,.$$

The result follows by taking expectation over $\Delta_{[t_0,t_1]^c}$ under the random sequence Δ. □

2.3 The Lower Bound for Online Convolution

We now show how a lower bound on the total number of cell reads over n next-operations can be obtained by summing the information transfer between many pairs of time intervals. We again follow the approach of Demaine and

Pătraşcu [12], which involves conceptually constructing a balanced tree over the time axis. This *lower bound tree*, denoted $\mathcal{T}$, is a balanced binary tree on n leaves. Recall that we have assumed that n is a power of two. The leaves, from left to right, represent the time t from 0 to $n-1$, respectively. An internal node v is associated with the times t_0, t_1 and t_2 such that the two intervals $[t_0, t_1]$ and $[t_1+1, t_2]$ span the left subtree and the right subtree of v, respectively.

For an internal node v of $\mathcal{T}$, we write $IT(v)$ to denote $IT(t_0, t_1, t_2)$, where t_0, t_1, t_2 are associated with v. We write $L(v)$ to denote the number of leaves in the left (same as the right) subtree of v. The key lemma, stated next, is a modified version of Theorem 3.6 in [11]. The statement of the lemma is adapted to our online convolution problem and the proof relies on Corollary 1.

Lemma 3. *For any fixed vector V and any algorithm solving the online convolution problem, the expected running time of the algorithm over a sequence Δ that is chosen uniformly at random from $[q]^n$ is at least*

$$\frac{\delta}{2w} \sum_{v \in \mathcal{T}} R_{V,L(v)} - \frac{n-1}{2},$$

where the sum is over the internal nodes of $\mathcal{T}$.

Proof. We first consider a fixed sequence Δ. We argue that the number of read instructions executed by the algorithm is at least $\sum_{v \in \mathcal{T}} |IT(v)|$. To see this, for any read instruction, let t_r be the time it is executed. Let $t_w \leqslant t_r$ be the time the cell was last written, ignoring $t_r = t_w$. Then this read instruction (the cell it acts upon), is contained in $IT(v)$, where v is the lowest common ancestor of t_w and t_r. Thus, $\sum_{v \in \mathcal{T}} |IT(v)|$ never double-counts a read instruction.

For a random Δ, an expected lower bound on the number of read instructions is therefore $\mathbb{E}[\sum_{v \in \mathcal{T}} |IT(v)|]$. Using linearity of expectation and Corollary 1, we obtain the lower bound in the statement of the lemma. □

Lower bound with a random vector V. We have seen in Lemma 3 that a lower bound is highly dependent on the recovery numbers of the vector V. In the next lemma, we show that a random vector V has recovery numbers that are large.

Lemma 4. *Suppose that q is a prime and the vector V is chosen uniformly at random from $[q]^n$. Then $\mathbb{E}[R_{V,\ell}] \geqslant \ell/2$ for every length ℓ.*

Proof. Recall that $M_{V,\ell}$ is an $\ell \times \ell$ Toeplitz matrix. It has been shown in [8] that for any ℓ, out of all the $\ell \times \ell$ Toeplitz matrices over a finite field of q elements, a fraction of exactly $(1 - 1/q)$ is non-singular. This fact was actually already established in [3] almost 40 years earlier but incidentally reproved in [8]. Since we have assumed in the statement of the lemma that q is a prime, the ring $\mathbb{Z}/q\mathbb{Z}$ we operate in is indeed a finite field. The diagonals of $M_{V,\ell}$ are independent and uniformly distributed in $[q]$, hence the probability that $M_{V,\ell}$ is invertible is $(1 - 1/q) \geqslant 1/2$. If $M_{V,\ell}$ is invertible then the recovery number $R_{V,\ell} = \ell$; there

is a unique solution to the system of linear equations in Equation (1). On the other hand, if $M_{V,\ell}$ is not invertible then the recovery number will be lower. Thus, we can safely say that the expected recovery number $R_{V,\ell}$ is at least $\ell/2$, which proves the lemma. □

Before we give a lower bound for a random choice of V in Theorem 3 below, we state the following fact.

Fact 2. *For a balanced binary tree with n leaves, the sum of the number of leaves in the subtree rooted at v, taken over all internal nodes v, is $n \log_2 n$.*

Theorem 3. *Suppose that q is a prime. In the cell-probe model with w bits per cell, any algorithm solving the online convolution problem on a vector V and Δ, both chosen uniformly at random from $[q]^n$, will run in $\Omega\left(\frac{\delta}{w} n \log n\right)$ time in expectation, where $\delta = \log_2 q$.*

Proof. For a random vector V, a lower bound is obtained by taking the expectation of the bound in the statement of Lemma 3. Using linearity of expectation and applying Lemma 4 and Fact 2 completes the proof. □

Remark. Theorem 3 requires that q is a prime but for an integer $\delta > 1$, $q = 2^\delta$ is not a prime. However, we know that there is always at least one prime p such that $2^{\delta-1} < p < 2^\delta$. Thus, Theorem 3 is applicable for any integer δ, only with an adjustment by at most one.

Lower bound with a fixed vector V. We demonstrate next that it is possible to design a fixed vector V with guaranteed large recovery numbers. We will use this vector in the proof of Theorem 1. The idea is to let V consist of stretches of 0s interspersed by 1s. The distance between two succeeding 1s is an increasing power of two, ensuring that for half of the alignments in the interval $[t_1 + 1, t_2]$, all but exactly one element of $\Delta_{[t_0,t_1]}$ are simultaneously aligned with a 0 in V. We define the binary vector $K_n \in [2]^n$ to be

$$K_n = (\ldots \mathtt{0000000000001000000000000001000000010001011 0}),$$

or formally,

$$K_n[i] = \begin{cases} 1, & \text{if } n-1-i \text{ is a power of two;} \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

Lemma 5. *Suppose $V = K_n$ and $\ell \geqslant 1$ is a power of two. The recovery number $R_{V,\ell} \geqslant \ell/2$.*

Proof. Recall that entry $M_{V,\ell}(i,j) = V[n-1-(\ell+i)+j]$. Thus, $M_{V,\ell}(i,j) = 1$ if and only if $n-1-(n-1-(\ell+i)+j) = \ell+i-j$ is a power of two. It follows that for row $i = \ell/2, \ldots, \ell-1$, $M_{V,\ell}(i,j) = 1$ for $j = i$ and $M_{V,\ell}(i,j) = 0$ for $j \neq i$. This implies that the recovery number $R_{V,\ell}$ is at least $\ell/2$. □

We finally give the proof of Theorem 1.

Proof (Theorem 1). We assume that n is a power of two. Let $V = K_n$. It follows from Lemma 5 and Fact 2 that $\sum_{v \in \mathcal{T}} R_{V,L(v)} \geqslant \sum_{v \in \mathcal{T}} L(v)/2 = \Omega(n \log n)$. Note that $L(v)$ is a power of two for every node v in $\mathcal{T}$. For Δ chosen uniformly at random from $[q]^n$, apply Lemma 3 to obtain the expected running time $\Omega\left(\frac{\delta}{w} n \log n\right)$ over n next-operations. □

3 Online Multiplication

In this section we consider online multiplication of two n-digit numbers in base $q \geqslant 2$. For a non-negative integer X, let $X[i]$ denote the ith digit of X in base q, where the positions are numbered starting with 0 at the right (lower-order) end. We think of X padded with zeros to make sure that $X[i]$ is defined for arbitrary large i. For $j \geqslant i$, we write $X[i\,..\,j]$ to denote the integer that is written $X[j] \cdots X[i]$ in base q.

The *online multiplication problem* is defined as follows. The input is two n-digit numbers $X, Y \in [q^n]$ in base q (higher order digits may be zero). Let $Z = X \times Y$. We want to output the n lower order digits of Z in base q (i.e. $Z[0], \ldots, Z[n-1]$) under the constraint that $Z[i]$ must be outputted before $Z[i+1]$ and when $Z[i]$ is outputted, we are not allowed to use any knowledge of the digits $X[i+1], \ldots, X[n-1]$ and $Y[i+1], \ldots, Y[n-1]$. We can think of the digits of X and Y arriving one pair at a time, starting with the least significant pair of digits, and we output the corresponding digit of the product of the two numbers seen so far.

We also consider a variant of the online multiplication problem when one of the two input numbers, say Y, is known in advance. That is, all its digits are available at every stage of the algorithm and only the digits of X arrive in an online fashion. In particular we will consider the case when $Y = K_{q,n}$ is fixed, where we define $K_{q,n}$ to be the largest number in $[q^n]$ such that the ith bit in the binary expansion of $K_{q,n}$ is 1 if and only if i is a power of two (starting with $i = 0$ at the lower-order end). Note that the binary expansion of $K_{q,n}$ is the reverse of $K_{(n \log_2 q)}$ in Equation (2). We will prove the following theorem.

Theorem 4. *For any positive integers δ and n in the cell probe model with w bits per cell, the expected running time of any algorithm solving the online multiplication problem on two n-digit random numbers $X, Y \in [q^n]$ is $\Omega(\frac{\delta}{w} n \log n)$, where $q = 2^\delta$ is the base. The same bound holds even under full access to every digit of Y, and when $Y = K_{q,n}$ is fixed.*

It suffices to prove the lower bound for the case when we have full access to every digit of Y; we could always ignore digits. We prove Theorem 4 using the same approach as for the online convolution problem. Here the next-operation delivers a new digit of X, which is chosen uniformly at random from $[q]$, and outputs the corresponding digit of the product of X and Y.

For $t_0, t_1, t_2 \in [n]$ such that $t_0 \leqslant t_1 < t_2$, we write $X[t_0, t_1]^{\mathrm{c}}$ to denote every digit of X (in base q) except for those at position t_0 through t_1. It is helpful to think of $X[t_0, t_1]^{\mathrm{c}}$ as a vector of digits rather than a single number. We write

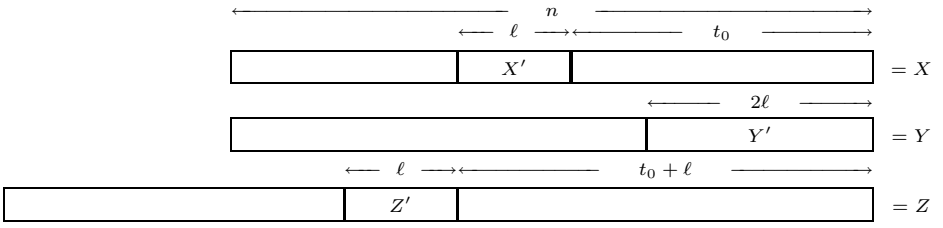


Fig. 1. X, Y and $Z = X \times Y$ in base q

$X^{\text{fix}}[t_0, t_1]^{\text{c}}$ to denote a fixed choice of $X[t_0, t_1]^{\text{c}}$. During the interval $[t_1 + 1, t_2]$, we output $Z[(t_1 + 1)\,..\,t_2]$. The information transfer is defined as before, and Lemma 1 is replaced with the following lemma.

Lemma 6. *The entropy*

$$H(Z[(t_1+1)\,..\,t_2] \mid X[t_0,t_1]^{\text{c}} = X^{\text{fix}}[t_0,t_1]^{\text{c}}) \leqslant w + 2w \cdot \mathbb{E}\left[|IT(t_0,t_1,t_2)| \mid X[t_0,t_1]^{\text{c}} = X^{\text{fix}}[t_0,t_1]^{\text{c}}\right] .$$

3.1 Retrorse Numbers and the Lower Bound

In Figure 1, the three numbers X, Y and $Z = X \times Y$ are illustrated with some segments of their digits labelled X', Y' and Z'. Informally, we say that Y' is *retrorse* if Z' depends heavily on X'. We have borrowed the term from Paterson, Fischer and Meyer [10], however, we give it a more precise meaning, formalised below.

Suppose $[t_0, t_1]$ and $[t_1 + 1, t_2]$ both have the same length ℓ. For notational brevity, we write X' to denote $X[t_0\,..\,t_1]$, Y' to denote $Y[0\,..\,(2\ell - 1)]$ and Z' to denote $Z[(t_1 + 1)\,..\,t_2]$ (see Figure 1). We say that Y' is *retrorse* if for any fixed values of t_0, $X[t_0, t_1]^{\text{c}}$ (the digits of X outside $[t_0, t_1]$) and $Y[2\ell\,..\,(n-1)]$, each value of Z' can arise from at most four different values of X'. That is to say there is at most a four-to-one mapping from possible values of X' to possible values of Z'. We define $I_{Y,\ell} = \ell$ if Y' is retrorse, otherwise $I_{Y,\ell} = 0$. Note that $I_{Y,\ell}$ only depends on Y and ℓ. We will use $I_{Y,\ell}$ similarly to the recovery number $R_{V,\ell}$ from Section 2.2 and replace Lemma 2 with Lemma 7 below, which combined with Lemma 6 gives us Corollary 2.

Lemma 7. *If the intervals $[t_0, t_1]$ and $[t_1 + 1, t_2]$ both have the same length ℓ, then the entropy*

$$H(Z[(t_1+1)\,..\,t_2] \mid X[t_0,t_1]^{\text{c}} = X^{\text{fix}}[t_0,t_1]^{\text{c}}) \geqslant \frac{\delta I_{Y,\ell}}{4} - \frac{1}{2} .$$

Corollary 2. *For any fixed number Y, two intervals $[t_0, t_1]$ and $[t_1 + 1, t_2]$ of the same length ℓ, and any algorithm solving the online multiplication problem on X chosen uniformly at random from $[q^n]$,*

$$\mathbb{E}\left[\,|IT(t_0,t_1,t_2)|\,\right] \geqslant \frac{\delta I_{Y,\ell}}{8w} - 1\,.$$

We take the same approach as in Section 2.3 and use a lower-bound tree $\mathcal{T}$ with n leaves to obtain the next lemma. The proof is identical to the proof of Lemma 3, only that we use Corollary 2 instead of Corollary 1.

To avoid technicalities we will assume that n and δ are both powers of two and we let the base $q = 2^\delta$.

Lemma 8. *For any fixed number Y and any algorithm solving the online multiplication problem, the expected running time of the algorithm with the number X chosen uniformly at random from $[q^n]$ is at least*

$$\frac{\delta}{8w} \sum_{v \in \mathcal{T}} I_{Y,L(v)} - (n-1)\,.$$

Before giving the proof of Theorem 4, we bound the value of $I_{Y,\ell}$ for both a random number Y and $Y = K_{q,n}$. In order to do so, we will use the following two results by Paterson, Fischer and Meyer [10] which apply to binary numbers. The lemmas are stated in our notation, but the translation from the original notation of [10] is straightforward.

Lemma 9 (Lemma 1 of [10]). *For the base $q = 2$ and fixed values of t_0, ℓ, n and $X[t_0,t_1]^{\mathrm{c}}$ (where $t_1 = t_0 + \ell - 1$), such that ℓ is a power of two, each value of Z' can arise from at most two values of X' when $Y = K_{2,n}$.*

Lemma 10 (Corollary of Lemma 5 in [10]). *For the base $q = 2$ and fixed values of t_0, ℓ, n and $X[t_0,t_1]^{\mathrm{c}}$ (where $t_1 = t_0 + \ell - 1$), such that ℓ is a power of two, at least half of all possible values of Y' have the property that each value of Z' can arise from at most four different values of X'.*

Lemma 11. *If ℓ is a power of two, then for a random $Y \in [q^n]$, $\mathbb{E}[I_{Y,\ell}] \geqslant \ell/2$, and for $Y = K_{q,n}$, $I_{Y,\ell} = \ell$.*

Proof. Suppose first that $Y = K_{q,n}$. Let ℓ be a power of two and t_0 a non-negative integer. We define X', Y' and Z' as before (see Figure 1). Instead of writing the numbers in base q, we consider their binary expansions, in which each digit is represented by $\delta = \log_2 q$ bits. In binary, we can write X, Y and Z as in Figure 1 if we replace n, t_0 and ℓ with δn, δt_0 and $\delta\ell$, respectively. Note that $\delta\ell$ is a power of two. Since $K_{q,n} = K_{2,\delta n}$, it follows immediately from Lemma 9 that Y' is retrorse and hence $I_{Y,\ell} = \ell$.

Suppose now that Y is chosen uniformly at random from $[q^n]$, hence Y' is a random number in $[q^{2\ell}]$. From Lemma 10 it follows that Y' is retrorse with probability at least a half. Thus, $\mathbb{E}[I_{Y,\ell}] \geqslant \ell/2$. □

Proof (Theorem 4). We assume that n is a power of two. Let Y be a random number in $[q^n]$, either under the uniform distribution or the distribution in which $K_{q,n}$ has probability one and every other number has probability zero.

A lower bound on the running time is obtained by taking the expectation of the bound in the statement of Lemma 8. Using linearity of expectation and applying Lemma 11 and Fact 2 finish the proof. Note from Lemma 11 that the expected value $\mathbb{E}[I_{Y,\ell}] = \ell$ when $Y = K_{q,n}$. □

Acknowledgements

We are grateful to Mihai Pătraşcu for suggesting the connection between online lower bounds and the recent cell-probe results for dynamic data structures and for very helpful discussions on the topic. MJ was supported by the EPSRC.

References

1. Clifford, R., Efremenko, K., Porat, B., Porat, E.: A black box for online approximate pattern matching. In: Ferragina, P., Landau, G.M. (eds.) CPM 2008. LNCS, vol. 5029, pp. 143–151. Springer, Heidelberg (2008)
2. Clifford, R., Sach, B.: Pattern matching in pseudo real-time. Journal of Discrete Algorithms 9(1), 67–81 (2011)
3. Daykin, D.E.: Distribution of bordered persymmetric matrices in a finite field. Journal für die reine und angewandte Mathematik 203, 47–54 (1960)
4. Fischer, M.J., Stockmeyer, L.J.: Fast on-line integer multiplication. In: STOC 1973: Proc. 5th Annual ACM Symposium Theory of Computing, pp. 67–72 (1973)
5. Fredman, M.L.: Observations on the complexity of generating quasi-gray codes. SIAM Journal on Computing 7(2), 134–146 (1978)
6. Fredman, M.L., Saks, M.: The cell probe complexity of dynamic data structures. In: STOC 1989: Proc. 21st Annual ACM Symposium Theory of Computing, pp. 345–354 (1989)
7. Galil, Z.: String matching in real time. Journal of the ACM 28(1), 134–149 (1981)
8. Kaltofen, E., Lobo, A.: On rank properties of Toeplitz matrices over finite fields. In: ISSAC 1996: Proc. of the 1996 International Symposium on Symbolic and Algebraic Computation, pp. 241–249 (1996)
9. Minsky, M., Papert, S.: Perceptrons: An Introduction to Computational Geometry. MIT Press, Cambridge (1969)
10. Paterson, M.S., Fischer, M.J., Meyer, A.R.: An improved overlap argument for on-line multiplication. In: Proceedings of SIAM-AMS, vol. 7, pp. 97–111. Amer. Math. Soc., Providence (1974)
11. Pătraşcu, M.: Lower Bound Techniques for Data Structures. PhD thesis, Massachusetts Institute of Technology (2008)
12. Pătraşcu, M., Demaine, E.D.: Tight bounds for the partial-sums problem. In: SODA 2004: Proc. 15th ACM/SIAM Symposium on Discrete Algorithms, pp. 20–29 (2004)
13. Pătraşcu, M., Demaine, E.D.: Logarithmic lower bounds in the cell-probe model. SIAM Journal on Computing 35(4), 932–963 (2006)
14. Yao, A.C.-C.: Probabilistic computations: Toward a unified measure of complexity. In: FOCS 1977: Proc. 18th Annual Symposium on Foundations of Computer Science, pp. 222–227 (1977)
15. Yao, A.C.-C.: Should tables be sorted? Journal of the ACM 28(3), 615–628 (1981)

Automatizability and Simple Stochastic Games

Lei Huang* and Toniann Pitassi*

University of Toronto
{leih,toni}@cs.toronto.edu

Abstract. The complexity of simple stochastic games (SSGs) has been open since they were defined by Condon in 1992. Despite intensive effort, the complexity of this problem is still unresolved. In this paper, building on the results of [4], we establish a connection between the complexity of SSGs and the complexity of an important problem in proof complexity–the proof search problem for low depth Frege systems. We prove that if depth-3 Frege systems are weakly automatizable, then SSGs are solvable in polynomial-time. Moreover we identify a natural combinatorial principle, which is a version of the well-known Graph Ordering Principle (GOP), that we call the integer-valued GOP (IGOP). We prove that if depth-2 Frege plus IGOP is weakly automatizable, then SSG is in P.

1 Introduction

In a groundbreaking paper from 1992, Condon defined the family of *simple stochastic games* (SSGs) [13]. A simple stochastic game is a directed graph with four types of vertices: min nodes, max nodes, average nodes, and two sinks, a 0-sink and a 1-sink. The game is played by two players, Max and Min, from a given start node. The goal of Max is to reach the 1-sink, while the goal of Min is to reach the 0-sink, or to continue the game indefinitely. The value of the game is the probability that Max wins, assuming that both players play optimally.

A fascinating open question is determining the complexity of finding the value of a game, or equivalently, the complexity of finding an optimal strategy. From a practical point of view, stochastic games are used to model a variety of problems in software verification and controller optimization, and from a theoretical point of view, the SSG problem is fundamental since an efficient algorithm for it will imply algorithms for a host of other problems. Indeed, it is known to be polynomial-time equivalent to many other computational problems such as the generalized linear complementarity problem, and the minimum stable circuit problem. Many other game-theoretic problems, such as mean payoff games, discounted payoff games, and parity games, all reduce to SSGs [2]. Furthermore, SSG is also the complete problem for the class AUC-SPACE($\log n$) of logspace bounded alternating random turing machines [13].

Despite considerable effort over the last decades, it remains a longstanding open problem to determine the complexity of SSGs. Condon [13,14] proved that the SSG decision problem, "does the max player have a strategy ensuring at least $\frac{1}{2}$ probability of winning", is in both NP and coNP. This makes SSGs one of the few combinatorial problems known to be in $NP \cap coNP$ and suggests that it is very unlikely to be

* Supported by NSERC.

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 605–617, 2011.

NP-complete. Another natural avenue for obtaining a hardness result for SSGs are the complexity classes associated with total functions in NP, such as PLS and PPAD. In fact, recently the complexity of the Nash Equilibrium problem was resolved in a celebrated sequence of papers, culminating in a proof of the PPAD-completeness of Nash [11,15] Unfortunately, such a hardness result for SSGs is unlikely, as it was shown to lie in *both* PLS and PPAD [6].

The SSG problem has also been studied extensively from an algorithmic point of view [14,17,24,22,18,5,21]. Many restrictions such as allowing only two of three types of vertices, or to a constant number of random nodes, are known to have polynomial time algorithms [14,17]. However, the current best algorithm for the full game is a $O(2^{\sqrt{n}})$ time randomized algorithm [22,18], and a $O(n \cdot |V_r|!)$ time deterministic algorithm [17], where V_r is the set of average nodes.

A seemingly unrelated, but also longstanding and important problem is to determine whether proofs in standard proof systems can be found efficiently. A proof system $\mathcal{P}$ is *automatizable* if there exists an algorithm that takes as input a tautology f, and outputs a $\mathcal{P}$-proof of f, and such that the runtime is polynomial in the size of the shortest $\mathcal{P}$-proof of f. $\mathcal{P}$ is *weakly automatizable* if there is an algorithm that on input f and a number r in unary, can distinguish the case where f is not a tautology from the case where f has a $\mathcal{P}$-proof of size at most r. It is known that $\mathcal{P}$ is weakly-automatizable if and only if there is an automatizable proof system that simulates $\mathcal{P}$ [3].

The question of whether standard proof systems are automatizable is a fundamental question in logic and automated theorem proving [9]. Following [20], it was shown that Frege systems are not weakly automatizable under a widely believed cryptographic assumption – that the Diffie Hellman (DH) problem (and hence factoring) is hard to compute. However, despite considerable effort, the weak automatizability question for low-depth Frege systems is unresolved. [7] showed that AC_0^k-Frege is not weakly automatizable under the assumption that the DH problem can be solved in time $exp(n^{c/k})$, where $c > 2$. The best algorithm for DH runs in time $exp(n^{1/2})$, and moreover the number field sieve is conjectured to solve DH in time $exp(n^{1/3})$; thus for small k (k less than 5 or 6), the weak automatizability of depth-k Frege is unresolved. Even for depth-1 Frege (i.e. resolution), there is no clear evidence for or against weak automatizability, despite results in both directions [1,12].

Results and Related Work. In a recent paper, Atserias and Maneva made an important new link between automatizability and game theory by proving that solving mean payoff games (MPGs) is reducible to the weak automatizability of depth-2 Frege systems and to feasible interpolation of depth-3 Frege systems[4]. In this paper, we prove that if depth-3 Frege systems are weakly automatizable, then simple stochastic games are solvable in polynomial time, thus establishing a link between SSGs and an important open problem in proof complexity.

Mean payoff games can be viewed as a special case of simple stochastic games, but depth-2 Frege systems can also be viewed as a special case of depth-3 Frege thus the results cannot be directly compared. However, the increase in depth suggests an approach to pinpointing a difference between MPGs and SSGs, an interesting open problem in its own right. Moreover, the only part of our proof that is not contained in depth-2 is in the proof of a natural combinatorial property about graphs that we will

call the Integer-Valued Graph Ordering Principle (IGOP). IGOP states informally that in any finite undirected graph where all nodes are labelled by integers, there exists a vertex whose value is at least as large as its neighbors. This principle is expressible as a CNF formula, and we actually prove that if depth-2 Frege, augmented with IGOP, is weakly automatizable, then SSGs are in P. This raises the very interesting question as to the exact proof theoretic strength required to prove IGOP: If it has a polynomial size depth-2 Frege proof, then our result subsumes that of [4]. On the other hand, if not, then we have found a natural CNF formula separating depth-2 from depth-3 Frege, and furthermore, expose an essential difference between mean payoff games and simple stochastic games.

Our proof builds on [4] in that we use and further develop very low depth circuits that they invent for performing addition of a constant number of integers. However, at a high level our proofs are very different. [4] goes through an intermediate transformation to the Max Atom Problem that appears to hold only for mean payoff games. In contrast our proof strategy is much more generalizable. The main idea relies on the unique fixed point property of SSGs - ie SSGs can be characterized as a fixed point computation where the unique fixed point indicates the winner. This property is shared by a variety of other problems, including the more general class of stochastic games defined by Shapley [23].

Proof Overview. The basic idea behind our proof is to study the complexity of *stopping* games, which are polynomial-time equivalent to general SSGs. In a stopping game, the optimal solution is also the only solution to a set of local optimality conditions. We formalize the formulas, $MinWin(G)$ (and $MaxWin(G)$) expressing that there is a locally optimal strategy with value less than or equal to one half (or greater than one half). Since the locally optimal solution is always unique for stopping games, the formula $F(G) = MinWin(G) \wedge MaxWin(G)$ is unsatisfiable. The bulk of our argument is thus to show that $F(G)$, has an efficient depth-3 Frege refutation. Then if depth-3 Frege has feasible interpolation, the interpolant for F will return whether $MinWin(G)$ or $MaxWin(G)$ is unsatisfiable, thus revealing the winner of the game. Since weak automatizability implies interpolation, it immediately follows that weak automatizability of depth-3 Frege implies that SSG is in P. The technical difficulty is to prove that for any stopping game G, the locally optimal solution is unique. This involves a very careful analysis of low depth circuits for performing arithmetic on sets of numbers, as well as very efficient reasoning about these arithmetic formulas.

2 Definitions

Simple Stochastic Games. In a *simple stochastic game* (SSG) two players take turns moving a pebble along a directed graph $G : (V, E)$ with two terminal positions, a 0-sink and a 1-sink. Conventionally the game also has a unique source node from which gameplay starts. All non terminal nodes of G are partitioned into max nodes, min nodes, and random nodes. From a max (min) nodes, Max (Min) chooses the out-edge the pebble takes next. At a random node, the successor node is chosen by chance. We will consider only binary SSGs where every node has out degree two and random nodes choose each out-edge with probability $\frac{1}{2}$. All SSGs have such a binary equivalent.

In a given play, the max player wins if the pebble reaches the 1-sink and the min player wins otherwise (either by reaching the 0-sink or continuing play indefinitely). Let σ be a strategy for Max, $\sigma : V^* \to V$ mapping paths in a play to a legal successor node, and τ be a strategy for Min. The *value* of a node under σ and τ, $v_{\sigma,\tau}(i)$, is defined to be the probability that the game ends at 1-sink if the players use strategies σ, τ and the pebble starts at node i. The *optimal value* $v_{opt}(i)$ of a node is the minimax over all strategies $v_{opt}(i) = \max_\sigma \min_\tau v_{\sigma,\tau}(i) = \min_\tau \max_\sigma v_{\sigma,\tau}(i)$ The value of the game is $v_{opt}(0)$ where node 0 is the start node.

Fact 1. *(Shapley, 53) For every SSG there exists pure positional strategies σ and τ that achieve $v_{opt}(i)$ at every node i.*

A strategy can be thought of as simply a mapping $V \to V$ and optimal strategies ergodic, they are optimal regardless of which vertex is the start vertex.

Fact 2. *(Condon, 92) The optimal values of any n node SSG can be written as $\frac{a}{b}$ where $a, b \in \mathbb{N}$, $b = O(2^n)$*

Definition 1. *For a simple stochastic game G we define the associated function $I_G : \{0,1\}^n \to \{0,1\}^n$ as follows.*

$$I_G(\vec{x})(i) = \begin{cases} \max\{\vec{x}(j), \vec{x}(k)\} & \text{if } i \text{ is a max vertex with successors } j,k \\ \min\{\vec{x}(j), \vec{x}(k)\} & \text{if } i \text{ is a min vertex with successors } j,k \\ \frac{1}{2}\vec{x}(j) + \frac{1}{2}\vec{x}(k) & \text{if } i \text{ is a average vertex with successors } j,k \\ 0/1 & \text{if } i \text{ is the zero/one sink} \end{cases}$$

The vector $\vec{x}$ is *stable* for G F if for all i, $I_G(\vec{x})(i) = \vec{x}(i)$. If $\vec{v}$ is the vector of optimal values, $\vec{v}(i) = v_{opt}(i)$ for all i, then $\vec{v}$ is *stable* for G. In general, there can be more than one stable vector. However, for every game G, there is a corresponding SSG, G' such that G' has a unique stable vector and moreover, G has optimal value greater than $1/2$ if and only if G' has optimal value greater than $1/2$, [13, 23].

Definition 2. *Let G be an n node SSG. The m-*stopping game *G' corresponding to G is a simple stochastic game with $n + mn$ nodes constructed such that any play eventually halts at a terminal node. For each vertex i in the original graph of G, G' has a network of average nodes structured as in Fig. 1. Every in-edge to i in G is replaced by an in-edge to e_{mi} in G'.*

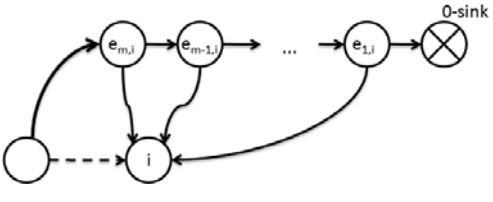


Fig. 1. Average node network added to every node i of original game G

Theorem 3. *(Condon '92, Shapley 53') If G is an m-*stopping game *(for any m), then G has a unique stable vector $\vec{x}$.*

Theorem 3 implies that finding the optimal values for a stopping game G is equivalent to finding a stable vector for I_G. This characterization of optimal values will be very useful to us since it allows us to express the statement "$\vec{v}$ is the vector of optimal values" in terms of very low level equalities and sums.

Theorem 4. *(Condon '92, Shapley 53') For every G there exists a constant c such that the cn-*stopping game *G' has value $> \frac{1}{2}$ if and only iff G has value $> \frac{1}{2}$.*

Formulas and Proof Systems. We will work with the propositional sequent calculus, PK. (See [19] for details.) The *size* of a PK proof is the sum of the sizes of all formulas occurring in the proof. The *depth* of a formula is the number of alternations of OR and ANDs. A formula is Σ_k^+ (Π_k^+) if it has depth $k+1$, where the top connective is OR (AND), and the bottom connective has constant size; a formula is Δ_k^+ if it can be written both as a Σ_k^+ and as a Π_k^+ formula. A depth-k PK proof is a PK proof where every formula in the proof has depth at most k. It is important to note that depth-k PK proofs are the same as depth-$k+1$ Frege proofs in other non-sequent style axiomatizations. We will present PK proofs where every formula in the proof is a Δ_k^+ formula, for some k. Such proofs translate into depth $k-1$ PK proofs, or equivalently, into depth k Frege proofs (in non sequent-style proof systems).

Definition 3. *A proof system $\mathcal{P}$ is automatizable if there exists an algorithm A such that for all unsatisfiable formulas f $A(f)$ returns a $\mathcal{P}$-proof of f, and the runtime of A on f is polynomial in the size of the smallest $\mathcal{P}$-proof of f. $\mathcal{P}$ is weakly automatizable if there exists a proof system that polynomially simulates $\mathcal{P}$ and that is automatizable.*

Definition 4. *Let $F = A(\vec{x},\vec{z}) \wedge B(\vec{y},\vec{z})$ be an unsatisfiable formula. An algorithm C is an interpolant for F if for all $\alpha \in \{0,1\}^{|\vec{z}|}$, $C(\alpha) = 1$ implies $A(\vec{x},\alpha)$ is unsatisfiable, and $C(\alpha) = 0$ implies $B(\vec{y},\alpha)$ is unsatisfiable. A proof system $\mathcal{P}$ admits feasible interpolation if for all unsatisfiable formulas $F = A(\vec{x},\vec{z}) \wedge B(\vec{y},\vec{z})$, there exists an algorithm C that is an interpolant for F, and the runtime of C is polynomial in the size of the shortest $\mathcal{P}$-proof of F. Note that, if $\mathcal{P}$ is weakly automatizable then $\mathcal{P}$ has feasible interpolation.*

Definition 5. *Let G be a n node graph with constant out-degree 2 where each node is labeled with an integer x_i, not all zero. The formula $IGOP(G)$ intuitively says that there exists some i with a nonzero label at least as large as that of its children. Formally, if $\mathbf{x}(i)$ is a set of variables representing the value x_i in binary, IGOP(G) is the formula $\bigvee_i \mathbf{x}(i) \neq 0 \longrightarrow \bigvee_i [\mathbf{x}(i) > 0] \wedge [\mathbf{x}(i) \geq \mathbf{x}(c_1(i))] \wedge [\mathbf{x}(i) \geq \mathbf{x}(c_2(i))]$ where $c_1(i)$ and $c_2(i)$ are the children of i and $\geq$ denotes a standard Δ_2^+ formula for comparison. Note that $[\mathbf{x}(i) \geq \mathbf{x}(c_1(i))] \wedge [\mathbf{x}(i) \geq \mathbf{x}(c_2(i))]$ is NC^0 of Δ_2^+ which can be written in Δ_2^+ so that $IGOP(G)$ is in Σ_2^+ formula.*

Definition 6. *The formula Max-Node$_n$ intuitively expresses that in any set of n integers, there exists a maximal element. Formally, let $\mathbf{x}(i)$ be the set of variables represending the value x_i. Max-Node$_n$ $= \bigvee_i \bigwedge_{j \neq i} [\mathbf{x}(i) \geq \mathbf{x}(j)]$.*

For our purposes, we'd like to apply the IGOP principle to graphs where the nodes are labeled with differences of integers. This does not inherently add to the depth $IGOP(G)$ since we will develop a Δ_2^+ circuit to compare difference (see Sect. 4).

Definition 7. *Let Σ_2^+-Frege+IGOP denote the Σ_2^+-Frege proof system augmented with the following axiom schema for IGOP. Let $\mathbf{d}(i)$ be shorthand for the difference $|\mathbf{x}(i) - \mathbf{y}(i)|$*

$$\bigvee_i [\mathbf{x}(i) \neq \mathbf{y}(i)] \longrightarrow \bigvee_i [\mathbf{d}(i) > 0] \wedge [\mathbf{d}(i)) \geq \mathbf{d}(c_1(i))] \wedge [\mathbf{d}(i) \geq \mathbf{d}(c_2(i))]$$

which states intuitively that if $\mathbf{x}$ and $\mathbf{y}$ are not equal for every i then there exists an i where the difference $\mathbf{d}(i)$ is nonzero at least as the value of $\mathbf{d}$ at the children of i.

We will show that *IGOP* has a Σ_3 proof (via Max-Node$_n$), Σ_2^+-Frege+*IGOP* is a special case of Σ_3-Frege.

3 Main Result

In this section we prove our main theorem, showing that if Σ_3-Frege admits feasible interpolation, then SSG can be solved in polynomial time.

Theorem 5. *If Σ_2^+-Frege+IGOP has feasible interpolation, then SSG is in P.*

Corollary 1. *If Σ_3-Frege has feasible interpolation, then SSG is in P.*

The main idea behind the reduction is as follows. Given an SSG, G, we first construct an m-stopping game G', where G' has value greater than $1/2$ if and only if G has value greater than $1/2$. This implies that the following statement is unsatisfiable.

$$F_{G'} = \left(I_{G'}(\vec{x}) = \vec{x} \wedge \vec{x}(0) > \frac{1}{2}\right) \bigwedge \left(I_{G'}(\vec{w}) = \vec{w} \wedge \vec{w}(0) \leq \frac{1}{2}\right),$$

where each $\vec{x}(i)$ and $\vec{w}(i)$ is an integer represented by a length N binary string where N is $O(n)$.

For every stopping game G' we will prove that $F_{G'}$ has a polynomial-sized Σ_3-Frege refutation. Thus if Σ_3-Frege has feasible interpolation, then the interpolant for F'_G solves the SSG-value problem for G. In order to provide a short Σ_3-Frege refutation of F'_G, it suffices to provide a polynomial-sized Σ_3-Frege proof of $Uniqueness(G) =$ "$I_G(\vec{x}) = \vec{x}, I_G(\vec{w}) = \vec{w} \longrightarrow \vec{x} = \vec{w}$".

To see this, note that if $Uniqueness(G)$ has a short Σ_3-Frege proof, then: (1) From $I_G(\vec{x}) = \vec{x}$ and $I_G(\vec{w}) = \vec{w}$, we can derive $\vec{x} = \vec{w}$; (2) Secondly, from $\vec{x} = \vec{w}$, we can derive $\vec{x}(0) = \vec{w}(0)$; and finally (3) From $\vec{x}(0) = \vec{w}(0)$, we can derive $\vec{x}(0) \leq \frac{1}{2}$, which contradicts F_G.

Theorem 6. *For any stopping simple stochastic game G, the statement $Uniqueness(G)$ can be proved in Σ_2^+-Frege+IGOP.*

We first present a proof of Theorem 3, the uniqueness theorem for SSGs. The remainder of our paper will focus on showing that this can be formalized in depth-3 Frege.

Proof. Theorem 3
Let G' be a m-stopping game. Suppose that $\vec{w}$ and $\vec{x}$ are stable for G' and $\vec{x} \neq \vec{w}$. Denote $\vec{d}$ the difference vector, $\vec{d} = |\vec{w} - \vec{x}|$. Let i be a node such that $\vec{d}(i) > 0$. The stopping game structure induces a discount on the original. If node i had children j and k in the original game, then in any stable solution $\vec{v}$ for G' i's children are nodes with the values $\left(1 - \frac{1}{2^m}\right)\vec{v}(j)$ and $\left(1 - \frac{1}{2^m}\right)\vec{v}(k)$. This implies that $\vec{d}(i) = \left(1 - \frac{1}{2^m}\right)(F(\vec{x}(j), \vec{w}(k)) - F(\vec{w}(j), \vec{w}(k)))$ where $F \in \{\min, \max, \text{ave}\}$, and thus at least one of $\vec{d}(j)$ or $\vec{d}(k)$ is strictly greater than $\vec{d}(i)$. If we choose i to be a node where $\vec{d}(i) \geq \vec{d}(j)$ and $\vec{d}(i) \geq \vec{d}(k)$ then we reach a contradiction. (Note that such an i exists since choosing i that maximizes $\vec{d}(i)$ satisfies these conditions.)

4 Arithmetic Formulas

We will represent integers using two's complement binary notation. In two's complement a set of variables $z_1 z_2 \ldots z_n$ has the value $z = -2^{n-1} \cdot z_1 + 2^{n-2} \cdot z_2 + \ldots + 2^1 \cdot z_{n-1} + 2^0 \cdot z_n$. To prevent overflow, we will also pad each integer by a constant k where k is the largest number of summands used in any operation so that $z_1 z_2 \ldots z_n \rightarrow z_1^k z_2 z_3 \ldots z_n$.

Let $\vec{x}$ be a vector that satisfies $\vec{x} = I_G(\vec{x})$ for some game G. Recall that in any SSG, the values can be written as $\frac{a}{b}$ where $b = O(2^n)$. After normalizing to a common denominator D and padding by k, each $x(i)$ will be expressed as a set of N binary variables $x_1 x_2 \ldots x_N$ so that $x(i) = \frac{1}{D}\left(-2^{N-1} \cdot x_1 + 2^{N-2} \cdot x_2 + \ldots + 2^1 \cdot x_{N-1} + 2^0\right)$

For the remainder of the paper, $[F(x)]$ will be used to represent the Δ_2^+ logical formula that is equivalent to the mathematical formula, $F(x)$. To simplify notation, only one set of brackets will be used when any nesting occurs so that $[[F(x)] \leftrightarrow [G(x)]]$ is just $[F(x) \leftrightarrow G(x)]$. Boldface variables $\mathbf{x}$ denote an array of $k \times N$ array of integers and $\mathbf{x}(i)$ is the i^{th} row. Normal font variables x represent a single integer expressed with N binary variables. We can represent the following formulas as Δ_2^+ formulas. See [19] for precise definitions.

- $[x \leftrightarrow y]$ is a formula that is true if and only if $\forall i\,(x_i \leftrightarrow y_i)$
- $[\mathbf{x}(1) + \ldots + \mathbf{x}(k) - \mathbf{y}(1) \ldots - \mathbf{y}(l)]$ represents a set of N formulas, such that the i^{th} formula is true iff the i^{th} bit of $\mathbf{x}(1) + \ldots + \mathbf{x}(k) - \mathbf{y}(1) \ldots - \mathbf{y}(k)$ is 1.
- $[\mathbf{x}(1) + \ldots + \mathbf{x}(k) - \mathbf{y}(1) \ldots - \mathbf{y}(l)] \geq z]$ is true if and only if the sum on the left is $\geq z$. And similarly $[\mathbf{x}(1) + \ldots + \mathbf{x}(k) - \mathbf{y}(1) \ldots - \mathbf{y}(l)] < z]$ is true iff the sum is less than z
- $[\max(x,y)]$ represents a set of formulas where the i^{th} formula is x_i iff $x = max(x,y)$ and $[\min(x,y)]$ represents the set of formulas where the i^{th} formula is x_i iff $x = min(x,y)$.

The technical difficulty lies in defining formulas, $F^k_{i \in 1,\ldots,N}$ for calculating all of the bits in the sum of k $N-$bit integers for any constant k. Given $\{F_i^k\}$ the remainder of the formulas are fairly straightforward. For arbitrary k, [4] shows that there exists a Δ_2^+ formula for calculating whether the sum of k integers generates an overflow. We extend their formula so that it calculates all bits, F_i^k, of the sum. The following lemmas comprise the bulk of the work of our proof. (Proofs are in [19].)

Lemma 1 (Substitution)
For any Δ_2^+ formula $F(x_1,x_2,...x_N,z...)$ there exists a short PK proof of the following sequent, where all formulas in the proof are Δ_2^+ formulas. $[x \leftrightarrow y] \longrightarrow [F(x_1,x_2,...x_N,z...) \leftrightarrow F(y_1,y_2,...y_N,z...)]$

Lemma 2. *For N bit integers x, y the following sequents have short PK proofs where all formulas are Δ_2^+: (1)* $[x \geq y] \longrightarrow [\max(x,y) = x]$, *and (2)* $[x \geq y] \longrightarrow [\min(x,y) = y]$.

Lemma 3 (Nested Addition). *For N bit integers a, b, c, and array of constant k $N-$bit integers* $\mathbf{x}$ *there exists short PK proofs of the following where all formulas are Δ_2^+:*

$$[a+b \leftrightarrow c] \longrightarrow [\mathbf{x}(1)+...+\mathbf{x}(k)+a+b \leftrightarrow \mathbf{x}(1)+...+\mathbf{x}(k)+c]$$
$$[a-b \leftrightarrow c] \longrightarrow [\mathbf{x}(1)+...+\mathbf{x}(k)+a+\overline{b}+1 \leftrightarrow \mathbf{x}(1)+...+\mathbf{x}(k)+c]$$

Corollary 2. *Given* $[x+a+b \leftrightarrow 1^N]$ *we can derive* $[\overline{x} \leftrightarrow a+b]$ *for any integers x, a, b. In particular,* $[x \leftrightarrow a-b] \longrightarrow [\overline{x} \leftrightarrow \overline{a}+b]$, *and* $[x \leftrightarrow a+b] \longrightarrow [\overline{x} \leftrightarrow \overline{a}-\overline{b}]$.

Lemma 4. *Let x,y,x',y' be N-bit integers, and $m > 0$, and $\frac{1}{2^m}y$ represents $0^m y_1 y_2 ... y_{N-m}$. Then there exists a short PK proof of the following formula, where all formulas are Δ_2^+:*

$$[x > x'], [x \leftrightarrow y - \frac{1}{2^m}y], [x' \leftrightarrow y' - \frac{1}{2^m}y'] \longrightarrow [x - x' - y + y' < 0]$$

Lemma 5. *Fix an $a \times N$ array* $\mathbf{z}$ *and a $b \times N$ array* $\mathbf{x}$ *where each integer is padded by $k = a+b$ bits. There is a short proof of the following:*

$$[\mathbf{z}(1)+...\mathbf{z}(a) < 0], [\mathbf{x}(1)+...+\mathbf{x}(b) < 0] \longrightarrow [\mathbf{z}(1)+...\mathbf{z}(a)+\mathbf{x}(1)+...\mathbf{x}(b) < 0]$$
$$[\mathbf{z}(1)+...\mathbf{z}(a) \geq 0], [\mathbf{x}(1)+...+\mathbf{x}(b) \geq 0] \longrightarrow [\mathbf{z}(1)+...\mathbf{z}(a)+\mathbf{x}(1)+...\mathbf{x}(b) \geq 0]$$

5 Proof of Uniqueness

Equipped with the above lemmas, we can now prove uniqueness for SSGs in depth-3 Frege. Let $\mathbf{x}$ and $\mathbf{w}$ be arrays of integers as defined in the previous section such that each integer is padded by k bits and $\mathbf{x}(i)$ is the value associated with the i^{th} node of a $m-$stopping game G. We define a formula, Uniqueness(G) on the $N \cdot (n+mn)$ variables defined above which will state that if $\mathbf{x} = I_G(\mathbf{x})$ and $\mathbf{w} = I_G(\mathbf{w})$, then $\mathbf{x}(i) = \mathbf{w}(i)$ for every i. The premise $\mathbf{x} = I_G(\mathbf{x})$ can be stated as follows, using the arithmetic formulas defined in the previous section.

(1) For every max node i, with children e_{mj} and e_{mk}, $[\mathbf{x}(i) \leftrightarrow \max(\mathbf{x}(e_{mj}),\mathbf{x}(e_{mk}))]$
(2) For every min node i, with children e_{mj} and e_{mk}, $[\mathbf{x}(i) \leftrightarrow \min(\mathbf{x}(e_{mj}),\mathbf{x}(e_{mk}))]$
(3) For every average node i, with children e_{mj} and e_{mk}, $[2 \cdot \mathbf{x}(i) \leftrightarrow \mathbf{x}(e_{mj}) + \mathbf{x}(e_{mk})]$
(4) For every new average node of the form e_{ij}, $[\mathbf{x}(e_{ij}) \leftrightarrow \mathbf{x}(j) - \frac{1}{2^i}\mathbf{x}(j)]$.

For any i, we will prove that if $d(i)$ (the absolute value of $x(i) - w(i)$) is nonzero and at least as large as its neighbors, $d(j)$ and $d(k)$, then we can derive a contradiction. Formally, let $M(i)$ be the formula:

$$[\mathbf{x}(i) > \mathbf{w}(i)], \forall s \in \{j,k\}\,([\mathbf{x}(i) - \mathbf{w}(i) - \mathbf{x}(s) + \mathbf{w}(s) \geq 0]),$$
$$\forall s \in \{j,k\}\,([\mathbf{x}(i) - \mathbf{w}(i) - \mathbf{w}(s) + \mathbf{x}(s) \geq 0])$$

We will prove for every i: $M(i), [\mathbf{x} = I_G(\mathbf{x})], [\mathbf{w} = I_G(\mathbf{w})] \longrightarrow \bot$. Finally we can use the IGOP axiom schema to prove that there exists an i such that $M(i)$ holds which allows us to conclude: $[\mathbf{x} = I_G(\mathbf{x})], [\mathbf{w} = I_G(\mathbf{w})] \longrightarrow [\mathbf{x}(1) \leftrightarrow \mathbf{w}(1)], ..., [\mathbf{x}(n) \leftrightarrow \mathbf{w}(n)]$.

Case 1. i is a max node with children j and k (Proof for i is a min node is similar.)
As in proof of Theorem 3, we can further divide Case 1 into four cases depending on the sign of $[\mathbf{x}(j) - \mathbf{x}(k)]$ and $[\mathbf{w}(j) - \mathbf{w}(k)]$. Because of the way we defined comparisons, $[a < b]$ is exactly $\neg[a \geq b]$ so it it's not hard to show that exactly one of four sub cases is true. We show a complete proof for when $[\mathbf{x}(e_{mj}) \geq \mathbf{x}(e_{mk})]$, $[\mathbf{w}(e_{mj}) \geq \mathbf{w}(e_{mk})]$, the remaining three subcases are in [19].

Let $H(\mathbf{x}, \mathbf{w})$ denote the subcase premise $[\mathbf{x}(e_{mj}) \geq \mathbf{x}(e_{mk})]$, $[\mathbf{w}(e_{mj}) \geq \mathbf{w}(e_{mk})]$. We prove for contradiction that $M(i), H(\mathbf{x}, \mathbf{w}), [\mathbf{x} = I_G(\mathbf{x})], [\mathbf{w} = I_G(\mathbf{w})] \longrightarrow \bot$. Using Lemma 2 we can prove $[\mathbf{x}(e_{mj}) \geq \mathbf{x}(e_{mk})] \longrightarrow [\max(\mathbf{x}(e_{mj}), \mathbf{x}(e_{mk})) \longrightarrow \mathbf{x}(e_{mj})]$. And by weakening the LHS we have $H(\mathbf{x}, \mathbf{w}) \longrightarrow [\max(\mathbf{x}(e_{mj}), \mathbf{x}(e_{mk})) \leftrightarrow \mathbf{x}(e_{mj})]$. From the statements forming $[\mathbf{x} = I_G(\mathbf{x})]$ above we can apply $[\mathbf{x}(i) \leftrightarrow \max(\mathbf{x}(e_{mj}), \mathbf{x}(e_{mk}))]$ and $[\mathbf{x}(e_{mj}) \leftrightarrow \mathbf{x}(j) - \frac{1}{2^m}\mathbf{x}(j)]$ to prove

$(1) : [\mathbf{x} = I_G(\mathbf{x})], H(\mathbf{x}, \mathbf{w}) \longrightarrow [\mathbf{x}(i) \leftrightarrow \mathbf{x}(j) - \frac{1}{2^m}\mathbf{x}(j)]$ and $(2) : [\mathbf{w} = I_G(\mathbf{w})], H(\mathbf{x}, \mathbf{w}) \longrightarrow [\mathbf{w}(i) \leftrightarrow \mathbf{w}(j) - \frac{1}{2^m}\mathbf{w}(j)]$. The next step is to prove the following sequent, which follows from Lemma 4:

$$(3) : [\mathbf{x}(i) > \mathbf{w}(i)], [\mathbf{x}(i) \leftrightarrow \mathbf{x}(j) - \frac{1}{2^m}\mathbf{x}(j)], [\mathbf{w}(i) \leftrightarrow \mathbf{w}(j) - \frac{1}{2^m}\mathbf{w}(j)] \longrightarrow [\mathbf{x}(i) - \mathbf{w}(i) - \mathbf{x}(j) + \mathbf{w}(j) < 0]$$

Finally to put things together, we apply cut on (1),(2) and (3) to obtain:

$$(4) : M(i), H(\mathbf{x}, \mathbf{w}), [\mathbf{x} = I_G(\mathbf{x})], [\mathbf{w} = I_G(\mathbf{w})] \longrightarrow [\mathbf{x}(i) - \mathbf{w}(i) - \mathbf{x}(j) + \mathbf{w}(j) < 0]$$

which is the same as $M(i), H(\mathbf{x}, \mathbf{w}), [\mathbf{x} = I_G(\mathbf{x})], [\mathbf{w} = I_G(\mathbf{w})] \longrightarrow \bot$.

Case 2. For an average node i, it is easy to show that $\mathbf{x}(j) - \mathbf{w}(j) > \mathbf{x}(e_{mj}) - \mathbf{w}(e_{mj})$ and $\mathbf{x}(k) - \mathbf{w}(k) > \mathbf{x}(e_{mk}) - \mathbf{w}(e_{mk})$ just as we did in the max node case using Lemma 4. By Lemma 5 we can combine the two negative sums above to the larger negative sum

$$[\mathbf{x}(e_{mj}) - \mathbf{w}(e_{mj}) - \mathbf{x}(j) + \mathbf{w}(j) + \mathbf{x}(e_{mk}) - \mathbf{w}(e_{mk}) - \mathbf{x}(k) + \mathbf{w}(k) < 0]$$

The goal is to make the substitutions that relate this inequality back to $\mathbf{x}(i) - \mathbf{w}(i)$. We can always add zero to any sum without changing its value, and due to Lemma 3 we can also add a pair of integers that sum to zero. Using this trick we will replace the terms containing e_{mj} and e_{mk} with terms containing i.

When i is an average node, $2\mathbf{x}(i) = [\mathbf{x}(e_{mj}) + \mathbf{x}(e_{mk})]$. Recall that $2x(i)$ does not represent a sum but a new set of binary variables $\mathbf{x}(i)_2 ... \mathbf{x}(i)_N 0$. This is very convenient as we can now freely use Lemma 3 to add $0 = 2\mathbf{x}(i) - 2\mathbf{x}(i)$ and $0 = 2\mathbf{w}(i) - 2\mathbf{w}(i)$ to the negative sum above to prove that

$$[2\mathbf{x}(i) - 2\mathbf{x}(i) + 2\mathbf{w}(i) - 2\mathbf{w}(i) + \mathbf{x}(e_{mj}) - \mathbf{w}(e_{mj}) - \mathbf{x}(j) + \mathbf{w}(j) + \mathbf{x}(e_{mk}) - \mathbf{w}(e_{mk}) - \mathbf{x}(k) + \mathbf{w}(k) < 0]$$

In order to cancel out the e_{mj} and e_{mk} terms, we need to make the following four substitutions:

- $2\mathbf{x}(i) \rightarrow [\mathbf{x}(i) + \mathbf{x}(i)]$
- $2\mathbf{x}(i) \rightarrow [\mathbf{x}(e_{mj}) - \mathbf{x}(e_{mk})]$
- $2\mathbf{w}(i) \rightarrow [\mathbf{w}(e_{mj}) + \mathbf{w}(e_{mk})]$
- $2\mathbf{w}(i) \rightarrow [\mathbf{w}(i) - \mathbf{w}(i)]$

After removing the zero terms, we finally arrive at

$$[\mathbf{x} = I_G(\mathbf{x})], [\mathbf{w} = I_G(\mathbf{w})] \longrightarrow [\mathbf{x}(i) + \mathbf{x}(i) - \mathbf{w}(i) - \mathbf{w}(i) - \mathbf{x}(j) + \mathbf{w}(j) - \mathbf{x}(k) + \mathbf{w}(k) < 0]$$

On the other hand, if $M(i)$ is true then both $[\mathbf{x}(i) - \mathbf{w}(i) - \mathbf{x}(j) + \mathbf{w}(j) \geq 0]$ and $[\mathbf{x}(i) - \mathbf{w}(i) - \mathbf{x}(k) + \mathbf{w}(k) \geq 0]$ are true. Applying Lemma 5 to the two positive sums we can derive that the same sum must be positive.

$$M(i) \longrightarrow [\mathbf{x}(i) - \mathbf{w}(i) - \mathbf{x}(j) + \mathbf{w}(j) + \mathbf{x}(i) - \mathbf{w}(i) - \mathbf{x}(k) + \mathbf{w}(k) \geq 0]$$

As with the max/min nodes, we have shown that for all average nodes i from $M(i), [\mathbf{x} = I_G(\mathbf{x})], [\mathbf{w} = I_G(\mathbf{w})]$ we can derive a contradiction. Thus for any i, $M(i) \longrightarrow Uniqueness(G)$. Using the *IGOP* axiom schema, we can immediately derive $\bigvee_i M(i)$ from which we can conclude $Uniqueness(G)$.

5.1 Proving Max-Node$_n$

To achieve the Σ_3 result, it remains to prove Max-Node$_n$. Suppose that $x(i) - w(i)$ is maximal among the first $n-1$ nodes, and $x(n) \geq w(n)$ ($w(n) > x(n)$ follows by symmetry). Either (1) $[x(i) - w(i) \geq x(n) - w(n)]$ or (2) $[x(i) - w(i) < x(n) - w(n)]$. If (1) is true then $x(i) - w(i)$ remains maximal. If (2) is true then for each j, $[x(n) - w(n) - x(j) + w(j) < 0]$ would imply that $[x(i) - w(i) - x(j) + w(j)]$ as well (Lemma 5 followed by Lemma 3). Since we know the latter is not true, we can prove that for all j, $[x(n) - w(n) \geq x(j) - w(j)]$. In either case, we've proven Max-Node$_n$, by Σ_3^+ proofs.

5.2 Removing the Bottom Fan-In

So far, we have shown how to prove the uniqueness property with Σ_3^+ proofs, or with depth Σ_2^+ proofs plus the IGOP principle. With one additional idea, we can remove the small bottom fanin. Recall that the bottom fanin was only a constant – suppose the largest bottom fanin is c. We can replace our original formula, F (in our case the formula asserting uniqueness), by a new formula, F', as follows. First, we introduce (polynomially many) new variables, one associated with each conjunction of at most c literals. The new uniqueness formula F' asserts that G implies F, where G is a conjunction of new clauses (called "extension axioms") asserting that each new variable is equivalent to its associated conjunction. The new formula F' is still of polynomial size, and furthermore it is a tautology if and only if F is a tautology. It is a standard argument to show that if F has a PK proof where all formulas in the proof are Δ_k^+, then F' has a PK proof where all formulas in the proof are Δ_k. To complete the proof of Theorem 5, we replace our original uniqueness formula, F, by the new uniqueness formula, F'. Since F has efficient Σ_3^+-Frege proofs, F' has efficient Σ_3-Frege proofs.

6 Consequences and Open Problems

Tightness of our Result. In this paper, we have shown that if depth-2 Frege systems plus the Integer-Valued Graph Ordering Principle (IGOP) is automatizable, then SSGs are in P. This can be viewed as either an "easyness" result for SSGs or a hardness result for automatizability but more importantly this paper and [4] open up new ways of approaching both problems. Of course, any result in the converse direction would further strengthen this connection. An interesting question is whether there exists an efficient depth-2 proof of uniqueness for SSGs (this would follow immediately if IGOP could be proven in depth-2). However, we conjecture that IGOP does not have efficient depth-2 proofs. Our conjecture is based on the fact that this principle is simply an instance of the usual GOP but where the variables are replaced by Δ_2 formulas. GOP is a well-known propositional tautology as it is the classic (and only) example of a family of formulas that exhibits the tightness of size-width tradeoffs lower bounds for resolution in that it has efficient proofs but requires $O(\sqrt{n})$ clause width [8,16]. Our conjecture just says that this phenomena continues to hold when we scale GOP up to higher depth. In fact we make the stronger conjecture that our depth-3 proof of Uniqueness for SSGs is tight. Moreover, we conjecture that the depth-2 proofs of totality for mean payoff games [4] are also tight:

Conjecture 1. (1)Uniqueness for SSGs does not have polynomial-size depth-2 Frege proofs; (2)Totality of MPGs does not have polynomial-size depth-1 (resolution) proofs.

If Conjecture (1) is true, it would imply that SSGs cannot be efficiently reduced (in the type-2 setting) to mean payoff games. This follows from the main result of Atserias and Maneva [4], together with a theorem due to Morioka and Buresh-Oppenheim [10] (Theorem 10 in their paper).

More generally, it is interesting to study low-depth, efficient reductions between statements of totality of various game-theoretic problems. Just as we study the relative strength of various search problems, it is natural to consider the relative proof theoretic strength of the underlying principles, by studying efficient reductions between them in standard low-depth propositional proof systems. The study of total search problems in general, and their corresponding proofs of totality are widely studied in proof complexity. Indeed, the strength of weak systems of arithmetic are classified in terms of what kinds of search problems are provably total in the theory. (For every search problem there is a first order formula expressing that the search problem is total, and conversely for every statement of the form $\exists x A(x,y)$, there is a corresponding total search problem.) PLS figures prominently in this research, as it is precisely the class of total functions that are provably total in the theory TV^1. It would be interesting to expand this work to include the study of statements of totality for various game-theory problems, as well as to study low-depth reductions between such statements. In particular, consider the propositional statements expressing the following principles: (a) totality for SSGs, (b) totality of mean payoff games, (c) the iteration principle (underlying PLS), (d) the pigeonhole principle (underlying PPAD). Which ones can and cannot be reduced in low-depth to one another? Answers to these questions would likely yield an understanding of the relative strengths of the corresponding search classes: SSG, MPG, PLS, and PPAD.

References

1. Alekhnovich, M., Razborov, A.: Resolution is not automatizable unless w[p] is tractable. In: IEEE Symposium on Foundations of Computer Science (2001)
2. Andersson, D., Miltersen, P.B.: The complexity of solving stochastic games on graphs. In: Dong, Y., Du, D.-Z., Ibarra, O. (eds.) ISAAC 2009. LNCS, vol. 5878, pp. 112–121. Springer, Heidelberg (2009)
3. Atserias, A., Bonet, M.: On the automatizability of resolution and related propositional proof systems. Information and Computation 189(2), 182–201 (2004)
4. Atserias, A., Maneva, E.: Mean-payoff games and propositional proofs. In: Abramsky, S., Gavoille, C., Kirchner, C., Meyer auf der Heide, F., Spirakis, P.G. (eds.) ICALP 2010. LNCS, vol. 6198, pp. 102–113. Springer, Heidelberg (2010)
5. Bjorklund, H., Vorobyov, S.: Combinatorial structure and randomized subexponential algorithms for infinite games. Theoretical Computer Science 349, 347–360 (2005)
6. Blum, M., Juba, B., Williams, R.: Non-monotone behaviors in min/max/avg circuits and their relationship to simple stochastic games
7. Bonet, M., Domingo, C., Gavalda, R., Maciel, A., Pitassi, T.: No feasible interpolation or automatization for AC^0-Frege proof systems (1998) (manuscript)
8. Bonet, M.L., Galesi, N.: optimality of size-width tradeoffs for resolution. Computational Complexity 10, 261–276 (2001)
9. Bonet, M.L., Pitassi, T., Raz, R.: On interpolation and automatization for frege systems. SIAM J. Comput. 29, 1939–1967 (2000)
10. Buresh-Oppenheim, J., Morioka, T.: Relativized np search problems and propositional proof systems. In: 19th IEEE Conference on Computational Complexity, CCC (2004)
11. Chen, X., Deng, X., Teng, S.H.: Settling the complexity of two-player nash equilibria. Journal of the ACM 56 (2009)
12. Clegg, M., Edmonds, J., Impagliazzo, R.: Using the Gröbner basis algorithm to find proofs of unsatisfiability. In: Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing, Philadelphia, PA, pp. 174–183 (May 1996)
13. Condon, A.: The complexity of stochastic games. Information and Computation 96, 203–224 (1992)
14. Condon, A.: On algorithms for simple stochastic games. In: Advances in Computational Complexity Theory. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 13, pp. 51–73 (1993)
15. Daskalakis, C., Goldberg, P., Papadimtriou, C.H.: The complexity of computing a nash equilibrium. SIAM Journal on Computing 39, 195–259 (2009)
16. Galesi, N., Lauria, M.: Optimality of size-degree tradeoffs for polynomial calculus. ACM Trans. Comput. Logic 12, 4:1–4:22 (2010)
17. Gimbert, H., Horn, F.: Simple stochastic games with few random vertices are easy to solve. In: Amadio, R. (ed.) FOSSACS 2008. LNCS, vol. 4962, pp. 5–19. Springer, Heidelberg (2008)
18. Halman, N.: Simple stochastic games, parity games, mean payoff games and discounted payoff games are all lp-type problems. Algorithmica 49(1), 37–50 (2007)
19. Huang, L., Pitassi, T.: Automatizability and simple stochastic games (2011), http://www.cs.toronto.edu/~leih/ssg_full.pdf
20. Krajicek, K., Pudlak, P.: Some consequences of cryptographic conjectures for S_2^1 and EF. In: Leivant, D. (ed.) LCC 1994. LNCS, vol. 960, pp. 210–220. Springer, Heidelberg (1995)
21. Kumar, V., Tripathi, R.: Algorithmic results in simple stochastic games. Technical Report 855, University of Rochester (2004)

22. Ludwig, W.: A subexponential randomized algorithm for the simple stochastic game problem. Inf. Comput. 117, 151–155 (1995)
23. Shapley, L.S.: Stochastic games. Proceedings of the National Academy of Sciences U.S.A. 39, 1095–1100 (1953)
24. Somla, R.: New algorithms for solving simple stochastic games. Electron. Notes Theor. Comput. Sci. 119(1), 51–65 (2005)

Exponential Lower Bounds for AC^0-Frege Imply Superpolynomial Frege Lower Bounds

Yuval Filmus[1,⋆], Toniann Pitassi[1,⋆], and Rahul Santhanam[2]

[1] Univeristy of Toronto
{yuvalf,toni}@cs.toronto.edu
[2] University of Edinburgh
rsanthan@inf.ed.ac.uk

Abstract. We give a general transformation which turns polynomial-size Frege proofs to subexponential-size AC^0-Frege proofs. This indicates that proving exponential lower bounds for AC^0-Frege is hard, since it is a longstanding open problem to prove super-polynomial lower bounds for Frege. Our construction is optimal for tree-like proofs.

As a consequence of our main result, we are able to shed some light on the question of weak automatizability for bounded-depth Frege systems. First, we present a simpler proof of the results of Bonet et al. [5] showing that under cryptographic assumptions, bounded-depth Frege proofs are not weakly automatizable. Secondly, we show that because our proof is more general, under the right cryptographic assumptions, it could resolve the weak automatizability question for lower depth Frege systems.

1 Introduction

The fundamental question in computational complexity is the P vs. NP question. Though we are very far from resolving this question, over the past few decades we have made substantial progress in understanding why certain approaches, for example diagonalization and the use of combinatorial or algebraic techniques to prove circuit lower bounds, are unlikely to work. Various barriers such as the relativization, natural proofs and algebrization barriers have been formulated to capture the limitations of known techniques, and in turn, this meta-level understanding of complexity lower bound problems has led to developments in areas such as low-level complexity and derandomization.

However, there are still approaches whose power is not well understood, such as those used in proof complexity. Proof complexity was introduced by Cook and Reckhow [8] as a framework within which to study the NP vs. coNP problem. Cook and Reckhow defined *propositional proof systems* in a very general way by insisting only that proofs be verifiable in polynomial time, and showed that the existence of a propositional proof system in which all tautologies have polynomial size proofs is equivalent to NP = coNP. They suggested a program to separate NP and coNP (and thereby P and NP) by showing superpolynomial proof size

⋆ Supported by NSERC.

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 618–629, 2011.

lower bounds for explicit tautologies in progressively stronger proof systems. The hope was that techniques from logic and proof theory could be effective where techniques inspired by recursion theory or combinatorics are not. The fact that the very definition of the P vs. NP question involves the notion of "proof" in a fundamental way makes this hope somewhat plausible.

Indeed, over the past couple of decades, lower bounds have been shown for various natural proof systems [9,3]. However, lower bounds for natural systems such as Frege and Extended Frege still seem out of reach. We seem to have hit a "wall" with proof complexity lower bounds, just as with circuit complexity lower bounds.

To an extent, this reflects the fact that the techniques used for the currently strongest proof complexity lower bounds are adaptations of the techniques used in circuit complexity, and limitations of the circuit complexity techniques carry over to the versions used in proof complexity. There is an informal "mapping" from proof systems to complexity classes, where a proof system Q corresponds to the smallest complexity class C such that the lines of polynomial-sized proofs in Q are functions in C. In this way, Resolution maps to DNFs, Bounded-Depth Frege to non-uniform AC^0, Frege to non-uniform NC^1, and Extended Frege to SIZE($poly$).

In the circuit complexity world, we have lower bounds for explicit functions against DNFs and non-uniform AC^0, but not against non-uniform NC^1 and SIZE($poly$); correspondingly, in the proof complexity setting we have strong lower bounds for Resolution and fairly strong lower bounds for bounded-depth Frege, but no non-trivial lower bounds for Frege and Extended Frege.

The question arises whether this connection between circuit complexity and proof complexity is fundamental or not. No formal connection is known either way — we don't have theorems to the effect that circuit complexity lower bounds yield proof complexity lower bounds, nor any implications in the reverse direction. Moreover, barriers such as the natural proofs barrier don't seem to apply to proof complexity. This suggests that perhaps completely different techniques, say from proof theory or finite model theory, might help in showing proof complexity lower bounds. Is there a sense in which there are barriers to making progress in proof complexity? Formulating and understanding such barriers would not only guide us towards the "right" techniques, but might have collateral benefits as well, as in the circuit complexity setting.

In this paper, we shed some light on these questions. We draw a connection between two fundamental lower bound questions in proof complexity. The first question is to prove strong lower bounds for bounded-depth Frege. Superpolynomial lower bounds are known for this proof system, but there aren't any lower bounds known that are purely exponential, i.e., $2^{\Omega(n^c)}$ where the constant c doesn't depend on the depth of lines in the proof (the best known lower bound is $\Omega(2^{n^{5^{-d}}})$ [3]). The second question, which is perhaps the major open question in proof complexity, is to obtain superpolynomial lower bounds for Frege. This question is believed to be very hard — it is non-trivial even to think of plausible candidate tautologies for which superpolynomial lower bounds are believed to

hold [4]. We show that progress on the first question would lead to progress on the second, by giving a general simulation of polynomial size Frege proofs by subexponential size bounded-depth Frege proofs. More precisely, we show that even a $2^{n^{\omega(1/d)}}$ proof size lower bound for proving CNF tautologies in depth d Frege would translate to a superpolynomial proof size lower bound for Frege.

The proof of this connection is inspired by a result in circuit complexity, further strengthening the "mapping" between proof complexity and circuit complexity. The circuit complexity result we draw inspiration from is that NC^1 can be simulated by bounded-depth circuits with sub-exponential size [1]. The standard proof of this goes via a divide-and-conquer technique. We use a similar technique in our context, however our task is made harder in a sense by the fact that we need to reason within bounded-depth Frege about equivalence of various alternative representations of a function. The technical heart of our proof involves such reasoning.

Our result is also relevant to algorithmic analysis, which is another major motivation for studying proof complexity. A propositional proof system can be thought of as a non-deterministic algorithm for deciding if a formula is a tautology or not. Proof systems such as bounded-depth Frege and Frege provide particularly simple and natural examples of such algorithms. Indeed, many of the algorithms and heuristics used in practice for solving SAT, such as DPLL and Clause Learning, arise from *determinizing* the non-deterministic algorithm corresponding to some natural proof system. Thus lower bounds for proof systems give us information on the performance of algorithms used in practice.

Algorithmic analysis would appear to be a simpler question than proving complexity lower bounds, since a complexity lower bound is a statement about *any* possible algorithm for a problem, while algorithmic analysis deals with specific algorithms. There are somewhat artificial algorithms such as Levin's optimal algorithm for SAT whose analysis is just as difficult as proving complexity lower bounds. However, one might expect that for more natural algorithms, such as those corresponding to natural propositional proof systems, this is not the case. Our current lack of progress in proving proof complexity lower bounds indicates that there might be barriers even in algorithmic analysis of natural algorithms. Our main result here can be interpreted as saying that the algorithmic analysis question for the algorithm corresponding to bounded-depth Frege is as hard as the question for the algorithm corresponding to Frege (which in some sense is a more sophisticated algorithm). In general, it would be useful to have a theory of algorithmic analysis which gives us information about the relative difficulty of analyzing various natural algorithms. We make a small step in this direction in the setting of non-deterministic algorithms for TAUT.

There are a couple of interesting byproducts of our main result. First, we are able to prove *tight* bounds for proving certain explicit tautologies in treelike bounded-depth Frege. Lower bounds for the tautologies we consider were already shown by Krajíček [10]. We give corresponding upper bounds as a corollary of our simulation of Frege by bounded-depth Frege.

Second, we address the question of *weak automatizability* for bounded-depth Frege systems. A proof system $\mathcal{P}$ is weakly automatizable if there is an algorithm that on input f and a number r in unary, can distinguish the case where f is not a tautology from the case where f has a $\mathcal{P}$-proof of size at most r. Despite considerable effort, the question of whether low depth proof systems are weakly automatizable is unresolved. Bonet, Domingo, Gavaldá, Maciel and Pitassi [5] show that depth k Frege systems are not weakly automatizable under a cryptographic assumption, but their result breaks down for small k (less than 6). We use our main result to re-derive their main theorem. Our proof is cleaner and simpler than theirs, and we show that it could potentially resolve the weak automatizability question for lower depth Frege systems than what is currently known.

1.1 Proof Overview

Suppose that P is a Frege proof of some formula f. We want to simulate P by a subexponential-size depth d Frege proof of f. The high-level idea behind the simulation is to replace every formula in the proof by its equivalent depth d (subexponential-size) *flattened* formula, and then to show that if C was derived by a rule from A and B, then the flattened version of C can be efficiently derived from the flattened versions of A and B.

We can assume without loss of generality that all formulas f in the proof are balanced (Reckhow's theorem). We first review the translation of a balanced formula f to its flattened form. We say that a formula has logical depth at most d if the depth of the binary tree representing the formula is at most d. Suppose that we want to replace f, of size n and logical depth $\log n$, by a depth 3 formula. The idea is to view f as consisting of two layers: the top layer is a formula, f_1, of height $(\log n)/2$, and the bottom layer consists of $2^{(\log n)/2} = \sqrt{n}$ subformulas, $g_1, \ldots, g_{\sqrt{n}}$, each of height $(\log n)/2$. Since f_1 has height $(\log n)/2$, it has at most $\sqrt{n}$ inputs, and thus can be written as either a CNF or a DNF formula (of its inputs) of size $\sqrt{n}2^{\sqrt{n}}$. Similarly, each formula in the bottom layer can be written as either a CNF or a DNF formula of size $\sqrt{n}2^{\sqrt{n}}$. Writing f_1 as a CNF formula, and writing all formulas g_j in the bottom layer as DNF formulas, we obtain a new formula for f of depth 3 and total size $O(n2^{2\sqrt{n}})$. (The depth is 3 because we can merge the middle two AND layers.) In a similar manner, we can replace any formula f, of size n and logical depth $\log n$, by a depth $d+2$ formula: Now we break up f into d equally-spaced layers, each of size $(\log n)/d$. Again, we write the formula at the top layer as a CNF formula, the formulas at the next layer as DNF formulas, and so on. This gives a formula of depth $2(d+1)$ and total size $O(n2^{dn^{1/d}})$, but since we alternated CNF/DNFs, we can collapse every other layer to obtain a new flattened formula of depth $2(d+1) - d = d + 2$.

Now that we have flattened translations of each formula in P, it remains to fill in the proof, to show that the flattened versions can be derived from one another. In order to carry this out, we define a more general procedure for flattening a formula as follows. Let $\boldsymbol{d}$ be any *depth vector* – i.e., it is a sequence of increasing numbers, where each number in the sequence is between 1 and $\log n$. Then from

a balanced formula f of size n and logical depth $\log n$, $\boldsymbol{d}$ defines a new flattened formula of depth $|\boldsymbol{d}|+2$: we break f up into $|\boldsymbol{d}|$ many levels, where now instead of the levels being equally spaced, the breakpoints are specified by $\boldsymbol{d}$. For example, if $\boldsymbol{d} = (4, 12)$ and f has depth 20, then the $\boldsymbol{d}$-flattened version of f will have 3 levels, the top level containing levels 1 through 3, the second level 4 through 11, and the third level 12 through 20. Our main lemma shows that for any balanced formula f and any two depth vectors $\boldsymbol{d}_1$, $\boldsymbol{d}_2$, there are efficient low-depth Frege proofs showing that the $\boldsymbol{d}_1$-flattened version of f is equivalent to the $\boldsymbol{d}_2$-flattened version of f. This main lemma will then allow us to prove that for any rule of our proof system, the flattened versions of the antecedent formulas derive the flattened version of the consequent formula.

2 Proof Systems

We will work with the propositional sequent calculus, PK. In the fundamental work of Cook and Reckhow [8], many reasonable formulations of Frege systems (including *all* PK-like systems) were studied and shown to be polynomially equivalent; we work with PK for convenience, but any other Frege system will do.

Each line in a PK proof is a *sequent* of the form $A_1, \ldots, A_k \longrightarrow B_1, \ldots, B_m$ where $\longrightarrow$ is a new symbol, and A_i, B_j are formulas. The intended meaning is that the conjunction of the A_i's implies the disjunction of the B_j's.

A PK *proof* of $\longrightarrow f$ is a sequence of sequents, such that each sequent is either an instance of the axiom $A \longrightarrow A$, or follows from previous sequents from one of the inference rules, and such that the final sequent is $\longrightarrow f$.

The rules of PK are of three types: (i) the structural rules, (ii) the logical rules, and (iii) the cut rule.

The structural rules are weakening, contraction and permutation.

The logical rules allow us to introduce each connective on both the left side and the right side.

The final rule is the cut rule, which allows us to derive $\Gamma \longrightarrow \Delta$ from $A, \Gamma \longrightarrow \Delta$ and $\Gamma \longrightarrow A, \Delta$. We call formula A the *cut formula*.

A full description of PK is found in the appendix.

The *size* of a PK proof is the sum of the sizes of all formulas occurring in the proof.

The *logical depth* of a formula φ, denoted by $\mathrm{ldp}(\varphi)$, is the depth of the formula when considered as a binary tree. For example, $(A \wedge B) \wedge C$ has logical depth 2. A formula whose logical depth is D has size at most $2^{D+1} - 1$, and can depend on at most 2^D variables.

The *depth* of a formula φ, denoted by $\mathrm{dp}(\varphi)$, is the maximum number of alternations between AND and OR connectives from root to leaf, not counting negations, plus one. For example, $(A \wedge B) \wedge C$ has depth 1, and the depth of a CNF or DNF formula is two.

We have given definitions of two different notions of depth. We will use logical depth to reason about formulas in Frege proofs, and depth to reason about formulas in bounded depth proofs.

A *cut-depth k* proof, also called an AC^0_k-*Frege* proof, is a PK proof where every *cut formula* in the proof has depth at most k (other formulas are allowed to have arbitrary depth). Note that in the literature, an AC^0_k-Frege proof is often defined to be a PK proof where *all* formulas have depth at most k. This definition is equivalent to ours if the proven formula has depth at most k.

A PK proof is *tree-like* if the underlying dag structure of the proof forms a tree, i.e. each sequent is used only once.

For technical reasons, we will need all the formulas in our proofs to be balanced. By the following result of Reckhow, this can be assumed without loss of generality.

Theorem 1 (Reckhow, [11, Lemma 4.4.14]). *If a formula of logical depth D has a PK proof of size s, then it has a PK proof of size $s^{O(1)}$ in which all formulas have logical depth $D + O(\log s)$. If the original proof is tree-like, then the new balanced proof is also tree-like.*

Definition 1. *A proof system S is automatizable if there exists an algorithm A such that for all unsatisfiable formulas f, $A(f)$ returns an S-proof of f, and the runtime of A on f is polynomial in the size of the smallest S-proof of f. S is weakly automatizable if there exists a proof system that polynomially simulates S and that is automatizable.*

3 Reducing Formula Depth

We reduce the depth of a formula using a divide-and-conquer technique. The idea is to decompose the formula into relatively small sub-trees, and replace each sub-tree by a CNF or DNF which is equivalent to the formula computed by the sub-tree.

Definition 2. *Let φ be an arbitrary formula depending on n variables. Denote by* $\mathrm{CNF}(\varphi)$ *(*$\mathrm{DNF}(\varphi)$*) some canonically chosen CNF (DNF) representing φ of size $O(n2^n)$. We require that* $\mathrm{CNF}(p \wedge q) = \mathrm{DNF}(p \wedge q) = p \wedge q$*, and similarly for $p \vee q$ and $\neg p$, when p and q are variables.*

We think of formulas as trees in which internal nodes are either binary (if the corresponding connective is $\wedge$ or $\vee$) or unary (when the connective is $\neg$), and leaves are labelled by variables. Each formula has an equivalent formula of the same size where negations only appear immediately above leaves, just by applying De Morgan's laws repeatedly to "move" negations down. We will call such formulas *quasi-monotone*, and will work with them throughout our simulation.

Definition 3. *A* quasi-monotone *formula is one in which negations only appear next to variables, and there are no double negations. Let φ be a quasi-monotone formula. Its* dual *form* $\mathrm{M}(\varphi)$ *is obtained from φ by switching $\wedge$ and $\vee$ and negating all literals, that is for each variable x switching x and $\neg x$; $M(\varphi)$ is logically equivalent to $\neg\varphi$.*

We define two *canonical flattened forms* in parallel.

Definition 4. *Let $\boldsymbol{d} = d_1, \ldots, d_k$ be a vector of increasing positive integers. The* conjunctive flattened form $\mathrm{C}(\varphi; \boldsymbol{d})$ *and* disjunctive flattened form $\mathrm{D}(\varphi; \boldsymbol{d})$ *of a formula φ are defined recursively as follows. If $k = 0$ (i.e., $\boldsymbol{d}$ is the empty vector) or $d_1 \geq \mathrm{ldp}(\varphi)$ then $\mathrm{C}(\varphi; \boldsymbol{d}) = \mathrm{CNF}(\varphi)$ and $\mathrm{D}(\varphi; \boldsymbol{d}) = \mathrm{DNF}(\varphi)$. Otherwise, let ψ be the formula obtained from φ by trimming the tree at depth d_1. The formula ψ depends on the variables of φ as well as on variables corresponding to subformulas of φ at depth d_1; we call these* true variables *and* subformula variables*, respectively. Let v_χ denote the subformula variable corresponding to the subformula χ.*

We explain how to calculate the conjunctive flattened form; the disjunctive flattened form is analogous. Start with $\mathrm{CNF}(\psi)$. Let $\boldsymbol{e} = d_2 - d_1, \ldots, d_k - d_1$. Replace each positive occurrence of a subformula variable v_χ in $\mathrm{CNF}(\psi)$ with $\mathrm{D}(\chi; \boldsymbol{e})$, and each negative occurrence with $\mathrm{M}(\mathrm{C}(\chi; \boldsymbol{e}))$. The result is $\mathrm{C}(\varphi)$.

The flattened forms are both shallow and not too large.

Definition 5. *Let φ be a formula and $\boldsymbol{d} = d_1, \ldots, d_k$ be a vector of increasing positive integers, such that $d_1 \leq \mathrm{ldp}(\varphi)$. Let $d_0 = 0$ and $d_{k+1} = \mathrm{ldp}(\varphi)$. The* extent *of φ with respect to $\boldsymbol{d}$ is*

$$\mathrm{ex}(\varphi; \boldsymbol{d}) = \max\{d_{i+1} - d_i : 0 \leq i \leq k\}.$$

Lemma 1. *Let φ be a formula and $\boldsymbol{d}$ a vector of length k and extent $x = \mathrm{ex}(\varphi; \boldsymbol{d})$. Then $\mathrm{C}(\varphi; \boldsymbol{d})$ and $\mathrm{D}(\varphi; \boldsymbol{d})$ are formulas of depth at most $k+2$ and size $2^{O(k2^x)}$ equivalent to φ.*

4 Proof of Main Theorem

In this section, we will prove the following theorem.

Theorem 2. *Let φ be a formula provable in Frege in size s, satisfying $\mathrm{ldp}(\varphi) \leq C \log s$. For every $k \geq 1$ there is an AC^0_{k+2}-Frege proof of φ of size $2^{O(ks^{O(C/k)})}$. Furthermore, if the original proof is tree-like, so is the new one.*

Corollary 1. *Let φ be a formula of size s and logical depth at most $C \log s$. If φ has a Frege proof of size $O(s^c)$ then for every $k \geq 1$ there is an AC^0_{k+2}-Frege proof of φ of size $2^{O(cks^{O(C/k)})}$.*

We will first state some simple lemmas which will enable us to reason about flattened forms. The proofs of these lemmas appear in the Appendix.

Lemma 2. *Let $\Gamma \longrightarrow \Delta$ be a valid sequent of size m, in which n variables appear. The sequent is provable using a tree-like proof of size $O(m^2 n 2^n)$ which cuts only on variables.*

Our next lemma states that we can substitute formulas for variables to get a valid proof.

Lemma 3. *Let π be a proof of $\Gamma \longrightarrow \Delta$ of size s, and let x be a variable appearing in $\Gamma \longrightarrow \Delta$. If we substitute everywhere a formula φ of size m for x then we get a valid proof of size at most sm.*

The preceding lemma shows that we can lift a proof of a sequent by attaching stuff 'below'. The next lemma shows that we can also lift a proof by attaching stuff 'above'; this corresponds to deep inference.

Definition 6. *The double sequent $P \longleftrightarrow Q$ is the pair of sequents $P \longrightarrow Q$ and $Q \longrightarrow P$.*

Lemma 4. *Let $P \longrightarrow Q$ be a sequent of size m, and $\varphi(x)$ be a formula of size n in which the variable x appears only once (other variables may also appear). The double sequent $\varphi(x|P) \longleftrightarrow \varphi(x|Q)$ has a cut-free, tree-like proof from the double sequent $P \longleftrightarrow Q$ of size $O(n(m+n))$ (this means that each of $P \longrightarrow Q$ and $Q \longrightarrow P$ is used only once in the joint proof).*

We next state two easy lemmas on dualization.

Lemma 5. *Let φ be a quasi-monotone formula of size n. The double sequent $\mathrm{M}(\varphi) \longleftrightarrow \neg\varphi$ has a cut-free, tree-like proof of size $O(n^2)$.*

The second lemma allows us to lift an equivalence to its dualized version.

Lemma 6. *Let φ, ψ be quasi-monotone formulas. Suppose that the double sequent $\varphi \longleftrightarrow \psi$ has a proof of size s cutting on formulas of depth at most D. Then the double sequent $\mathrm{M}(\varphi) \longleftrightarrow \mathrm{M}(\psi)$ has a proof of size $O(s)$ cutting on formulas of depth at most D. Furthermore, if the original proof is tree-like then so is the new proof.*

We comment that the preceding lemma can be strengthened to produce cut-free proofs.

4.1 Moving Down the Depth Vector

In this section we show how to prove the equivalence of two flattened forms of the same formula which correspond to two different depth vectors.

Lemma 7. *Let φ be a formula of logical depth D, and δ a positive integer. Consider $\mathrm{CNF}(\varphi)$ and $\mathrm{C}(\varphi;\delta)$ as monotone formulas depending on literals $x, \bar{x}$; in other words, for each variable x, we replace $\neg x$ by $\bar{x}$. The double sequent $\mathrm{CNF}(\varphi) \longleftrightarrow \mathrm{C}(\varphi;\delta)$ has a tree-like proof of size $2^{O(2^D)}$ cutting only on literals.*

Lemma 8. *Let φ be a formula, $\boldsymbol{d} = d_1, \ldots, d_k$ a vector of increasing positive integers, and $\delta < d_1$ be a positive integer. The double sequent $\mathrm{C}(\varphi;\boldsymbol{d}) \longleftrightarrow \mathrm{C}(\varphi;\delta,\boldsymbol{d})$ has a tree-like proof of size $2^{O(k2^x)}$ cutting only on formulas of depth at most $k+1$, where $x = \mathrm{ex}(\varphi;\boldsymbol{d})$.*

Lemma 9. *Let φ be a formula, $\boldsymbol{d} = d_1, \ldots, d_k$ a vector of increasing positive integers, and $d_i < \delta < d_{i+1}$, where $1 \leq i \leq k$. Define $\boldsymbol{e} = d_1, \ldots, d_i, \delta, d_{i+1}, \ldots, d_k$. The double sequent has a tree-like proof of size $2^{O(k2^x)}$ cutting only on formulas of depth at most $k+1$, where $x = \mathrm{ex}(\varphi; \boldsymbol{d})$.*

Lemma 10. *Let φ be a formula, and $\boldsymbol{d} = d_1, \ldots, d_k$ be a vector of increasing positive integers. Define $\boldsymbol{e} = 1, d_1 + 1, \ldots, d_k + 1$. The double sequent $\mathrm{C}(\varphi; \boldsymbol{d}) \longleftrightarrow \mathrm{C}(\varphi; \boldsymbol{e})$ has a tree-like proof of size $2^{O(k2^x)}$ cutting only on formulas of depth at most $k+3$, where $x = \max(\mathrm{ex}(\varphi; \boldsymbol{d}), \mathrm{ex}(\varphi; \boldsymbol{e}))$.*

The same methods used to prove Lemma 9 enable us to prove the following lemma.

Lemma 11. *Let φ be a formula, and $\boldsymbol{d} = d_1, \ldots, d_k$ be a vector of increasing positive integers. The double sequent $\mathrm{C}(\varphi; \boldsymbol{d}) \longleftrightarrow \mathrm{D}(\varphi; \boldsymbol{d})$ has a tree-like proof of size $2^{O(k2^x)}$ cutting only on formulas of depth at most $k+2$, where $x = \mathrm{ex}(\varphi; \boldsymbol{d})$.*

4.2 Putting It Together

In this section we show how to transform a Frege proof to an AC^0-Frege proof. We begin by proving intensional comprehension.

Lemma 12. *Let φ, ψ be formulas, and $\boldsymbol{d}$ be a vector of increasing positive integers of length k. The double sequents*

$$\mathrm{C}(\varphi \wedge \psi; \boldsymbol{d}) \longleftrightarrow \mathrm{C}(\varphi; \boldsymbol{d}) \wedge \mathrm{C}(\psi; \boldsymbol{d}), \quad \mathrm{C}(\varphi \vee \psi; \boldsymbol{d}) \longleftrightarrow \mathrm{C}(\varphi; \boldsymbol{d}) \vee \mathrm{C}(\psi; \boldsymbol{d})$$

have tree-like proofs of size $2^{O(k2^x)}$ with cuts on formulas of depth at most $k+3$, where $x = \mathrm{ex}(\varphi \wedge \psi; \boldsymbol{d})$.

Lemma 13. *Let φ be a formula, and $\boldsymbol{d}$ be a vector of increasing positive integers of length k. The double sequent $\mathrm{C}(\neg\varphi; \boldsymbol{d}) \longleftrightarrow \neg\,\mathrm{C}(\varphi; \boldsymbol{d})$ has a tree-like proof of size $2^{O(k2^x)}$ with cuts on formulas of depth at most $k+3$, where $x = \mathrm{ex}(\neg\varphi; \boldsymbol{d})$.*

The preceding lemmas allow us to unroll flattened forms.

Lemma 14. *Let φ be a formula, and $\boldsymbol{d}$ be a vector of increasing positive integers of length k. The double sequent $\varphi \longleftrightarrow \mathrm{C}(\varphi; \boldsymbol{d})$ has a tree-like proof of size $2^{O(k2^x)}$ with cuts on formulas of depth at most $k+3$, where $x = \mathrm{ex}(\varphi; \boldsymbol{d})$.*

Proof. The proof is by structural induction. If φ is a literal then there is nothing to prove. If $\varphi = \neg\psi$, then use Lemma 13 to prove $\mathrm{C}(\varphi; \boldsymbol{d}) \longleftrightarrow \neg\,\mathrm{C}(\psi; \boldsymbol{d})$. The induction hypothesis gives us a proof of $\psi \longleftrightarrow \mathrm{C}(\psi; \boldsymbol{d})$; move both ψ and its flattened form to the other side using four $\neg$ introduction rules, and apply cut twice to prove the required double sequent.

If $\varphi = \psi \wedge \chi$ then start with proofs of the following sequents, obtained by Lemma 12 and the induction hypothesis:

$$\mathrm{C}(\varphi; \boldsymbol{d}) \longleftrightarrow \mathrm{C}(\psi; \boldsymbol{d}) \wedge \mathrm{C}(\chi; \boldsymbol{d}), \qquad \psi \longleftrightarrow \mathrm{C}(\psi; \boldsymbol{d}), \qquad \chi \longleftrightarrow \mathrm{C}(\chi; \boldsymbol{d}).$$

Now prove $\mathrm{C}(\psi;\boldsymbol{d}) \wedge \mathrm{C}(\chi;\boldsymbol{d}) \longleftrightarrow \psi \wedge \chi$ as follows:

$$\dfrac{\dfrac{\mathrm{C}(\psi;\boldsymbol{d}) \longrightarrow \psi}{\mathrm{C}(\psi;\boldsymbol{d}) \wedge \mathrm{C}(\chi;\boldsymbol{d}) \longrightarrow \psi}\wedge\mathsf{L} \qquad \dfrac{\mathrm{C}(\chi;\boldsymbol{d}) \longrightarrow \chi}{\mathrm{C}(\psi;\boldsymbol{d}) \wedge \mathrm{C}(\chi;\boldsymbol{d}) \longrightarrow \chi}\wedge\mathsf{L}}{\mathrm{C}(\psi;\boldsymbol{d}) \wedge \mathrm{C}(\chi;\boldsymbol{d}) \longrightarrow \psi \wedge \chi}\wedge\mathsf{R}$$

The other sequent is proved similarly. Complete the proof using the cut rule. The case $\varphi = \psi \vee \chi$ is similar.

The proof of the main theorem is now simple.

Lemma 15. *Let φ be a formula provable in Frege in size s using a proof with maximum logical depth D. For every k there is an AC^0_{k+2}-Frege proof of φ of size $s2^{O(k2^{D/k})}$. Furthermore, if the original proof is tree-like, so is the new one.*

Proof. Let $\boldsymbol{d} = \lceil D/k \rceil, 2\lceil D/k \rceil, \ldots, (k-1)\lceil D/k \rceil$. Note that the extent of each formula with respect to $\boldsymbol{d}$ is at most $x = \lceil D/k \rceil$. Take the original proof and replace each formula ψ by $\mathrm{C}(\psi;\boldsymbol{d})$. Each application of a rule is still valid, but the proof as a whole isn't valid since not all formulas are in flattened form. We address this issue by tampering with the introduction rules, as in the following example, corresponding to the right $\wedge$ introduction rule:

$$\dfrac{\dfrac{\Gamma \longrightarrow \Delta, \mathrm{C}(\psi;\boldsymbol{d}) \qquad \Gamma \longrightarrow \Delta, \mathrm{C}(\psi;\boldsymbol{d})}{\Gamma \longrightarrow \Delta, \mathrm{C}(\psi;\boldsymbol{d}) \wedge \mathrm{C}(\chi;\boldsymbol{d})}\wedge\mathsf{R} \qquad \dfrac{}{\mathrm{C}(\psi;\boldsymbol{d}) \wedge \mathrm{C}(\chi;\boldsymbol{d}) \longrightarrow \mathrm{C}(\psi \wedge \chi;\boldsymbol{d})}\,\textbf{Lem. 12}}{\Gamma \longrightarrow \Delta, \mathrm{C}(\psi \wedge \chi;\boldsymbol{d})}\,\textbf{Cut}$$

Applying the same transformation for all introduction rules, we are left with a valid proof of $\longrightarrow \mathrm{C}(\varphi;\boldsymbol{d})$, where each sequent is now replaced by sequents of total size $2^{O(k2^x)}$; the total size so far is $s2^{O(k2^x)}$. Lemma 14 proves $\mathrm{C}(\varphi;\boldsymbol{d}) \longrightarrow \varphi$, and the proof is complete by cutting on $\mathrm{C}(\varphi;\boldsymbol{d})$.

The lemmas we used employ cuts of depth at most $k+2$. All cuts in the original proof now cut flattened formulas, which are of depth at most $k+1$.

Proof (of Theorem 2). Reckhow's Theorem (Theorem 1) supplies us with an $\mathrm{AC}^0_{O(\log s)}$ proof of φ of size $s^{O(1)}$. The theorem now follows by substituting $D = C\log(s)$ in Lemma 15.

5 Applications and Consequences

5.1 Tightness of Our Simulation

We first address the tightness of our simulation. The analogous result for circuit complexity shows that any function computable by a polynomial-size formula can be computed by depth d circuits of size $\exp(n^{O(1/d)})$. This result is tight, since Håstad's theorem proves that the parity function on n Boolean variables requires AC^0_d circuits of size $\exp(n^{1/d})$.

Similarly we can show that our result is also tight. The following theorem states that there are formulas that have polynomial-size Frege proofs, but that require AC^0_d proofs of size exponential in $n^{1/d}$.

Theorem 3. *For every d there is a sequence of balanced formulas φ_n of depth $d+2$ provable in Frege by a tree-like proof of size s_n such that every tree-like AC^0_d proof of φ_n requires size $2^{s_n^{\Omega(1/d)}}$.*

Proof. The formula φ_n is PHP_n, the pigeonhole principle with $n+1$ pigeons and n holes, with each variable replaced by a Sipser function of depth d. Buss [7] has shown how to prove PHP_n using a Frege proof of size $n^{O(1)}$, which can be made tree-like by squaring its size. Substituting the Sipser functions, we obtain a Frege proof of size $n^{d+O(1)}$.

Conversely, Krajíček [10] gives a lower bound of $2^{n^{\Omega(1)}}$ for proving φ_n in tree-like AC^0_d.

Since the formulas φ_n are balanced, Theorem 2 applies, and with $k = d-2$, gives proofs essentially matching the lower bound.

The above result proves tightness for formulas of high depth. We conjecture that our simulation is also tight with respect to CNF formulas and general, dag-like proofs. The obvious formula for witnessing the lower bound is the pigeonhole principle itself. However, as an artifact of the switching lemma technique used to obtain depth d Frege lower bounds for the pigeonhole principle, the current best lower bound is exponential in $n^{1/2^d}$. It is a well-known open problem to improve the lower bound to $\exp(n^{1/d})$ for the pigeonhole principle, or for any other CNF formula. Such a result would show that our simulation is tight even for CNF formulas and arbitrary dag-like proofs.

5.2 Weak Automatizability

Using our theorem, we are able to show that bounded-depth Frege is not weakly automatizable, under an assumption about the hardness of factoring. While this result has already been known [5], we first show how to prove it as a simple corollary of our main theorem.

Theorem 4 ([6]). *Frege systems do not have feasible interpolation and are not weakly automatizable unless the Diffie Hellman problem is computable by polynomial size circuits.*

The Diffie Hellman problem is based on a prime number p, $|p| = n$. The input to the problem is a number g less than p, and numbers $g^a \pmod p$, $g^b \pmod p$, for some numbers $a, b \leq p$. The output should be $g^{ab} \pmod p$. The main lemma from [6] shows that a particular tautology, DH_p, stating that the Diffie Hellman function is well defined, has Frege proofs of size $O(|p|^c)$, where $c \leq 4$. Take DH_p where $|p| = (\log n)^q$ for some constant q. By our normal form theorem, this implies that DH_p has AC^0_k-Frege proofs of size $2^{O(k(\log n)^{cq/k})}$. Thus for $k > cq$, this is polynomial in n. Hence it follows that if AC^0_k-Frege is weakly automatizable (or has feasible interpolation), then the Diffie Hellman problem for $|p| = n' = (\log n)^{k/c}$ can be solved in time $n^{O(k)} = 2^{O(k \log n)} = \exp O(k(n')^{c/k})$.

Unfortunately, the quality of this negative result degrades for small k. Indeed despite considerable effort, it is unknown whether or not very low depth Frege

systems (when k is less than 5) are weakly automatizable (the recent paper [2] reveals a connection between automatizability of AC^0_2-Frege with bottom fan-in 2 and feasibility of mean-payoff games). The main reason for this is that the Diffie Hellman function is not hard enough! Algorithms exist for computing discrete log over all finite fields, and hence for Diffie Hellman, that run in time exponential in $\sqrt{n}$. Moreover, the number field sieve is conjectured to solve discrete log (and thus Diffie Hellman) in time exponential in $\sqrt[3]{n}$. On the other hand, it seems entirely possible to come up with a different interpolant statement for another function that is much harder – truly exponential in n, and that still has efficient Frege proofs. Using our main theorem (which scales down *any* Frege proof), this would imply new negative results for weak automatizability and feasible interpolation for lower depth Frege systems than what is currently known.

References

1. Allender, E., Hellerstein, L., McCabe, P., Pitassi, T., Saks, M.: Minimizing disjunctive normal form formulas and AC^0 circuits given a truth table. SIAM Journal on Computing 38(1), 63–84 (2008)
2. Atserias, A., Maneva, E.: Mean-payoff games and propositional proofs. Inf. and Comp. 209(4), 664–691 (2011)
3. Beame, P.W., Impagliazzo, R., Krajíček, J., Pitassi, T., Pudlák, P., Woods, A.: Exponential lower bounds for the pigeonhole principle. In: Proceedings of the Twenty-Fourth Annual ACM Symposium on Computing, Victoria, B.C., Canada, pp. 200–220 (May 1992)
4. Bonet, M.L., Buss, S.R., Pitassi, T.: Are there hard examples for Frege systems? In: Feasible Mathematics II, pp. 30–56. Birkhäuser, Basel (1995)
5. Bonet, M.L., Domingo, C., Gavaldà, R., Maciel, A., Pitassi, T.: Non-automatizability of bounded-depth Frege proofs. Computational Complexity 13(1-2), 47–68 (2004)
6. Bonet, M.L., Pitassi, T., Raz, R.: On interpolation and automatization for Frege systems. SIAM Journal on Computing 29(6), 1939–1967 (2000)
7. Buss, S.R.: Polynomial size proofs of the pigeonhole principle. Journal of Symbolic Logic 57, 916–927 (1987)
8. Cook, S.A., Reckhow, R.A.: The relative efficiency of propositional proof systems. Journal of Symbolic Logic 44(1), 36–50 (1979)
9. Haken, A.: The intractability of resolution. Theoretical Computer Science 39, 297–305 (1985)
10. Krajíček, J.: Lower bounds to the size of constant-depth propositional proofs. Journal of Symbolic Logic 59(1), 73–86 (1994)
11. Krajíček, J.: Bounded arithmetic, propositional logic, and complexity theory. Cambridge University Press, New York (1995)

Parameterized Bounded-Depth Frege Is Not Optimal

Olaf Beyersdorff[1,⋆], Nicola Galesi[2,⋆⋆],
Massimo Lauria[2], and Alexander Razborov[3,⋆⋆⋆]

[1] Institut für Theoretische Informatik, Leibniz Universität Hannover, Germany
[2] Dipartimento di Informatica, Sapienza Università di Roma, Italy
[3] Department of Computer Science, The University of Chicago

Abstract. A general framework for parameterized proof complexity was introduced by Dantchev, Martin, and Szeider [9]. There the authors concentrate on tree-like Parameterized Resolution—a parameterized version of classical Resolution—and their gap complexity theorem implies lower bounds for that system.

The main result of the present paper significantly improves upon this by showing optimal lower bounds for a parameterized version of bounded-depth Frege. More precisely, we prove that the pigeonhole principle requires proofs of size $n^{\Omega(k)}$ in parameterized bounded-depth Frege, and, as a special case, in dag-like Parameterized Resolution. This answers an open question posed in [9]. In the opposite direction, we interpret a well-known technique for FPT algorithms as a DPLL procedure for Parameterized Resolution. Its generalization leads to a proof search algorithm for Parameterized Resolution that in particular shows that tree-like Parameterized Resolution allows short refutations of all parameterized contradictions given as bounded-width CNF's.

1 Introduction

Recently, Dantchev, Martin, and Szeider [9] introduced the framework of *parameterized proof complexity*, an extension of the proof complexity approach of Cook and Reckhow to parameterized complexity. One motivation for this is the quest for efficient algorithms which solve optimization problems [10,11,16]. Since Resolution is very important for SAT solving, its analogue in this context, Parameterized Resolution, combines these two approaches, and its investigation might provide new insights into proof search for tractable fragments of classically hard problems. Some results in this direction are already outlined in the

⋆ Part of this work was done while the first author was visiting Sapienza University Rome under support of grant N. 20517 by the John Templeton Foundation.
⋆⋆ Supported by grant "Limiti di compressione in combinatoria e complessità computazionale" by Sapienza University Rome.
⋆⋆⋆ Part of this work was done while the author was at Steklov Mathematical Institute, supported by the Russian Foundation for Basic Research, and at Toyota Technological Institute, Chicago.

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 630–641, 2011.

work of Gao [13] where he analyzes the effect of the standard DPLL algorithm on the problem of weighted satisfiability for random d-CNF. However, the study of Parameterized Resolution and our understanding of the possible implications for SAT-solving algorithms are still at a very early stage.

More generally, *parameterized complexity* is a branch of complexity theory where problems are analyzed in a different way than in the classical approach: we say that a problem is *fixed-parameter tractable* (FPT) with parameter k if any instance of size n can be solved in time $f(k)n^{O(1)}$ for some computable function f of arbitrary growth. In this setting, classically intractable problems may have efficient solutions for small choices of the parameter, even if the total size of the input is large. Consider e.g. the classical satisfiability problem of finding a truth assignment that satisfies all clauses of a formula in conjunctive normal form. Bounded CNF Sat and Weighted CNF Sat are parameterized variants of CNF satisfiability in which the satisfying assignment is required to have Hamming weight at most k or exactly k, respectively. Many parameterized combinatorial problems can be naturally encoded in Bounded CNF Sat or Weighted CNF Sat: finding a vertex cover of size at most k, finding a clique of size k, or finding a dominating set of size at most k. In the theory of parameterized complexity, the hardness of both problems is reflected by their W[2]-completeness.

In [9], Dantchev, Martin, and Szeider laid the foundations to study complexity of proofs in a parameterized setting. The dual problem of Bounded CNF Sat is that of deciding *parameterized contradictions* PCon: it consists of all pairs (F, k) where F is a propositional formula F which has no satisfying assignment of weight $\leq k$. After considering this notion of propositional *parameterized tautologies*, Dantchev et al. [9] introduced the concepts of parameterized proof systems and of fpt-bounded proof systems (see Section 2 for a discussion). The main motivation behind the work of [9] was that of generalizing the classical approach of Cook and Reckhow to the parameterized case and working towards a separation of parameterized complexity classes as FPT and W[2] by techniques developed in proof complexity. In fact, we obtain an analogous result to the well-known Cook-Reckhow theorem from [8]: A parameterized language L has an fpt-bounded proof system if and only if $L \in$ para-NP (Theorem 1).

In [9] (tree-like) *Parameterized Resolution* was defined as a refutation system for the set of parameterized contradictions. If $(F, k) \in$ PCon is defined on variables $x_1, \ldots, x_n$ then a (tree-like) *Parameterized Resolution refutation* of (F, k) is a (tree-like) Resolution refutation of $F \cup \{\neg x_{i_1} \vee \cdots \vee \neg x_{i_{k+1}} \mid 1 \leq i_1 < \cdots < i_{k+1} \leq n\}$. Thus, in (tree-like) Parameterized Resolution we have built-in access to all parameterized clauses of the form $\neg x_{i_1} \vee \cdots \vee \neg x_{i_{k+1}}$. All these clauses are available in the system, but when measuring the size of a derivation we only count those which actually appear in the derivation. This concept can be straightforwardly generalized to an arbitrary proof system P, be it dag-like or tree-like, that understands clauses and works with lines.

Dantchev et al. [9] prove an extension of Riis' *gap theorem* [20] and obtain a model theoretic classification for the complexity of tree-like Parameterized Res-

olution refutations for parameterized contradictions originating as propositional encodings of first-order formulas. In particular, their main result implies that tree-like Parameterized Resolution is not fpt-bounded. A similar question for dag-like Parameterized Resolution was left open in [9]. More specifically, they asked if (the parameterized version of) the pigeonhole principle is hard for dag-like Parameterized Resolution.

We answer this question by proving that PHP_n^{n+1} requires proofs of size $n^{\Omega(k)}$ not only in Parameterized Resolution but in the much stronger system of bounded-depth Frege. Our result is in sharp contrast with [9, Proposition 17] that gives efficient proofs of PHP_n^{n+1} in Parameterized Resolution using a more sophisticated encoding with auxiliary variables. We discuss these augmented proof systems in the final Section 5. Our lower bound for the pigeonhole principle is a rather simple application of the method of random restrictions introduced in proof complexity by Haken in his seminal paper [14]. But our choice of parameters is totally different and allows us to kill with the restriction any small prescribed set of parameterized axioms. While the technique is routine, it nonetheless seems to be its first application in the context of *parameterized* complexity, be it computational or proof complexity.

As our second contribution we investigate classes of parameterized contradictions that have short refutations in tree-like Parameterized Resolution. The notion of *kernelization* plays an important role in the theory of parameterized complexity to design fpt-algorithms. Here we propose a notion of *core* for parameterized proof complexity: the core of a parameterized contradiction (F, k) is a subset of clauses $F' \subseteq F$ whose size is bounded by a function of k only, and such that (F', k) is still a parameterized contradiction. We observe that if a formula has a core, then it can be efficiently refuted in tree-like Parameterized Resolution with a refutation of size independent of the size of F. As an immediate consequence, several examples of formulas hard for tree-like Resolution are instead efficiently refutable in the parameterized case: pebbling contradictions, linear ordering principles, graph pigeonhole principles, and colorability principles. But sometimes a core of a formula is not explicit or immediate to find. In Theorem 3 we prove that contradictions of bounded width have a core and thus very efficient tree-like Parameterized Resolution refutations.

2 Parameterized Proof Complexity

A *parameterized language* is a language $L \subseteq \Sigma^* \times \mathbb{N}$. For an instance (x, k), we call k the *parameter of* (x, k). A parameterized language L is *fixed-parameter tractable* if L has a deterministic decision algorithm running in time $f(k)|x|^{O(1)}$ for some computable function f. The class of all fixed-parameter tractable languages is denoted by FPT.

Besides FPT there is a wealth of complexity classes containing problems which are not believed to be fixed-parameter tractable. The most prominent classes lie in the *weft hierarchy* forming a chain $\mathsf{FPT} \subseteq \mathsf{W}[1] \subseteq \mathsf{W}[2] \subseteq \cdots \subseteq \mathsf{W}[\mathsf{P}]$. All these classes are included in $\mathsf{para\text{-}NP}$ (see Definition 6). The classes of the weft

hierarchy are usually defined as the closure of a canonical problem under fpt-reductions. For W[2] this canonical problem is Weighted CNF Sat containing instances (F, k) with a propositional formula F in CNF and a parameter $k \in \mathbb{N}$. Weighted CNF Sat asks whether F has a satisfying assignment of weight exactly k, where the weight of an assignment α, denoted as $w(\alpha)$, is the number of variables that α assigns to 1. Instead of asking for an assignment α with $w(\alpha) = k$ we can also ask for α with $w(\alpha) \leq k$ and still get the W[2]-complete problem Bounded CNF Sat (cf. [9] or the full version of this paper for the proof of its W[2]-completeness).

Like in the classical duality between tautologies and satisfiability, the complement of Bounded CNF Sat is a complete problem for coW[2]:

Definition 1 (Dantchev, Martin, Szeider [9]). *A* parameterized contradiction *is a pair (F, k) consisting of a propositional formula F, given as a CNF, and $k \in \mathbb{N}$ such that F has no satisfying assignment of weight $\leq k$. We denote the set of all parameterized contradictions by* PCon.

Next we discuss the general definition of a parameterized proof system from [9].

Definition 2 (Dantchev, Martin, Szeider [9]). *A* parameterized proof system *for a parameterized language $L \subseteq \Sigma^* \times \mathbb{N}$ is a function $P : \Sigma^* \times \mathbb{N} \to \Sigma^* \times \mathbb{N}$ such that $rng(P) = L$ and $P(x, k)$ can be computed in time $f(k)|x|^{O(1)}$ with some computable function f.*

The purpose of the second argument in P remains a little bit unclear to us since all natural proof systems we can think of do not have this feature. Thus, we propose the following simplification.

Definition 3. *A* proof system *for a parameterized language $L \subseteq \Sigma^* \times \mathbb{N}$ is a polynomial-time computable function $P : \Sigma^* \to \Sigma^* \times \mathbb{N}$ such that $rng(P) = L$.*

Now we would like to show that both versions are even formally equivalent in the sense that a parameterized language has a proof system in which all strings possess "short" proofs if and only if it has a parameterized proof system with this property. First we have to formalize the notion of "short". In the framework of [9] it goes as follows:

Definition 4 (Dantchev, Martin, Szeider [9]). *A parameterized proof system P for a parameterized language L is* fpt-bounded *if there exist computable functions f and g such that every $(x, k) \in L$ has a P-proof (y, k') with $|y| \leq f(k)|x|^{O(1)}$ and $k' \leq g(k)$.*

Again, our analogue is simpler.

Definition 5. *A proof system P for a parameterized language L is* fpt-bounded *if there exists a computable function f such that every $(x, k) \in L$ has a P-proof of size at most $f(k)|x|^{O(1)}$.*

Recall that by the theorem of Cook and Reckhow [8], the class of all languages with polynomially bounded proof systems coincides with NP. To obtain a similar result in the parameterized world, we use the following parameterized version of NP.

Definition 6 (Flum, Grohe [11]). *The class* para-NP *contains all parameterized languages which can be decided by a nondeterministic Turing machine in time* $f(k)|x|^{O(1)}$ *for a computable function* f.

Theorem 1. *Let* $L \subseteq \Sigma^* \times \mathbb{N}$ *be a parameterized language. Then the following statements are equivalent: (1) There exists an fpt-bounded proof system for* L. *(2) There exists an fpt-bounded parameterized proof system for* L. *(3)* $L \in$ para-NP.

Proof (Sketch). (1) equivalent to (2): by padding the proof, the proof verification procedure can be made polynomial in proof length. (1) equivalent to (3): a proof can be guessed nondeterministically and then verified. □

2.1 Parameterized Versions of Ordinary Proof Systems

A *literal* is a positive or negated propositional variable and a *clause* is a set of literals. The *width* of a clause is the number of its literals. A clause is interpreted as the disjunction of its literals and a set of clauses as the conjunction of the clauses. Hence clause sets correspond to formulas in CNF.

The system of *Parameterized Resolution* was introduced by Dantchev, Martin, and Szeider [9]. Parameterized Resolution is a refutation system for the set PCon of parameterized contradictions (cf. Definition 1). Given a set of clauses F in variables $x_1, \ldots, x_n$ with $(F, k) \in$ PCon, a *Parameterized Resolution refutation* of (F, k) is a Resolution refutation of

$$F \cup \{\neg x_{i_1} \vee \cdots \vee \neg x_{i_{k+1}} \mid 1 \leq i_1 < \cdots < i_{k+1} \leq n\}. \tag{1}$$

Thus, in Parameterized Resolution we have built-in access to all parameterized clauses of the form $\neg x_{i_1} \vee \cdots \vee \neg x_{i_{k+1}}$. All these clauses are available in the system, but when measuring the size of a derivation we only count those which appear in the derivation. Note that Parameterized Resolution is actually a proof system for PCon in the sense of Definition 3, i. e., verification proceeds in polynomial time. This definition allows the following straightforward generalization.

Definition 7. *Let* $P : \Sigma^* \to$ *Con be an ordinary proof system for the language Con of all (ordinary) CNF contradictions. We define the* parameterized version $\widehat{P}$ *of* P *by letting* $\widehat{P}(F, k, x) = (F, k)$ *whenever* $P(x)$ *is an arbitrary subset of the set of axioms* (1). *If* $P(x)$ *does not have this form,* $\widehat{P}(F, k, x)$ *outputs something trivial.*

The only specific proof system we would like to comment on is tree-like Parameterized Resolution (as it will be needed in Section 4). As explained in [9], a tree-like Parameterized refutation of (F, k) can equivalently be described as a *boolean decision tree*. A boolean decision tree for (F, k) is a binary tree where inner nodes are labeled with variables from F and leafs are labeled with clauses from F or parameterized clauses $\neg x_{i_1} \vee \cdots \vee \neg x_{i_{k+1}}$. Each path in the tree corresponds to a partial assignment where a variable x gets value 0 or 1 according to whether the path branches left or right at the node labeled with x. The condition

on the decision tree is that each path α must lead to a clause which is falsified by the assignment corresponding to α. Therefore, a boolean decision tree solves the *search problem* for (F, k) which, given an assignment α, asks for a clause falsified by α. It is easy to verify that each tree-like Parameterized Resolution refutation of (F, k) yields a boolean decision tree for (F, k) and vice versa, where the size of the Resolution proof equals the number of nodes in the decision tree.

An embarrassing fact about Parameterized Proof Complexity (brought to our attention by an anonymous referee of a previous version of this paper) is that, as defined in Definition 7, $\widehat{P}$ is *never* bounded for some dull reasons.

Example 1. Let (F, k) be the parameterized contradiction in which F is the set of positive clauses $\{x_{1,1} \vee \ldots \vee x_{1,n}, \ldots, x_{k+1,1} \vee \ldots \vee x_{k+1,n}\}$. Then in order to make this set even *semantically* invalid, one has to append to it all n^{k+1} parameterized axioms of the form $\neg x_{1,j_1} \vee \ldots \vee \neg x_{k+1,j_{k+1}}$.

Obviously, this is not the kind of phenomena we want to study (and not the kind of methods we want to develop) so we have to try to somehow isolate such pathological examples. One approach (borrowed from circuit complexity) would be simply to declare some parameterized contradictions "natural", "interesting" or "explicit" without giving precise definitions or even revealing exact reasons for this classification. Another possibility (that we adopt in this paper) is to *formally* restrict the set of contradictions we are interested in.

Definition 8. *A parameterized contradiction* (F, k) *is* strong *if* F *itself is a contradiction. A proof system* P *for the set* PCon *is* weakly fpt-bounded *if there exists a computable function* f *such that every* **strong** $(F, k) \in$ PCon *has a* P*-proof of size at most* $f(k)|F|^{O(1)}$.

One reason to introduce this restriction is that "interesting" contradictions are almost always strong. In fact, the only exception we are aware of (even if it is the one that inspired almost all material in Section 4) is the vertex cover problem.

On a more philosophical level, the concept of a strong parameterized contradiction intends to capture the idea that the new knowledge provided by parameterized axioms should be rather thought of as a *helper* or an *additional feature* made available to already existing DPLL algorithms rather than being the prime source of the *validity* of the statement. It is perhaps useful to note at this point that there is no monotonicity on k in Definition 8, either way. Since, as k grows, our size constraint $f(k)|F|^{O(1)}$ gets looser, but, on the other hand, the helper axioms become less helpful.

Finally, we are not aware of any analogue of Example 1 for strong parameterized contradictions.

Yet another possibility to get rid of this example is to try to encode parameterized axioms in (1) in a more economical way (so that their number stays small), possibly using some auxiliary variables. For Parameterized Resolution this possibility was discussed already in [9], and we continue this discussion in a broader context in Section 5.

3 Parameterized Bounded-Depth Frege Is Not Weakly fpt-Bounded

The *pigeonhole principle* PHP_n^{n+1} uses variables $x_{i,j}$ with $i \in [n+1]$ and $j \in [n]$, indicating that pigeon i goes into hole j. PHP_n^{n+1} consists of the clauses $\bigvee_{j\in[n]} x_{i,j}$ for all pigeons $i \in [n+1]$ and $\neg x_{i_1,j} \vee \neg x_{i_2,j}$ for all choices of two distinct pigeons $i_1, i_2 \in [n+1]$ and a hole $j \in [n]$.

Let F_d be the fragment of the Frege system over de Morgan basis $\{\neg, \wedge, \vee\}$ that operates with formulas of logical depth at most d.

Theorem 2. *For any fixed $d, k \geq 0$ and all sufficiently large n, any refutation of (PHP_n^{n+1}, k) in $\widehat{F_d}$, the parameterized version of F_d, requires size $\geq n^{k/5}$.*

Note that, somewhat surprisingly, d does not appear in the final bound at all (although it implicitly appears in the bound assumed in the "sufficiently large" premise).

Proof. Choose uniformly at random a set I of $n - \sqrt{n}$ pigeons and match them with a set J of $n - \sqrt{n}$ uniformly chosen holes. Such partial matching f induces the following natural partial assignment of the variables of PHP_n^{n+1}:

$$\begin{array}{ll} x_{i,j} = 1 & \text{whenever } i \in I \text{ and } f(i) = j, \\ x_{i,j} = 0 & \text{whenever } i \in I \text{ and } f(i) \neq j, \\ x_{i,j} = 0 & \text{whenever } j \in J \text{ and there exist } i' \neq i \text{ such that } f(i') = j, \text{ and} \\ x_{i,j} = \star & \text{otherwise.} \end{array}$$

We claim that with non-zero probability such partial assignment satisfies all parameterized axioms used in the refutation, as long as there are at most $n^{k/5}$ of them. (Notice that we do not care if such assignment falsifies unused parameterized axioms.) Before proving this claim, we show how the theorem follows.

The refutation, restricted with such assignment, does not contain parameterized axiom anymore. Thus it is a classical F_d-refutation for the restricted formula, which in turn is equivalent (up to a re-indexing of pigeons and holes) to $PHP_{\sqrt{n}}^{\sqrt{n}+1}$. Such refutation must be of size at least $2^{n^{c_d}}$ [17,15] for some $c_d > 0$, thus bigger than $n^{k/5}$ if n is sufficiently large. This concludes the proof.

The missing part is to show that the probabilistic choice of the partial matching realizes the desired properties with positive probability. Consider a parameterized axiom $\neg x_{i_1,j_1} \vee \ldots \vee \neg x_{i_{k+1},j_{k+1}}$. If there are two equal indexes j_a and j_b for $a \neq b$, then such axiom is just a weakening of a standard clause of the pigeonhole principle and does not need any special treatment.

We can now focus on a parameterized axiom in which exactly $k+1$ holes are represented: the probability that such axiom fails to be satisfied is the probability that all x_{i_l,j_l} are either true or unassigned for $1 \leq l \leq k+1$. Let $J_0 = \{j_1, \ldots, j_{k+1}\}$ be the set of all holes represented in our axiom. The probability that the support J of our random restriction contains at most $k/2$ of them (and hence the complement to J that has size $\sqrt{n}$ contains at least $k/2$ of them) is bounded by $\binom{k+1}{k/2} \cdot \frac{\binom{n-k/2}{\sqrt{n}-k/2}}{\binom{n}{\sqrt{n}}} \leq 2^{k+1} n^{-k/4}$. And, conditioned

by the event $|J \cap J_0| \geq k/2$, the probability that every hole $j_a \in J \cap J_0$ is sent by the matching f to the right pigeon i_a (so that x_{i_a,j_a} is not set to 0) is at most $(n-k/2)^{-k/2}$. Thus, the overall probability that our random partial assignment does not satisfy an individual parameterized axiom is bounded by $2^{k+1}n^{-k/4} + (n-k/2)^{-k/2} < n^{-k/5}$ for sufficiently large n. By the union bound, if our refutation has size $\leq n^{k/5}$, then for at least one particular choice of f the corresponding assignment satisfies all parameterized axioms actually used in the refutation. As we already observed, this concludes the proof. □

The same proof works for weaker versions of the pigeonhole principle, like functional or onto, and it works up to $k \leq n^{\epsilon_d}$, where $\epsilon_d > 0$ depends only on the depth of the Frege system. If we consider Parameterized Resolution instead of parameterized bounded-depth Frege, our proof applies also to the pigeonhole principle with arbitrarily many pigeons. See the full version of this paper for details.

4 Cores and Small Refutations

The notion of *kernelization* plays an important role in the theory of parameterized complexity. A kernelization for a parameterized language L is a polynomial-time procedure $A : \Sigma^* \times \mathbb{N} \to \Sigma^* \times \mathbb{N}$ such that for each (x,k) it holds that (1) $(x,k) \in L$ if and only if $A(x,k) \in L$ and (2) if $A(x,k) = (x',k')$, then $k' \leq k$ and $|x'| \leq f(k)$ for some computable function f independent of $|x|$.

It is clear that if a parameterized language admits a kernelization then it is fixed-parameter tractable. The converse is also true for decidable languages (cf. [12]). For parameterized proof complexity we suggest a similar notion of core for parameterized contradictions:

Definition 9. *A* core *for a set* $\Gamma \subseteq \mathrm{PCon}$ *of parameterized contradictions is a mapping which maps every* $(F,k) \in \Gamma$ *to a subset* $F' \subseteq F$ *of clauses satisfying the following conditions: (1)* F' *contains at most* $f(k)$ *variables and (2)* (F',k) *is a parameterized contradiction, where* f *is a computable function depending only on the mapping.*

We will sometimes abuse terminology by saying that a set of clauses $F' \subseteq F$ is a core of F when it is clear from the context that F is a member of a family of parameterized contradictions and that F' can be chosen for any k and any F in that family.

Clearly, any $k+1$ positive clauses of width $f(k)$ and with pairwise disjoint sets of variables make a core that we will call a *trivial core*. It is very easy to come up with many parameterized contradictions (pebbling contradictions, colorability, sparse pigeonhole principle etc.) that possess trivial cores. Thus an interesting question is the following: do there exist "natural" parameterized contradictions that possess only non-trivial cores? And do we have a "parameterized automatizability", i.e. , is it easy to find a core once we know that it exists?

Our motivating example for the latter question is the *vertex cover* problem. A vertex cover for a graph G is a set $C \subseteq V(G)$ such that for any $\{u,v\} \in E(G)$

either $u \in C$ or $v \in C$ or both. To determine whether G has a vertex cover of size at most k there is a well-known [10, Chapter 3] fixed parameter tractable algorithm (here the parameter is k). This algorithm is based on the following observation: if a vertex is not in C, then all its neighbors must be in C. The algorithm is a simple recursive procedure which focuses on an arbitrary vertex u, and on its neighbors $v_1, \ldots, v_l$: if neither $G \setminus \{u\}$ has a vertex cover of size $k-1$ nor $G \setminus \{u, v_1, \ldots, v_l\}$ has a vertex cover of size $k-l$, then G has no vertex cover of size k.

This is interpretable as a parameterized DPLL procedure on the 2-CNF $F_G = \bigwedge_{\{u,v\} \in E(G)} (x_u \vee x_v)$ where x_u indicates whether $u \in C$. The DPLL procedure fixes an arbitrary variable x_u and branches on it. When $x_u = 1$, then the DPLL algorithm proceeds with analyzing $F_G \restriction_{x_u=1}$ which is equal to $F_{G \setminus \{u\}}$. When $x_u = 0$, then $x_{v_1} = 1, \ldots, x_{v_l} = 1$ by unit propagation. Thus the DPLL proceeds on formula $F_G \restriction_{\{x_u=0, x_{v_1}=1, \ldots, x_{v_l}=1\}} = F_{G \setminus \{u, v_1, \ldots, v_l\}}$. If at any point the DPLL has more than k variables set to one, it stops and backtracks.

And now we establish a far-reaching generalization of this example.

Theorem 3. *If F is a CNF of width d and (F, k) is a parameterized contradiction, then (F, k) has a tree-like Parameterized Resolution refutation of size $O(d^{k+1})$. Moreover, there is an algorithm that for any (F, k) either finds such tree-like refutation or finds a satisfying assignment for F of weight $\leq k$. The algorithm runs in time $O(|F| \cdot k \cdot d^{k+1})$.*

A related result was obtained in [7, Theorem 12]. Notice that while Bounded CNF Sat and Weighted CNF Sat are both W[2]-complete, Bounded d-CNF Sat is in FPT and Weighted d-CNF Sat is known to be W[1]-complete. This means that reducing the case of exact weight to bounded weight requires large clauses unless FPT = W[1]. We state two interesting consequences of Theorem 3.

Corollary 1. *For each $d \in \mathbb{N}$, the set of all parameterized contradictions in d-CNF has a core.*

Proof. The refutations constructed in Theorem 3 contain $O(d^k)$ initial clauses in $O(d^{k+1})$ variables. These clauses form a core. □

The following corollary expresses some restricted form of automatizability (cf. also the discussion in Section 5).

Corollary 2. *If $\Gamma \subseteq$ PCon has a core, then there exists an fpt-algorithm which on input $(F, k) \in \Gamma$ returns both a core and a refutation of (F, k).*

Proof. Let Γ have a core of size $f(k)$. Then the core only contains clauses of width $\leq f(k)$. On input (F, k) we run the algorithm of Theorem 3 on the CNF formula consisting of all clauses of F with width $\leq f(k)$. This yields a core together with its refutation. □

Finally, we comment on the question whether the existence of a core is a necessary condition for a parameterized contradiction to have an fpt-bounded refutation in tree-like Parameterized Resolution. A trivial counterexample to this

conjecture is made by the CNF $(x_1 \vee x_2 \vee \ldots \vee x_n) \wedge \neg x_1 \wedge \ldots \wedge \neg x_n$. A more interesting example—a version of the linear ordering principle—will be included in the full version of this paper.

5 Discussion and Open Problems

Is the parameterized proof system $\widehat{P}$ from Definition 7 the most natural way to define the parameterized analogue of P? The answer depends on the original proof system P, of course. The main (unspoken) reason why [9] defined it in this way is simply because weak proof systems cannot directly talk about the weight of the input. Let us first discuss two familiar systems that are strong enough to overcome this limitation: Frege and Cutting Planes.

The problem of getting super-polynomial lower bounds for the Frege proof system F is one of the biggest open problems in Logic and Theoretical Computer Science. Lower bounds for its parameterized version $\widehat{F}$ seem even harder to achieve for strong contradictions (as we just add new axioms). A similar conclusion remains true if we combine all parameterized axioms into one (using e.g. [6]) but allow arbitrary parameterized contradictions, not necessarily strong.

The case of Cutting Planes (CP) is way more interesting. First of all, we do not seem to know lower bounds even for the "canonical" version $\widehat{CP}$:

Question 1. Is $\widehat{CP}$ weakly fpt-bounded?

This, of course, is yet another reflection of the mysterious status of this proof system: the only known lower bounds for it are based on very indirect methods (interpolation, see [5,18]) and no direct, combinatorial proof is currently known. And if we try to generalize the methods from [5,18] (at least in a straightforward way) then we immediately arrive at a problem in parameterized circuit complexity that seems to be widely open (at least, we do not see how known methods can be applied to it).

Question 2. Find an explicit *partial* monotone function (a.k.a. a monotone promise problem) $f : [n]^{\leq k} \to \{0,1\}$ defined only on inputs of Hamming weight $\leq\ k$ that does not possess monotone circuits of size $f(k)n^{O(1)}$.

The problem of finding (say) a $\sqrt{k}$-clique does become easy in this context.

For weaker proof systems, [9, Section 4] proposed to use auxiliary variables. Their suggestion was to add new "pigeonhole variables" $p_{i,j}$ $(i \in [n],\ j \in [k])$ and "pigeonhole clauses" $\neg x_i \vee \bigvee_{j\in[k]} p_{i,j}$ for all $i \in [n]$ (pigeon clauses) and $\neg p_{i_1,j} \vee \neg p_{i_2,j}$ for all $i_1 \neq i_2 \in [n],\ j \in [k]$ (hole clauses), where $x_1, \ldots, x_n$ are the original variables. Remarkably, they proved that the pigeonhole principle has fpt-bounded refutations in this version of Parameterized Resolution.

The disturbing Example 1 turns into an instance of PHP_k^{k+1} with large "metapigeons" that has an fpt-bounded proof (e.g., the straightforward adaption of the rectangular proof from [19, Example 1]). Thus, following [9], we ask:

Question 3. Is Parameterized Resolution with auxiliary variables fpt-bounded?

Let us now point out that there is an interesting and well-studied class of contradictions for which the difference between these two encodings disappears, and these are independent set principles. Following [2], let G be a graph $[n]$ in which vertices are split into k subsets $V_1, \ldots, V_k$ of size n/k each called *blocks*. The principle $\alpha_{block}(G,k)$ encodes the fact that G has a block-respecting independent set of size k; it has the variables x_v ($v \in [n]$) and the axioms $\neg x_u \vee \neg x_v$ for all $(u,v) \in E(G)$ (edge clauses), $\bigvee_{v \in V_i} x_v$ for all $i \in [k]$ (block clauses), and $\neg x_u \vee \neg x_v$ for all $u \neq v$ in the same block (1–1 clauses).

The fact that all satisfying assignments have at most k ones is already built in this principle: all parameterized axioms are subsumed by the 1–1 clauses above. Auxiliary clauses in the sense of [9] (both pigeon and holes) also do not help to reduce the refutation size, as witnessed by the substitution of the pigeonhole variables $p_{v,j} \mapsto 0$ if $v \notin V_j$ and $p_{v,j} \mapsto x_v$ if $v \in V_j$. Thus, we are also asking the following specific form of Question 3:

Question 4. Do the principles $\alpha_{block}(G,k)$ always have fpt-bounded Resolution refutations as long as the graph G does not contain block-respecting independent sets of size k?

One good candidate for a lower bound here would be Erdös-Rényi random graphs $G(n,p)$ for an appropriately chosen value of p. Such lower bound has been recently proved for tree-like Resolution in [4].

Let us recall that a proof system P is *automatizable* if there exists an algorithm which for a tautology F with a P-proof of size S finds a P-proof for F of size at most $S^{O(1)}$ and runs in time $S^{O(1)}$. Alekhnovich and Razborov [1] proved that if (classical) Resolution or tree-like Resolution were automatizable, then $\mathsf{W[P]}$ would coincide with FPR, the randomized version of FPT. On the other hand, tree-like Resolution is quasi-polynomially automatizable (see e.g. [3]).

We point out that the concept of quasi-polynomial automatizability is meaningless in the context of Parameterized Resolution, because every $(F,k) \in \mathrm{PCon}$ with $|F| = n$ has a refutation of size $c \cdot \binom{n}{k+1}$ for some constant c. If $k \leq \log n$ this is smaller than $n^{\log n}$; otherwise $\binom{n}{k+1} \leq 2^{(k+1)^2}$ which is fpt with respect to k. On the contrary, the concept of polynomial automatizability can be extended to parameterized proof systems in an obvious way. Thus, we ask:

Question 5. Is (tree-like) Parameterized Resolution, with or without auxiliary variables, fpt-automatizable or fpt-automatizable w.r.t. strong contradictions? That is, does there exist an algorithm that for any (strong) parameterized contradiction $(F,k) \in \mathrm{PCon}$ outputs its refutation within time $f(k)S^{O(1)}$, where S is the minimal possible size of a parameterized refutation of (F,k)?

Naturally, unconditional results of this sort are completely out of reach for the moment, so we are willing to allow here any reasonable complexity assumption (that will most likely reside in the realm of parameterized complexity itself).

References

1. Alekhnovich, M., Razborov, A.A.: Resolution is not automatizable unless W(P) is tractable. SIAM Journal on Computing 38(4), 1347–1363 (2008)
2. Beame, P., Impagliazzo, R., Sabharwal, A.: The resolution complexity of independent sets and vertex covers in random graphs. Computational Complexity 16(3), 245–297 (2007)
3. Beame, P., Karp, R.M., Pitassi, T., Saks, M.E.: The efficiency of resolution and Davis–Putnam procedures. SIAM J. Comput. 31(4), 1048–1075 (2002)
4. Beyersdorff, O., Galesi, N., Lauria, M.: Parameterized complexity of DPLL search procedures. In: Proc. 14th International Conference on Theory and Applications of Satisfiability Testing (to appear, 2011)
5. Bonet, M.L., Pitassi, T., Raz, R.: Lower bounds for cutting planes proofs with small coefficients. The Journal of Symbolic Logic 62(3), 708–728 (1997)
6. Buss, S.R.: Polynomial size proofs of the propositional pigeonhole principle. The Journal of Symbolic Logic 52, 916–927 (1987)
7. Chen, Y., Flum, J.: The parameterized complexity of maximality and minimality problems. Annals of Pure and Applied Logic 151(1), 22–61 (2008)
8. Cook, S.A., Reckhow, R.A.: The relative efficiency of propositional proof systems. The Journal of Symbolic Logic 44(1), 36–50 (1979)
9. Dantchev, S.S., Martin, B., Szeider, S.: Parameterized proof complexity. In: Proc. 48th IEEE Symposium on the Foundations of Computer Science, pp. 150–160 (2007)
10. Downey, R.G., Fellows, M.R.: Parameterized Complexity. Springer, Heidelberg (1999)
11. Flum, J., Grohe, M.: Describing parameterized complexity classes. Information and Computation 187(2), 291–319 (2003)
12. Flum, J., Grohe, M.: Parameterized Complexity Theory. Springer, Heidelberg (2006)
13. Gao, Y.: Data reductions, fixed parameter tractability, and random weighted d-CNF satisfiability. Artificial Intelligence 173(14), 1343–1366 (2009)
14. Haken, A.: The intractability of resolution. Theor. Comput. Sci. 39, 297–308 (1985)
15. Krajíček, J., Pudlák, P., Woods, A.: Exponential lower bounds to the size of bounded depth Frege proofs of the pigeonhole principle. Random Structures and Algorithms 7(1), 15–39 (1995)
16. Niedermeier, R.: Invitation to Fixed-Parameter Algorithms. Oxford Lecture Series in Mathematics and Its Applications. Oxford University Press, Oxford (2006)
17. Pitassi, T., Beame, P., Impagliazzo, R.: Exponential lower bounds for the pigeonhole principle. Computational Complexity 3, 97–140 (1993)
18. Pudlák, P.: Lower bounds for resolution and cutting planes proofs and monotone computations. The Journal of Symbolic Logic 62(3), 981–998 (1997)
19. Razborov, A., Wigderson, A., Yao, A.: Read-once branching programs, rectangular proofs of the pigeonhole principle and the transversal calculus. Combinatorica 22(4), 555–574 (2002)
20. Riis, S.: A complexity gap for tree resolution. Computational Complexity 10(3), 179–209 (2001)

On Minimal Unsatisfiability and Time-Space Trade-offs for k-DNF Resolution

Jakob Nordström[1] and Alexander Razborov[2]

[1] KTH Royal Institute of Technology, SE-100 44 Stockholm, Sweden
[2] Department of Computer Science, University of Chicago, Chicago, IL 60637, USA

Abstract. A well-known theorem by Tarsi states that a minimally unsatisfiable CNF formula with m clauses can have at most $m-1$ variables, and this bound is exact. In the context of proving lower bounds on proof space in k-DNF resolution, [Ben-Sasson and Nordström 2009] extended the concept of minimal unsatisfiability to sets of k-DNF formulas and proved that a minimally unsatisfiable k-DNF set with m formulas can have at most $(mk)^{k+1}$ variables. This result is far from tight, however, since they could only present explicit constructions of minimally unsatisfiable sets with $\Omega(mk^2)$ variables.

In the current paper, we revisit this combinatorial problem and significantly improve the lower bound to $(\Omega(m))^k$, which almost matches the upper bound above. Furthermore, using similar ideas we show that the analysis of the technique in [Ben-Sasson and Nordström 2009] for proving time-space separations and trade-offs for k-DNF resolution is almost tight. This means that although it is possible, or even plausible, that stronger results than in [Ben-Sasson and Nordström 2009] should hold, a fundamentally different approach would be needed to obtain such results.

1 Introduction

A formula in conjunctive normal form, or *CNF formula*, is said to be *minimally unsatisfiable* if it is unsatisfiable but deleting any clause makes it satisfiable. A well-known result by Tarsi [1], reproven several times by various authors (see, for instance, [6, 12, 15]), says that the number of variables in any such CNF formula is always at most $m-1$, where m is the number of clauses.

Motivated by problems in proof complexity related to the space measure in the *k-DNF resolution* proof systems introduced by Krajíček [14], Ben-Sasson and Nordström generalized this concept in [9]. In that paper, later published as part of [10], they studied the minimal unsatisfiability of conjunctions of formulas in disjunctive normal form where all terms in the disjunctions have size at most k, henceforth *k-DNF formulas*. We begin by reviewing their definition.

Assume that $\mathbb{D} = \{D_1, \ldots, D_m\}$ is the set of k-DNF formulas appearing in our conjunction, and that $\mathbb{D}$ itself is unsatisfiable. What should it mean that $\mathbb{D}$ is *minimally* unsatisfiable?

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 642–653, 2011.

The first, naive, attempt at a definition would be to require, by analogy with the $k = 1$ case, that $\mathbb{D}$ becomes satisfiable after removing any D_i from it. However, the following simple example of two 2-DNF formulas

$$\{(x \wedge y_1) \vee \ldots \vee (x \wedge y_n),\ (\bar{x} \wedge y_1) \vee \ldots \vee (\bar{x} \wedge y_n)\} \tag{1}$$

that is minimally unsatisfiable in this sense shows that we can not hope to get any meaningful analogue of Tarsi's lemma under this assumption only.

The reason for this is that the 2-DNF set (1) is *not* minimally unsatisfiable in the following sense: even if we "weaken" a formula in the set (i.e., make it easier to satisfy) by removing any, or even all, y-variables, what remains is still an unsatisfiable set. This leads us to the stronger (and arguably more natural) notion that the formula set should be minimally unsatisfiable not only with respect to removing DNF formulas but also with respect to shrinking terms (i.e., conjunctions) in these formulas. Fortunately, this also turns out to be just the right notion for the proof complexity applications given in [10]. Therefore, following [10], we say that a set $\mathbb{D}$ of k-DNF formulas is *minimally unsatisfiable* if weakening any single term (i.e., removing from it any literal) appearing in a k-DNF formula from $\mathbb{D}$ will make the "weaker" set of formulas satisfiable. This raises the following combinatorial question:

> *How many variables (as a function of k and m) can appear in a minimally unsatisfiable set $\{D_1, \ldots, D_m\}$ of k-DNF formulas?*

Tarsi's lemma states that for $k = 1$ the answer is $m - 1$. This result has a relatively elementary proof based on Hall's marriage theorem, but its importance to obtaining lower bounds on resolution length and space is hard to overemphasize. For instance, the seminal lower bound on refutation length of random CNF formulas in [12] makes crucial use of it, as does the proof of the "size-width trade-off" in [11]. Examples of applications of this theorem in resolution space lower bounds include [3, 7, 8, 10, 16, 18].

To the best of our knowledge, the case $k \geq 2$ had not been studied prior to [10]. That paper established an $(mk)^{k+1}$ upper bound and an $\Omega(mk^2)$ lower bound on the number of variables. The gap is large, and, as one of their open questions, the authors asked to narrow it.

In this paper, we almost completely resolve this problem by proving an $(\Omega(m))^k$ lower bound on the number of variables. Our construction is given in Section 3, following some preliminaries in Section 2. In Section 4, we show how a similar construction proves that in order to improve on the space complexity bounds from [10] a different approach would be needed. The paper is concluded with a few remarks and open problems in Section 5.

2 Preliminaries

Recall that a DNF formula is a disjunction of terms, or conjunctions, of literals, i.e., unnegated or negated variables. If all terms have size at most k, then the formula is referred to as a *k-DNF formula* (where k should be thought of as some arbitrary but fixed constant).

Definition 1 ([10]). *A set of DNF formulas $\mathbb{D}$ is* minimally unsatisfiable *if it is unsatisfiable but replacing any single term T appearing in any DNF formula $D \in \mathbb{D}$ with any proper subterm of T makes the resulting set satisfiable.*

Note that this indeed generalizes the well-known notion of minimally unsatisfiable CNF formulas, where a "proper subterm" of a literal is the empty term 1 that is always true and "weakening" a clause hence corresponds to removing it from the formula.

We are interested in bounding the number of variables of a minimally unsatisfiable k-DNF set in terms of the number of formulas in the set. For 1-DNF sets (i.e., CNF formulas), Tarsi's lemma says that the number of variables must be at most the number of formulas (i.e., clauses) minus one for minimal unsatisfiability to hold. This is easily seen to be tight by considering the example

$$\{x_1, x_2, \ldots, x_n,\ \bar{x}_1 \vee \bar{x}_2 \vee \ldots \vee \bar{x}_n\}\ . \tag{2}$$

No such bound holds for general k, however, since there is an easy construction shaving off a factor k^2. Namely, denoting by $\mathit{Vars}(\mathbb{D})$ the set of variables appearing somewhere in $\mathbb{D}$, we have the following lemma.

Lemma 2 ([10]). *There are arbitrarily large minimally unsatisfiable sets $\mathbb{D}$ of k-DNF formulas with $|\mathit{Vars}(\mathbb{D})| \geq k^2(|\mathbb{D}| - 1)$.*

Proof sketch. Consider any minimally unsatisfiable CNF formula consisting of $n + 1$ clauses over n variables (for example, the one given in (2)). Substitute every variable x_i with

$$\left(x_i^1 \wedge \cdots \wedge x_i^k\right) \vee \left(x_i^{k+1} \wedge \cdots \wedge x_i^{2k}\right) \vee \cdots \vee \left(x_i^{k^2-k+1} \wedge \cdots \wedge x_i^{k^2}\right) \tag{3}$$

and expand every clause to a k-DNF formula. (Note that for this to work, we also need the easily verifiable fact that the negation of (3) can be expressed as a k-DNF formula.) It is straightforward to verify that the result is a minimally unsatisfiable k-DNF set, and this set has $n + 1$ formulas over $k^2 n$ variables.

There is a big gap between this lower bound on the number of variables (in terms of the number of formulas) and the upper bound obtained in [10], stated next.

Theorem 3 ([10]). *Suppose that $\mathbb{D}$ is a minimally unsatisfiable k-DNF set containing m formulas. Then $|\mathit{Vars}(\mathbb{D})| \leq (km)^{k+1}$.*

A natural problem is to close, or at least narrow, this gap. In this work, we do so by substantially improving the bound in Lemma 2.

3 An Improved Lower Bound for Minimal Unsatisfiability

In this section, we present our construction establishing that the number of variables in a minimally unsatisfiable k-DNF set can be roughly at least the number of formulas raised to the kth power.

Theorem 4. *There exist arbitrarily large minimally unsatisfiable k-DNF sets $\mathbb{D}$ with m formulas over more than $\left(\frac{m}{4}\left(1-\frac{1}{k}\right)\right)^k$ variables.*

In particular, for any $k \geq 2$ there are minimally unsatisfiable k-DNF sets with m formulas over (more than) $(m/8)^k = (\Omega(m))^k$ variables.

Very informally, we will use the power afforded by the k-terms to construct a k-DNF set $\mathbb{D}$ consisting of roughly m formulas that encode roughly m^{k-1} "parallel" instances of the minimally unsatisfiable CNF formula in (2). These parallel instances will be indexed by coordinate vectors $\left(x^1_{i_1}, x^2_{i_2}, \ldots, x^{k-1}_{i_{k-1}}\right)$. We will add auxiliary formulas enforcing that only one coordinate vector $\left(x^1_{i_1}, x^2_{i_2}, \ldots, x^{k-1}_{i_{k-1}}\right)$ can have all coordinates true. This vector identifies which instance of the formula (2) we are focusing on, and all other parallel instances are falsified by their coordinate vectors not having all coordinates true.

Let us now formalize this intuition. We first present the auxiliary formulas constraining our coordinate vectors, which are the key to the whole construction.

3.1 A Weight Constraint k-DNF Formula Set

Let us write $\boldsymbol{x} = \left(x_1, \ldots, x_{m(k-1)}\right)$ to denote a vector of variables of dimension $m(k-1)$. Let $|\boldsymbol{x}| = \sum_{i=1}^{m(k-1)} x_i$ denote the *Hamming weight* of $\boldsymbol{x}$, i.e., the number of ones in the vector. We want to construct a k-DNF set $W_m(\boldsymbol{x})$ with $\mathrm{O}(m)$ formulas over $x_1, \ldots, x_{m(k-1)}$ and some auxiliary variables minimally expressing that $|\boldsymbol{x}| \leq 1$. That is, a vector $\boldsymbol{x}$ can be extended to a satisfying assignment for $W_m(\boldsymbol{x})$ if and only if $|\boldsymbol{x}| \leq 1$ but if we weaken any formula in the set, then there are satisfying assignments with $|\boldsymbol{x}| \geq 2$.

We define $W_m(\boldsymbol{x})$ to be the set of k-DNF formulas listed in Figure 1. The intuition for the auxiliary variables is that z_j can be set to true only if the first $j(k-1)$ variables $x_1, \ldots, x_{j(k-1)}$ are all false, and w_j can be set to true only if at most one of the first $j(k-1)$ variables $x_1, \ldots, x_{j(k-1)}$ is true. The set W_m contains $2m-1$ formulas. Let us see that W_m minimally expresses that $\boldsymbol{x}$ has weight at most 1. For ease of notation, we will call the group of variables $\{x_{(j-1)(k-1)+1}, \ldots, x_{j(k-1)}\}$ the *jth block* and denote it by X_j.

Every $\boldsymbol{x}$ with $|\boldsymbol{x}| \leq 1$ can be extended to a satisfying assignment for $W_m(\boldsymbol{x})$. Since all x-variables appear only negatively, we can assume without loss of generality that $|\boldsymbol{x}| = 1$, so that all x_i are false except for a single variable in, say, the j_0th block X_{j_0}. We simply set z_j to true for $j < j_0$ and false for $j \geq j_0$, and we set all w_j to true.

Every satisfying assignment for $W_m(\boldsymbol{x})$ satisfies $|\boldsymbol{x}| \leq 1$. Assume on the contrary that $x_{i_1} = x_{i_2} = 1$; $i_1 \in X_{j_1}$, $i_2 \in X_{j_2}$; $j_1 \leq j_2$. We have that the truth of x_{i_1} forces z_j to false for all $j \geq j_1$, and then $x_{i_2} = 1$ forces w_j to false for all $j \geq j_2$. But this means that there is no way to satisfy the final formula (4g). So for all satisfying assignments it must hold that $|\boldsymbol{x}| \leq 1$.

After weakening any term in $W_m(\boldsymbol{x})$, the resulting set can be satisfied by an assignment giving weight at least 2 to $\boldsymbol{x}$. First, notice that weakening

$$\overline{z}_1 \vee \left(\overline{x}_1 \wedge \cdots \wedge \overline{x}_{k-1}\right) \tag{4a}$$

$$\overline{z}_2 \vee \left(z_1 \wedge \overline{x}_k \wedge \cdots \wedge \overline{x}_{2(k-1)}\right) \tag{4b}$$

$$\vdots$$

$$\overline{z}_{m-1} \vee \left(z_{m-2} \wedge \overline{x}_{(m-2)(k-1)+1} \wedge \cdots \wedge \overline{x}_{(m-1)(k-1)}\right) \tag{4c}$$

$$\overline{w}_1 \vee z_1 \vee \bigvee_{i=1}^{k-1} \bigwedge_{\substack{i'=1 \\ i' \neq i}}^{k-1} \overline{x}_{i'} \tag{4d}$$

$$\overline{w}_2 \vee z_2 \vee \left(w_1 \wedge \overline{x}_k \wedge \cdots \overline{x}_{2(k-1)}\right) \vee \bigvee_{i=k}^{2(k-1)} \left(z_1 \wedge \bigwedge_{\substack{i'=k \\ i' \neq i}}^{2(k-1)} \overline{x}_{i'}\right) \tag{4e}$$

$$\vdots$$

$$\overline{w}_{m-1} \vee z_{m-1} \vee \left(w_{m-2} \wedge \overline{x}_{(m-2)(k-1)+1} \wedge \cdots \wedge \overline{x}_{(m-1)(k-1)}\right) \vee \bigvee_{i=(m-2)(k-1)+1}^{(m-1)(k-1)} \left(z_{m-2} \wedge \bigwedge_{\substack{i'=(m-2)(k-1)+1 \\ i' \neq i}}^{(m-1)(k-1)} \overline{x}_{i'}\right) \tag{4f}$$

$$\left(w_{m-1} \wedge \overline{x}_{(m-1)(k-1)+1} \wedge \cdots \wedge \overline{x}_{m(k-1)}\right) \vee \bigvee_{i=(m-1)(k-1)+1}^{m(k-1)} \left(z_{m-1} \wedge \bigwedge_{\substack{i'=(m-1)(k-1)+1 \\ i' \neq i}}^{m(k-1)} \overline{x}_{i'}\right) \tag{4g}$$

Fig. 1. Weight constraint k-DNF formulas $W_m(\boldsymbol{x})$

any of the unit terms (i.e., terms of size one) results in removing the formula in question altogether. This can only make it easier to satisfy the whole set than if we just shrink a k-term. Hence, without loss of generality we can focus on shrinking k-terms. Let us consider the formulas in $W_m(\boldsymbol{x})$ one by one.

If we remove some literal $\overline{x}_i$ in (4a)–(4c), then we can set $x_i = 1$ but still have $z_1 = \cdots = z_{m-1} = 1$. This will allows us to set also $x_{m(k-1)} = 1$ in (4g) and still satisfy the whole set of formulas although $|\boldsymbol{x}| \geq 2$.

If we instead remove some z_j $(j \leq m-2)$ in these formulas, then we can set all $x_i = 1$ for $x_i \in X_1 \cup \ldots \cup X_j$ (that already gives us weight ≥ 2) and $z_1 = \ldots = z_j = 0$, and then we set $z_{j+1} = \ldots = z_{m-1} = 1$ and $x_i = 0$ for $x_i \in X_{j+1} \ldots \cup \ldots X_m$. Note that $j \leq m-2$ implies that $z_{m-1} = 1$ which takes care of (4g), and then (4d)–(4f) are satisfied simply be setting all w_j to 0. This completes the analysis of the formulas (4a)–(4c).

In formula (4d), if we remove some $\overline{x}_{i'}$, then we can set $x_i = x_{i'} = w_1 = 1$ and extend this to a satisfying assignment for the rest of the formulas.

For the corresponding terms $z_{j-1} \wedge \bigwedge_{i'=(j-1)(k-1)+1,\ i' \neq i}^{j(k-1)} \overline{x}_j$ in (4e)–(4g), if we remove some $\overline{x}_{i'}$, we can again set $x_i = x_{i'} = 1$ and $z_1 = \ldots = z_{j-1} = 1$

$$W_m^j(\boldsymbol{x}^j) \qquad 1 \le j < k \tag{5a}$$

$$\bigvee_{\substack{(i_1,\ldots,i_{k-1})\\ \in[m(k-1)]^{k-1}}} \left(x^1_{i_1} \wedge \cdots \wedge x^{k-1}_{i_{k-1}} \wedge y^{\nu}_{i_1,\ldots,i_{k-1}}\right) \qquad 1 \le \nu \le m(k-1) \tag{5b}$$

$$\bar{u}_\nu \vee \bigvee_{\substack{(i_1,\ldots,i_{k-1})\\ \in[m(k-1)]^{k-1}}} \left(x^1_{i_1} \wedge \cdots \wedge x^{k-1}_{i_{k-1}} \wedge \overline{y}^{\nu}_{i_1,\ldots,i_{k-1}}\right) \qquad 1 \le \nu \le m(k-1) \tag{5c}$$

$$u_1 \vee \cdots \vee u_{m(k-1)} \tag{5d}$$

Fig. 2. Minimally unsatisfiable set of k-DNF formulas $\mathbb{D}^k_m$

and then $w_j = \ldots = w_{m-1} = 1$ to satisfy the rest of the set, whereas removing z_{j-1} would allow us to assign to 1 all $x_i \in X_1 \cup \ldots \cup X_{j-1}$ and then still assign $w_j = \ldots = w_{m-1} = 1$.

For the other kind of terms $w_{j-1} \wedge \overline{x}_{(j-1)(k-1)+1} \wedge \cdots \wedge \overline{x}_{j(k-1)}$ in (4e)–(4g), if some $\overline{x}_i$ with $x_i \in X_j$ is removed, we can set this x_i to true as well as an arbitrary $x_{i'} \in X_1 \cup \ldots \cup X_{j-1}$, whereas removing w_{j-1} would allow as again to set to 1 all variables in $X_1 \cup \ldots X_{j-1}$. This proves the minimality of $W_m(\boldsymbol{x})$.

3.2 The Minimally Unsatisfiable k-DNF Set

Let us write $\boldsymbol{x}^j = \left(x^j_1, x^j_2, \ldots, x^j_{m(k-1)}\right)$, and let $W^j_m(\boldsymbol{x}^j)$ be the k-DNF set with $\mathrm{O}(m)$ formulas constructed above (over disjoint sets of variables for distinct j) minimally expressing that $|\boldsymbol{x}^j| \le 1$. With this notation, let $\mathbb{D}^k_m$ be the k-DNF set consisting of the formulas in Figure 2. It is worth noting that the range of the index ν does not have any impact on the following proof of minimal unsatisfiability, and it was set to $m(k-1)$ only to get the best numerical results.

It is easy to verify that $\mathbb{D}^k_m$ consists of less than $4mk$ k-DNF formulas over more than $(m(k-1))^k = \left(\frac{1}{4}(4mk)\left(1-\frac{1}{k}\right)\right)^k$ variables. We claim that $\mathbb{D}^k_m$ is minimally unsatisfiable, from which Theorem 4 follows.

To prove the claim, let us first verify unsatisfiability. If the k-DNF formulas $W^j_m(\boldsymbol{x})$ in (5a) are to be satisfied for all $j < k$, then there exists at most one $(k-1)$-tuple $(i^*_1, i^*_2, \ldots, i^*_{k-1}) \in [m(k-1)]^{k-1}$ such that $x^1_{i^*_1}, x^2_{i^*_2}, \ldots, x^{k-1}_{i^*_{k-1}}$ are all true. This forces $y^{\nu}_{(i^*_1,i^*_2,\ldots,i^*_{k-1})}$ to true for all ν to satisfy the formulas in (5b), and then (5c) forces all u_ν to 0, so that (5d) is falsified. Hence, $\mathbb{D}^k_m$ is unsatisfiable.

Let us now argue that $\mathbb{D}^k_m$ is not only unsatisfiable, but *minimally* unsatisfiable in the sense of Definition 1. The proof is by case analysis over the different types of formulas in $\mathbb{D}^k_m$.

1. If we shrink any term in (5a)—say, in $W^1_m(\boldsymbol{x}^1)$—then by the minimality property in Section 3.1 we can set some $x^1_{i'_1} = x^1_{i''_1} = 1$ for $i'_1 \neq i''_1$ and fix some $x^2_{i^*_2} = \ldots = x^{k-1}_{i^*_{k-1}} = 1$ without violating the remaining clauses in

$W_m^1(\boldsymbol{x}^1), \ldots, W_m^{k-1}(\boldsymbol{x}^{k-1})$. This allows us to satisfy the formulas in (5b) and (5c) by setting $y^\nu_{(i'_1,i^*_2\ldots,i^*_{k-1})} = 1$ and $y^\nu_{(i''_1,i^*_2\ldots,i^*_{k-1})} = 0$ for all ν. Finally, set any u_ν to true to satisfy (5d). This satisfies the whole k-DNF set.

2. Next, suppose that we shrink some term $x^1_{i^*_1} \wedge x^2_{i^*_2} \wedge \cdots \wedge x^{k-1}_{i^*_{k-1}} \wedge y^\nu_{(i^*_1,\ldots,i^*_{k-1})}$ in the νth k-DNF formula in (5b). There are two subcases:
   (a) Some x-variable is removed, say, the variable $x^1_{i^*_1}$. Set $x^1_{i^*_1} = 0$ and $x^2_{i^*_2} = \ldots = x^{k-1}_{i^*_{k-1}} = y^\nu_{(i^*_1,i^*_2,\ldots,i^*_{k-1})} = 1$. This satisfies the νth formula in (5b). Then pick some $i'_1 \neq i^*_1$ and set $x^1_{i'_1} = 1$. All this can be done in a way that satisfies all clauses in (5a) since the weight of every $\boldsymbol{x}^j$ is one. Set $u_\nu = 1$ and $u_{\nu'} = 0$ for all $\nu' \neq \nu$ to satisfy (5d) and then $y^\nu_{(i'_1,i^*_2,\ldots,i^*_{k-1})} = 0$ to satisfy the νth formula in (5c) (all others are satisfied by literals $\bar{u}_{\nu'}$, $\nu' \neq \nu$). The νth formula in (5b) was satisfied above, and for all other $\nu' \neq \nu$ we set $y^{\nu'}_{(i'_1,i^*_2,\ldots,i^*_{k-1})} = 1$ to satisfy the rest of the formulas in (5b). This satisfies the whole k-DNF set.
   (b) The variable $y^\nu_{(i^*_1,\ldots,i^*_{k-1})}$ is eliminated. If so, set $x^1_{i^*_1} = \ldots = x^{k-1}_{i^*_{k-1}} = 1$ to satisfy the νth formula in (5b), $u_\nu = 1$ and $y^\nu_{(i^*_1,\ldots,i^*_{k-1})} = 0$ to satisfy (5d) and the νth formula in (5c), and $u_{\nu'} = 0$ and $y^{\nu'}_{(i^*_1,\ldots,i^*_{k-1})} = 1$ for all $\nu' \neq \nu$ to satisfy the rest of the formulas in (5b) and (5c). This is easily extended to an assignment satisfying (5a) as well.
3. For the νth formula in (5c), we may assume, for the same reasons as in Section 3.1, that we shrink a non-trivial k-term. Then we again have two subcases, treated similarly.
   (a) Some x-variable is removed, say $x^1_{i^*_1}$. Set $u_\nu = 1$, $x^1_{i^*_1} = 0$, $x^2_{i^*_2} = \ldots = x^{k-1}_{i^*_{k-1}} = 1$, and $y^\nu_{(i^*_1,i^*_2,\ldots,i^*_{k-1})} = 0$. This satisfies (5d) and the νth formula in (5c). Setting $u_{\nu'} = 0$ for $\nu' \neq \nu$ takes care of the rest of (5c). To satisfy (5b), pick some $i'_1 \neq i^*_1$ and set $x^1_{i'_1} = 1$, and $y^{\nu'}_{(i'_1,i^*_2,\ldots,i^*_{k-1})} = 1$ for all ν'. These assignments are all consistent with the weight constraints in (5a).
   (b) The literal $\overline{y}^\nu_{(i^*_1,\ldots,i^*_{k-1})}$ is eliminated. If so, set $x^1_{i^*_1} = \ldots = x^{k-1}_{i^*_{k-1}} = 1$ to satisfy the νth formula in (5c) and $u_\nu = 1$ to satisfy (5d). Setting $u_{\nu'} = 0$ for $\nu' \neq \nu$ takes care of the rest of (5c). Now we can satisfy all of (5b) by setting $y^{\nu'}_{(i^*_1,\ldots,i^*_{k-1})} = 1$ for all ν', and it is once again easy to see that the weight constraints in (5a) are also satisfied.
4. The disjunctive clause (5d) is removed. Set all u_ν to 0, and then set all $y^\nu_{i_1,\ldots,i_k}$ to 1, then (5a)–(5b) become easy to satisfy.

This completes the proof that $\mathbb{D}^k_m$ is minimally unsatisfiable as claimed, and Theorem 4 hence follows.

4 Implications for k-DNF Resolution Trade-offs

Let us start this section by a quick review of the relevant proof complexity context. The *k-DNF resolution* proof systems were introduced by Krajíček [14]

as an intermediate step between resolution and depth-2 Frege. Roughly speaking, the kth member of this family, denoted henceforth by $\mathfrak{R}(k)$, is a system for reasoning in terms of k-DNF formulas. For $k = 1$, the lines in the proof are hence disjunctions of literals, and the system $\mathfrak{R}(1)$ is standard resolution. At the other extreme, $\mathfrak{R}(\infty)$ is equivalent to depth-2 Frege.

Informally, we can think of an $\mathfrak{R}(k)$-proof as being presented on a blackboard. The allowed derivation steps are to write on the board a clause of the CNF formula being refuted, to deduce a new k-DNF formula from the formulas currently on the board, or to erase formulas from the board. The *length* of an $\mathfrak{R}(k)$-proof is the total number of formulas appearing on the board (counted with repetitions) and the *(formula) space* is the maximal number of formulas simultaneously on the board at any time during the proof.

A number of works, for example, [2, 4, 5, 19, 20, 21], have proven superpolynomial lower bounds on the length of k-DNF refutations. It was also shown in [20, 21] that the $\mathfrak{R}(k)$-family forms a strict hierarchy with respect to proof length. Just as in the case for standard resolution, however, our understanding of space complexity in k-DNF resolution has remained more limited. Esteban et al. [13] established essentially optimal space lower bounds for $\mathfrak{R}(k)$ and also proved that the family of *tree-like* $\mathfrak{R}(k)$ systems form a strict hierarchy with respect to space. They showed that there are formulas F_n of size n that can be refuted in tree-like $(k+1)$-DNF resolution in constant space but require space $\Omega(n/\log^2 n)$ in tree-like k-DNF resolution. It should be pointed out, however, that tree-like $\mathfrak{R}(k)$ for any $k \geq 1$ is strictly weaker than standard resolution, so the results in [13] left open the question of whether there is a strict space hierarchy for (non-tree-like) k-DNF resolution or not.

Recently, the first author in joint work with Ben-Sasson [10] proved that Krajíček's family of $\mathfrak{R}(k)$ systems do indeed form a strict hierarchy with respect to space. However, the parameters of the separation were much worse than for the tree-like systems in [13]; namely, the $\mathfrak{R}(k+1)$-proofs have constant space but any $\mathfrak{R}(k)$-proof requires space $\Omega\big(\sqrt[k+1]{n/\log n}\big)$. It is not clear that there has to be a $(k+1)$st root in this bound. No matching upper bounds are known, and indeed for the special case of $\mathfrak{R}(2)$ versus $\mathfrak{R}(1)$ the lower bound proven in [10] is $\Omega\big(n/\log n\big)$, i.e., without a square root. Also, [10] established strong length-space trade-offs for k-DNF resolution, but again a $(k+1)$st root is lost in the analysis compared to the corresponding results for standard resolution $\mathfrak{R}(1)$.

Returning now to the minimally unsatisfiable k-DNF sets, the reason [10] studied this concept was that it appeared related to a problem arising in their proof analysis, and they hoped that better *upper* bounds for minimal unsatisfiability would translate into improvements in the analysis. Instead, using ideas from the improved *lower* bound construction for minimal unsatisfiability in the previous section, we can show that the analysis of the particular proof technique employed in [10] is almost tight. Thus, any further substantial improvements of the bounds in that paper would have to be obtained by other methods.

We do not go into details of the proof construction in [10] here, since it is rather elaborate. Suffice it to say that the final step of the proof boils down to

studying k-DNF sets that imply Boolean functions with a particular structure, and proving lower bounds on the size of such DNF sets in terms of the number of variables in these Boolean functions. (Recall that a set $\mathbb{F}$ implies a function F, denoted $\mathbb{F} \models F$, if any satisfying truth value assignment to all of $\mathbb{F}$ must also satisfy F.) Having come that far in the construction, all that remains is a purely combinatorial problem, and no reference to space proof complexity or k-DNF resolution is needed.

For concreteness, below we restrict our attention to the case where the Boolean functions are exclusive or. More general functions can be considered, and have been studied in [10], and everything that will be said below applies to such Boolean functions with appropriate (and simple) modifications. Hence, from now on let us focus on DNF sets that "minimally imply" (in a sense made formal below) a particular kind of formulas that we will refer to as $\left(\wedge\vee\oplus^k\right)$*-block formulas*. A $\left(\wedge\vee\oplus^k\right)$-block formula is a CNF formula in which every variable x is replaced by $\bigoplus_{i=1}^{k} x_i$, where $x_1, \ldots, x_k$ are new variables. Thus, literals turn into unnegated or negated XORs, every XOR applies to exactly one "block" of k variables, and no XOR mixes variables from different blocks. Let us write this down as a formal definition.

Definition 5. *A* $\left(\wedge\vee\oplus^k\right)$-block formula G *is a conjunction of disjunctions of negated or unnegated exclusive ors. The variables of* G *are divided into disjoint blocks* $x_1, \ldots, x_k$, $y_1, \ldots, y_k$, $z_1, \ldots, z_k$ *et cetera, of* k *variables each, and every XOR or negated XOR is over one full block of variables.*

The key behind the lower bounds on space in [10] is the result that if a k-DNF set $\mathbb{D}$ implies a $\left(\wedge\vee\oplus^{k+1}\right)$-block formula G with many variables, then $\mathbb{D}$ must also be large.

Theorem 6 ([10]). *Let* k *be some fixed but arbitrary positive integer. Suppose* $\mathbb{D}$ *is a* k*-DNF set and* G *is a* $\left(\wedge\vee\oplus^{k+1}\right)$*-block formula such that* $\mathbb{D}$ *implies* G *"precisely," in the sense that if we remove a single XOR or negated XOR from* G *(thus making the formula stronger, i.e., harder to satisfy), it no longer holds that* $\mathbb{D}$ *implies* G*. Then* $|\mathit{Vars}(G)| = \mathrm{O}\left(|\mathbb{D}|^{k+1}\right)$.

Using this theorem, one can get the $\sqrt[k+1]{n/\log n}$ space separation mentioned above between $\mathfrak{R}(k)$ and $\mathfrak{R}(k+1)$. Any improvement in the exponent in the bound in Theorem 6 would immediately translate into an improved space separation, and would also improve the k-DNF resolution trade-offs in [10].

Prior to the current paper, the best lower bound giving limits on what one could hope to achieve in Theorem 6 was linear, i.e., $|\mathit{Vars}(G)| = \Omega(|\mathbb{D}|)$. Namely, let G be a conjunction of XORs $(\bigoplus_{i=1}^{k+1} x_i) \wedge (\bigoplus_{i=1}^{k+1} y_i) \wedge (\bigoplus_{i=1}^{k+1} z_i) \wedge \cdots$ and $\mathbb{D}$ be the union of the expansions of every $\bigoplus_{i=1}^{k+1} x_i$ as a CNF formula. For this particular structure of G it is also easy to prove that $|\mathit{Vars}(G)| = \mathrm{O}(|\mathbb{D}|)$ for *any* choice of $\mathbb{D}$, but it was open what happens when we consider general formulas G.

For $k = 1$, [10] proved that a linear bound $\mathrm{O}(|\mathbb{D}|)$ in fact holds for any set of clauses $\mathbb{D}$ and any $\left(\wedge\vee\oplus^2\right)$-block formula G, but all attempts to extend the

techniques used there to the case $k > 1$ have failed. And indeed, they have failed for a good reason, since building on the construction in Section 3 we can show that the best one can hope for in Theorem 6 is $|Vars(G)| = \mathrm{O}\big(|\mathbb{D}|^k\big)$.

Theorem 7. *For any $k > 1$ there are arbirarily large k-DNF sets $\mathbb{D}$ of size m and $\big(\wedge\vee\oplus^{k+1}\big)$-block formulas G such that $\mathbb{D}$ "precisely" implies G in the sense of Theorem 6 and $|Vars(G)| \geq (k+1)\big[\frac{m}{k+2}\big(1-\frac{1}{k}\big)\big]^k \geq k\big(\frac{m}{4k}\big)^k$.*

Proof. We utilize all the previous notation and start with the CNF formula

$$\bigwedge_{\nu\in[m(k-1)]} \bigvee_{(i_1,\ldots,i_{k-1})\in[m(k-1)]^{k-1}} y^{\nu}_{i_1,\ldots,i_{k-1}} \tag{6}$$

and substitute an exclusive or over variables $y^{\nu,r}_{i_1,\ldots,i_{k-1}}$, $r = 1,\ldots,k+1$, for every variable $y^{\nu}_{i_1,\ldots,i_{k-1}}$. This results in the formula

$$G = \bigwedge_{\nu\in[m(k-1)]} \bigvee_{(i_1,\ldots,i_{k-1})\in[m(k-1)]^{k-1}} \bigoplus_{r=1}^{k+1} y^{\nu,r}_{i_1,\ldots,i_{k-1}} \tag{7}$$

which will be our $\big(\wedge\vee\oplus^{k+1}\big)$-block formula. Clearly, G contains $(k+1)\cdot(m(k-1))^k$ variables. We claim that the following easy modification of the k-DNF set from Figure 2 "precisely" implies G in the sense of Theorem 6:

$$W^j_m(\boldsymbol{x}^j) \qquad 1 \leq j < k \tag{8a}$$

$$\bigvee_{(i_1,\ldots,i_{k-1})\in[m(k-1)]^{k-1}} \left(x^1_{i_1} \wedge \cdots \wedge x^{k-1}_{i_{k-1}} \wedge y^{\nu,1}_{i_1,\ldots,i_{k-1}}\right) \qquad 1 \leq \nu \leq m(k-1) \tag{8b}$$

$$\bigvee_{(i_1,\ldots,i_{k-1})\in[m(k-1)]^{k-1}} \left(x^1_{i_1} \wedge \cdots \wedge x^{k-1}_{i_{k-1}} \wedge \overline{y}^{\nu,r}_{i_1,\ldots,i_{k-1}}\right) \qquad \begin{array}{c} 1 \leq \nu \leq m(k-1), \\ 2 \leq r \leq k+1 \end{array} \tag{8c}$$

It is straightforward to verify that $\mathbb{D}$ consists of less than $m(k-1)(k+1) + 2mk \leq mk(k+2)$ k-DNF formulas. $\mathbb{D}$ implies G since once we have picked which variables $x^1_{i^*_1}, x^2_{i^*_2}, \ldots, x^{k-1}_{i^*_{k-1}}$ should be satisfied, $\mathbb{D}$ will force all XOR blocks $\bigoplus_{r=1}^{k+1} y^{\nu,r}_{i^*_1,\ldots,i^*_{k-1}}$, $j \in [m(k-1)]$ to true by requiring the variable $y^{\nu,1}_{i^*_1,\ldots,i^*_{k-1}}$ to be true and all other variables $y^{\nu,r}_{i^*_1,\ldots,i^*_{k-1}}$, $r \geq 2$, to be false. Finally, it is also easy to verify that if a single XOR block $\bigoplus_{r=1}^{k+1} y^{\nu,r}_{i^*_1,\ldots,i^*_{k-1}}$ is removed from G, then we can satisfy $\mathbb{D}$ but falsify the rest of the formula G (the proof is very similar to the one given in Section 3.2). Theorem 7 follows.

5 Concluding Remarks and Open Problems

We conclude this paper by discussing two remaining open problems. First, the most obvious problem still open is to close the gap between the lower bound

$(\Omega(m))^k$ and upper bound $(mk)^{k+1}$ on the number of variables that can appear in a minimally unsatisfiable k-DNF set with m formulas. A strong intuition expressed by [10] is that it should be possible to bring down the exponent from $k+1$ to k. Hence, we have the following conjecture, where for simplicity we fix k to remove it from the asymptotic notation.

Conjecture 8. Suppose that $\mathbb{D}$ is a minimally unsatisfiable k-DNF set for some arbitrary but fixed $k > 1$. Then the number of variables in $\mathbb{D}$ is at most $\mathrm{O}(|\mathbb{D}|)^k$.

Proving this conjecture would establish asymptotically tight bounds for minimally unsatisfiable k-DNF sets (ignoring factors involving the constant k).

Second, we again stress that the result in Theorem 7 does not per se imply any restrictions (that we are aware of) on what space separations or time-space trade-offs are possible for k-DNF resolution. The reason for this is that our improved lower bound only rules out *a particular approach* for proving better separations and trade-offs, but it does not say anything to the effect that the k-DNF resolution proof systems are strong enough to match this lower bound. It would be very interesting to understand better the strength of k-DNF resolution in this respect. Hence we have the following open problem (where due to space constraints we have to refer to [10] or [17] for the relevant formal definitions).

Problem 9. Let $Peb_G^{k+1}[\oplus]$ be the XOR-pebbling contradiction over some directed acyclic graph G. Is it possible that $\mathfrak{R}(k)$ can refute $Peb_G^{k+1}[\oplus]$ in space asymptotically better than the black-white pebbling price $\mathit{BW\text{-}Peb}(G)$ of G?

We remark that for standard resolution, i.e., 1-DNF resolution, the answer to this question is that XOR-pebbling contradictions over two or more variables *cannot* be refuted in space less than the black-white pebbling price, as proven in [10]. For k-DNF resolution with $k > 1$, however, the best known lower bound is $\Omega\big(\sqrt[k+1]{\mathit{BW\text{-}Peb}(G)}\big)$, as also shown in [10]. There is a wide gap here between the upper and lower bounds since, as far as we are aware, there are no known k-DNF resolution proofs that can do better than space linear in the pebbling price (which is achievable by standard resolution).

Acknowledgements. The authors would like to thank Eli Ben-Sasson for getting them to work together on this problem and for many stimulating discussions. We also gratefully acknowledge the *ICALP* anonymous reviewers, whose comments helped improve this manuscript considerably. The first author is grateful to Johan Håstad, Nati Linial, and Klas Markström for thoughtful comments and advice about the problem of minimally unsatisfiable k-DNF sets.

The first author did this work while at the Massachusetts Institute of Technology, supported by the Royal Swedish Academy of Sciences, the Ericsson Research Foundation, the Sweden-America Foundation, the Foundation Olle Engkvist Byggmästare, and the Foundation Blanceflor Boncompagni-Ludovisi, née Bildt. He is currently supported by Swedish Research Council grant 621-2010-4797.

The second author did part of this work while with Steklov Mathematical Institute, supported by the Russian Foundation for Basic Research, and with Toyota Technological Institute at Chicago.

References

[1] Aharoni, R., Linial, N.: Minimal non-two-colorable hypergraphs and minimal unsatisfiable formulas. J. Comb. Theory 43, 196–204 (1986)

[2] Alekhnovich, M.: Lower bounds for k-DNF resolution on random 3-CNFs. In: Proc. 37th ACM Symposium on Theory of Computing, pp. 251–256 (May 2005)

[3] Alekhnovich, M., Ben-Sasson, E., Razborov, A.A., Wigderson, A.: Space complexity in propositional calculus. SICOMP 31(4), 1184–1211 (2002)

[4] Atserias, A., Bonet, M.L.: On the automatizability of resolution and related propositional proof systems. Info. Comp. 189(2), 182–201 (2004)

[5] Atserias, A., Bonet, M.L., Esteban, J.L.: Lower bounds for the weak pigeonhole principle and random formulas beyond resolution. Info. Comp. 176(2), 136–152 (2002)

[6] Baumer, S., Esteban, J.L., Torán, J.: Minimally unsatisfiable CNF formulas. Bulletin of the EATCS 74, 190–192 (2001)

[7] Ben-Sasson, E., Galesi, N.: Space complexity of random formulae in resolution. Rand. Struc. Alg. 23(1), 92–109 (2003)

[8] Ben-Sasson, E., Nordström, J.: Short proofs be spacious: An optimal separation of space and length in resolution. In: Proc. 49th IEEE Symposium on Foundations of Computer Science, pp. 709–718 (2008)

[9] Ben-Sasson, E., Nordström, J.: A space hierarchy for k-DNF resolution. Technical Report TR09-047, Electronic Colloquium on Computational Complexity (2009)

[10] Ben-Sasson, E., Nordström, J.: Understanding space in proof complexity: Separations and trade-offs via substitutions. In: Proc. 2nd Symposium on Innovations in Computer Science, pp. 401–416 (2011)

[11] Ben-Sasson, E., Wigderson, A.: Short proofs are narrow—resolution made simple. J. ACM 48(2), 149–169 (2001)

[12] Chvátal, V., Szemerédi, E.: Many hard examples for resolution. J. ACM 35(4), 759–768 (1988)

[13] Esteban, J.L., Galesi, N., Messner, J.: On the complexity of resolution with bounded conjunctions. TCS 321(2-3), 347–370 (2004)

[14] Krajíček, J.: On the weak pigeonhole principle. Fund. Math. 170(1-3), 123–140 (2001)

[15] Kullmann, O.: An application of matroid theory to the SAT problem. In: Proc. 15th IEEE Conference on Computational Complexity, pp. 116–124 (2000)

[16] Nordström, J.: Narrow proofs be spacious: Separating space and width in resolution. SICOMP 39(1), 59–121 (2009)

[17] Nordström, J.: Pebble games, proof complexity and time-space trade-offs. Logical Methods in Computer Science (to appear, 2011)

[18] Nordström, J., Håstad, J.: Towards an optimal separation of space and length in resolution (Extended abstract). In: Proc. 40th ACM Symposium on Theory of Computing, pp. 701–710 (May 2008)

[19] Razborov, A.A.: Pseudorandom generators hard for k-DNF resolution and polynomial calculus resolution (manuscript)

[20] Segerlind, N.: Exponential separation between Res(k) and Res($k+1$) for $k \leq \epsilon \log n$. IPL 93(4), 185–190 (2005)

[21] Segerlind, N., Buss, S.R., Impagliazzo, R.: A switching lemma for small restrictions and lower bounds for k-DNF resolution. SICOMP 33(5), 1171–1200 (2004)

Sorting by Transpositions Is Difficult

Laurent Bulteau, Guillaume Fertin, and Irena Rusu

Laboratoire d'Informatique de Nantes-Atlantique (LINA), UMR CNRS 6241
Université de Nantes, 2 rue de la Houssinière, 44322 Nantes Cedex 3 - France
{Laurent.Bulteau,Guillaume.Fertin,Irena.Rusu}@univ-nantes.fr

Abstract. In comparative genomics, a transposition is an operation that exchanges two consecutive sequences of genes in a genome. The transposition distance, that is, the minimum number of transpositions needed to transform a genome into another, can be considered as a relevant evolutionary distance. The problem of computing this distance when genomes are represented by permutations, called the SORTING BY TRANSPOSITIONS problem (SBT), has been introduced by Bafna and Pevzner [3] in 1995. It has naturally been the focus of a number of studies, but the computational complexity of this problem has remained undetermined for 15 years.

In this paper, we answer this long-standing open question by proving that the SORTING BY TRANSPOSITIONS problem is NP-hard. As a corollary of our result, we also prove that the following problem from [10] is NP-hard: given a permutation π, is it possible to sort π using $d_b(\pi)/3$ permutations, where $d_b(\pi)$ is the number of breakpoints of π?

Introduction

Along with reversals, transpositions are one of the most elementary large-scale operations that can affect a genome. A transposition consists in swapping two consecutive sequences of genes or, equivalently, in moving a sequence of genes from one place to another in the genome. The transposition distance between two genomes is the minimum number of such operations that are needed to transform one genome into the other. Computing this distance is a challenge in comparative genomics, since it gives a maximum parsimony evolution scenario between the two genomes.

The SORTING BY TRANSPOSITIONS problem is the problem of computing the transposition distance between genomes represented by permutations: see [17] for a detailed review on this problem and its variants. Since its introduction by Bafna and Pevzner [3,4], the complexity class of this problem has never been established. Hence a number of studies [4,10,18,20,14,5,16] aim at designing approximation algorithms or heuristics, the best known fixed-ratio algorithm being a 1.375-approximation [14]. Other works [19,10,15,22,14,5] aim at computing bounds on the transposition distance of a permutation. Studies have also been devoted to variants of this problem, by considering, for example, prefix transpositions [13,23,8] (in which one of the blocks has to be a prefix of the sequence), or distance between strings [11,12,26,25,21] (where multiple occurrences of each element are allowed in the sequences), possibly with weighted or prefix transpositions [24,6,1,2,8]. Note also that sorting a permutation by block-interchanges (i.e. exchanges of non-necessarily consecutive sequences) is a polynomial problem [9].

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 654–665, 2011.

In this paper, we address the long-standing issue of determining the complexity class of the SORTING BY TRANSPOSITIONS problem, by giving a polynomial-time reduction from SAT, thus proving the NP-hardness of this problem. Our reduction is based on the study of transpositions that remove three breakpoints. A corollary of our result is the NP-hardness of the following problem, introduced in [10]: given a permutation π, is it possible to sort π using $d_b(\pi)/3$ permutations, where $d_b(\pi)$ is the number of breakpoints of π? Due to space constraints, this paper only describes the reduction itself. The detailed proof can be found in the full version of this paper [7].

1 Preliminaries

In this paper, n denotes a positive integer. Let $[\![a\,;\,b]\!] = \{x \in \mathbb{N} \mid a \leq x \leq b\}$, and Id_n be the identity permutation over $[\![0\,;\,n]\!]$. We consider only permutations of $[\![0\,;\,n]\!]$ such that 0 and n are fixed-points. Given a word $u_1\ u_2\ \ldots\ u_l$, a *subword* is a subsequence $u_{p_1}\ u_{p_2}\ \ldots\ u_{p_{l'}}$, where $1 \leq p_1 < p_2 < \ldots < p_{l'} \leq l$. A *factor* is a subsequence of contiguous elements, i.e. a subword with $p_{k+1} = p_k + 1$ for every $k \in [\![1\,;\,l'-1]\!]$.

Definition 1 (Transposition). *Given three integers $0 < i < j < k \leq n$, the* transposition $\tau_{i,j,k}$ *over $[\![0\,;\,n]\!]$ is the following permutation:*

$$\tau_{i,j,k} = \begin{pmatrix} 0 & \cdots & i-1 & i & \cdots & k+i-j-1 & k+i-j & \cdots & k-1 & k & \cdots & n \\ 0 & \cdots & i-1 & j & \cdots & k-1 & i & \cdots & j-1 & k & \cdots & n \end{pmatrix}$$

Let π be a permutation of $[\![0\,;\,n]\!]$. The transposition distance $d_t(\pi)$ *from π to Id_n is the minimum value k for which there exist k transpositions $\tau_1, \tau_2, \ldots, \tau_k$ such that $\pi \circ \tau_k \circ \ldots \circ \tau_2 \circ \tau_1 = Id_n$.*

The transposition $\tau_{i,j,k}$ is the operation that, when it is composed with a permutation, exchanges factors with indices $i, \ldots, j-1$ and $j, \ldots, k-1$, see Figure 1a. The inverse function of $\tau_{i,j,k}$ is also a transposition: $\tau_{i,j,k}^{-1} = \tau_{i,k+i-j,k}$. See Figure 1b for an example of the computation of the transposition distance.

We consider the following problem:

> SORTING BY TRANSPOSITIONS PROBLEM [3]
> INPUT: A permutation π, an integer k.
> QUESTION: Is $d_t(\pi) \leq k$?

Computing the transposition distance has often been linked to studying the breakpoints of a permutation. A breakpoint of π is a pair $(x-1, x)$, $x \in [\![1\,;\,n]\!]$, such that $\pi(x) \neq \pi(x-1)+1$. A transposition can decrease the number of breakpoints of a permutation, $d_b(\pi)$, by at most 3. In this paper we in fact focus on the "simpler" problem of determining whether $d_t(\pi) = d_b(\pi)/3$ for a given permutation π.

2 3-Deletion and Transposition Operations

In this section, we introduce 3DT-instances, which are the cornerstone of our reduction from SAT to the SORTING BY TRANSPOSITIONS problem, since they are used as an intermediate between instances of the two problems.

$$\begin{aligned}\pi &= (\pi_0\pi_1\dots\pi_{i-1}\underline{\pi_i\dots\pi_{j-1}}\,\underline{\pi_j\quad\dots\quad\pi_{k-1}}\pi_k\dots\pi_n)\\ \pi\circ\tau_{i,j,k} &= (\pi_0\pi_1\dots\pi_{i-1}\underline{\pi_j\quad\dots\quad\pi_{k-1}}\,\underline{\pi_i\dots\pi_{j-1}}\pi_k\dots\pi_n)\end{aligned}$$

(a)

$$\begin{aligned}\pi &= (0\,\underline{2\,4}\,\underline{3\,1}\,5)\\ \pi\circ\tau_{1,3,5} &= (0\,\underline{3\,1}\,\underline{2\,4}\,5)\\ \pi\circ\tau_{1,3,5}\circ\tau_{1,2,4} &= (0\,1\,2\,3\,4\,5)\end{aligned}$$

(b)

Fig. 1. (a) Representation of a transposition $\tau_{i,j,k}$ for $0 < i < j < k \leq n$ on a general permutation. (b) The transposition distance from $\pi = (0\ 2\ 4\ 3\ 1\ 5)$ to Id_5 is 2: it is at most 2 since $\pi \circ \tau_{1,3,5} \circ \tau_{1,2,4} = Id_5$, and it cannot be less than 2 since $d_t(\pi) \geq d_b(\pi)/3 = 5/3 > 1$.

Definition 2 (3DT-instance). *A* 3DT-instance $I = \langle \Sigma, T, \psi \rangle$ *of span* n *is composed of the following elements:*

- Σ*: an alphabet of at most* n *elements;*
- $T = \{(a_i, b_i, c_i) \mid 1 \leq i \leq |T|\}$*: a set of (ordered) triples of elements of* Σ*, partitioning* Σ *(i.e. all elements are pairwise distinct, and* $\bigcup_{i=1}^{|T|}\{a_i, b_i, c_i\} = \Sigma$*);*
- $\psi : \Sigma \to [\![1\,;\,n]\!]$*, an injection.*

The domain *of* I *is the image of* ψ*, that is the set* $L = \{\psi(\sigma) \mid \sigma \in \Sigma\}$*. The* word representation *of* I *is the* n*-letter word* $u_1\,u_2\dots u_n$ *over* $\Sigma \cup \{\centerdot\}$ *(where* $\centerdot \notin \Sigma$*), such that for all* $i \in L$*,* $\psi(u_i) = i$*, and for* $i \in [\![1\,;\,n]\!] - L$*,* $u_i = \centerdot$*. For* $\sigma_1, \sigma_2 \in \Sigma$*, we write* $\sigma_1 \prec \sigma_2$ *if* $\psi(\sigma_1) < \psi(\sigma_2)$*, and* $\sigma_1 \lhd \sigma_2$ *if* $\sigma_1 \prec \sigma_2$ *and* $\nexists x \in \Sigma,\ \sigma_1 \prec x \prec \sigma_2$*.*

Two examples of 3DT-instances are given in Figure 2. Note that such instances can be defined by their word representation and by their set of triples T. The empty 3DT-instance, in which $\Sigma = \emptyset$, can be written with a sequence of n dots, or with the empty word ε.

Using the triples in T, we can define a successor function over the domain L:

Definition 3. *Let* $I = \langle \Sigma, T, \psi \rangle$ *be a 3DT-instance with domain* L*. We write* $succ_I : L \to L$ *the function such that, for all* $(a, b, c) \in T$*,* $\psi(a) \mapsto \psi(b)$*,* $\psi(b) \mapsto \psi(c)$*, and* $\psi(c) \mapsto \psi(a)$*.*

Function $succ_I$ is a bijection, with no fixed-points, and such that $succ_I \circ succ_I \circ succ_I$ is the identity over L.

In the example of Figure 2, $succ_I = \begin{pmatrix}1&2&3&4&5&6\\3&6&5&2&1&4\end{pmatrix}$ and $succ_{I'} = \begin{pmatrix}2&4&6\\4&6&2\end{pmatrix}$.

Definition 4. *Let* $I = \langle \Sigma, T, \psi \rangle$ *be a 3DT-instance, and* (a, b, c) *be a triple of* T*. Write* $i = \min\{\psi(a), \psi(b), \psi(c)\}$*,* $j = succ_I(i)$*, and* $k = succ_I(j)$*. The triple* $(a, b, c) \in T$ *is* well-ordered *if we have* $i < j < k$*. In such a case, we write* $\tau[a, b, c, \psi]$ *the transposition* $\tau_{i,j,k}$*.*

An equivalent definition is that $(a, b, c) \in T$ is well-ordered iff one of abc, bca, cab is a subword of the word representation of I. In the example of Figure 2, (a_1, b_1, c_1) is well-ordered in I: indeed, we have $i = \psi(a_1)$, $j = \psi(b_1)$ and $k = \psi(c_1)$, so $i < j < k$. The triple (a_2, b_2, c_2) is also well-ordered in I' ($i = \psi'(b_2) < j = \psi'(c_2) < k = \psi'(a_2)$), but not in I: $i = \psi(c_2) < k = \psi(b_2) < j = \psi(a_2)$. In this example, we have $\tau[a_1, b_1, c_1, \psi] = \tau_{1,3,5}$ and $\tau[a_2, b_2, c_2, \psi'] = \tau_{2,4,6}$.

$$\begin{array}{lll} I &= a_1\ c_2\ b_1\ b_2\ c_1\ a_2 & \text{with} \quad T = \{(a_1,b_1,c_1),(a_2,b_2,c_2)\} \\ I' &= \ \cdot\ b_2\ \cdot\ c_2\ \cdot\ a_2 & \text{with} \quad T' = \{(a_2,b_2,c_2)\} \end{array}$$

Fig. 2. Two examples of 3DT-instances of span 6. We write $I = \langle \Sigma, T, \psi \rangle$ and $I' = \langle \Sigma', T', \psi' \rangle$. I has an alphabet of size 6, $\Sigma = \{a_1, b_1, c_1, a_2, b_2, c_2\}$, hence ψ is a bijection ($\psi(a_1) = 1$, $\psi(c_2) = 2$, $\psi(b_1) = 3$, etc). I' has an alphabet of size 3, $\Sigma' = \{a_2, b_2, c_2\}$, with $\psi'(b_2) = 2$, $\psi'(c_2) = 4$, $\psi'(a_2) = 6$.

Definition 5 (3DT-step). *Let $I = \langle \Sigma, T, \psi \rangle$ be a 3DT-instance with $(a,b,c) \in T$ a well-ordered triple. The* 3DT-step *of parameter (a,b,c) is the operation written $\xrightarrow{(a,b,c)}$, transforming I into the 3DT-instance $I' = \langle \Sigma', T', \psi' \rangle$ such that, with $\tau = \tau[a,b,c,\psi]$:*

$$\Sigma' = \Sigma - \{a,b,c\}; \qquad T' = T - \{(a,b,c)\}; \qquad \psi' : \begin{array}{ccc} \Sigma' & \to & [\![1\,;\,n]\!] \\ \sigma & \mapsto & \tau^{-1}(\psi(\sigma)) \end{array}.$$

If the word representation of I is $W\,a\,X\,b\,Y\,c\,Z$, then, after the 3DT-step $I \xrightarrow{(a,b,c)} I'$, the word representation of I' is $W \cdot Y \cdot X \cdot Z$. Note that a triple that is not well-ordered in I can become well-ordered in I', or vice-versa. In the example of Figure 2, I' can be obtained from I via a 3DT-step: $I \xrightarrow{(a_1,b_1,c_1)} I'$. Moreover, $I' \xrightarrow{(a_2,b_2,c_2)} \varepsilon$.

Definition 6 (3DT-collapsibility). *A 3DT-instance $I = \langle \Sigma, T, \psi \rangle$ is* 3DT-collapsible *if there exists a sequence of 3DT-instances $I_k, I_{k-1}, \ldots, I_0$ such that $I_k = I$, $I_0 = \varepsilon$, and $\forall i \in [\![1\,;\,k]\!]$, $\exists (a,b,c) \in T$, $I_i \xrightarrow{(a,b,c)} I_{i-1}$.*

In Figure 2, I and I' are 3DT-collapsible, since we have $I \xrightarrow{(a_1,b_1,c_1)} I' \xrightarrow{(a_2,b_2,c_2)} \varepsilon$.

3 3DT-Collapsibility Is NP-Hard to Decide

In this section, we define, for any boolean formula ϕ, a corresponding 3DT-instance I_ϕ. We also prove that I_ϕ is 3DT-collapsible iff ϕ is satisfiable (see Theorem 1).

3.1 Block Structure

The construction of the 3DT-instance I_ϕ uses a decomposition into blocks, defined below. Some triples will be included in a block, in order to define its behavior, while others will be shared between two blocks, in order to pass information. The former are unconstrained, so that we can design blocks with the behavior we need (for example, blocks mimicking usual boolean functions), while the latter need to follow several rules, so that the blocks can easily be arranged together.

Definition 7 (l-block-decomposition). *An l-*block-decomposition *$\mathcal{B}$ of a 3DT-instance I of span n is an l-tuple $(s_1, \ldots, s_l)$ such that $s_1 = 0$, for all $h \in [\![1\,;\,l-1]\!]$, $s_h < s_{h+1}$ and $s_l < n$. We write $t_h = s_{h+1}$ for $h \in [\![1\,;\,l-1]\!]$, and $t_l = n$.*

Let $I = \langle \Sigma, T, \psi \rangle$ and $u_1\ u_2\ \ldots u_n$ be the word representation of I. For $h \in [\![1\,;\,l]\!]$, the subword $u_{s_h+1}\ u_{s_h+2}\ \ldots u_{t_h}$ where every occurrence of $\cdot$ is deleted is called the

block $\mathcal{B}_h$. *For $\sigma \in \Sigma$, we write $block_{I,\mathcal{B}}(\sigma) = h$ if $\psi(\sigma) \in [\![s_h + 1\,;\, t_h]\!]$ (equivalently, if σ appears in the word $\mathcal{B}_h$). A triple $(a, b, c) \in T$ is said to be* internal *if $block_{I,\mathcal{B}}(a) = block_{I,\mathcal{B}}(b) = block_{I,\mathcal{B}}(c)$,* external *otherwise.*

Given a 3DT-step $I \xrightarrow{(a,\,b,\,c)} I'$, the arrow notation can be extended to an l-block-decomposition $\mathcal{B}$ of I, provided at least one of the following equalities is satisfied: $block_{I,\mathcal{B}}(a) = block_{I,\mathcal{B}}(b)$, $block_{I,\mathcal{B}}(b) = block_{I,\mathcal{B}}(c)$ or $block_{I,\mathcal{B}}(c) = block_{I,\mathcal{B}}(a)$. In this case, with $\tau = \tau[a, b, c, \psi]$, the l-tuple $\mathcal{B}' = (\tau^{-1}(s_1), \ldots, \tau^{-1}(s_l))$ is an l-block-decomposition of I', and we write $(I, \mathcal{B}) \xrightarrow{(a,\,b,\,c)} (I', \mathcal{B}')$.

Definition 8 (Variable). *A* variable *A of a 3DT-instance $I = \langle \Sigma, T, \psi \rangle$ is a pair of triples $A = [(a, b, c), (x, y, z)]$ of T. It is* valid *in an l-block-decomposition $\mathcal{B}$ if*

(i) $\exists h_0 \in [\![1\,;\, l]\!]$ *such that* $block_{I,\mathcal{B}}(b) = block_{I,\mathcal{B}}(x) = block_{I,\mathcal{B}}(y) = h_0$
(ii) $\exists h_1 \in [\![1\,;\, l]\!]$, $h_1 \neq h_0$, *such that* $block_{I,\mathcal{B}}(a) = block_{I,\mathcal{B}}(c) = block_{I,\mathcal{B}}(z) = h_1$
(iii) *if* $x \prec y$, *then we have* $x \lhd b \lhd y$
(iv) $a \prec z \prec c$

For such a valid variable A, the block $\mathcal{B}_{h_0}$ containing $\{b, x, y\}$ is called the source *of A, and the block $\mathcal{B}_{h_1}$ containing $\{a, c, z\}$ is called the* target *of A. For $h \in [\![1\,;\, l]\!]$, the variables of which $\mathcal{B}_h$ is the source (resp. the target) are called the* output *(resp. the* input*) of $\mathcal{B}_h$. The 3DT-step $I \xrightarrow{(x,\,y,\,z)} I'$ is called the* activation *of A (it requires that (x, y, z) is well-ordered).*

Note that, for any valid variable $A = [(a, b, c), (x, y, z)]$ in $(I, \mathcal{B})$, we have, according to condition (i), $block_{I,\mathcal{B}}(x) = block_{I,\mathcal{B}}(y)$, thus its activation can be written $(I, \mathcal{B}) \xrightarrow{(x,\,y,\,z)} (I', \mathcal{B}')$.

Property 1. Let $(I, \mathcal{B})$ be a 3DT-instance with an l-block-decomposition, and A be a variable of I that is valid in $\mathcal{B}$, $A = [(a, b, c), (x, y, z)]$. Then (x, y, z) is well-ordered iff $x \prec y$; and (a, b, c) is not well-ordered.

Definition 9 (Valid context). *A 3DT-instance with an l-block-decomposition $(I, \mathcal{B})$ is a* valid context *if the set of external triples of I can be partitioned into valid variables.*

Let B be a block in a valid context $(I, \mathcal{B})$ (in which $B = \mathcal{B}_h$, for some $h \in [\![1\,;\, l]\!]$), and $(I, \mathcal{B}) \xrightarrow{(d,\,e,\,f)} (I', \mathcal{B}')$ be a 3DT-step such that, writing $B' = \mathcal{B}'_h$, we have $B' \neq B$. Then, depending on the triple (d, e, f), we are in one of the following three cases:

– (d, e, f) is an internal triple of B. We write: $B \text{ —} (d, e, f) \text{→ } B'$
– $(d, e, f) = (x, y, z)$ for some output $A = [(a, b, c), (x, y, z)]$ of B. We write: $B \text{ ═} A \text{⟹ } B'$
– $(d, e, f) = (x, y, z)$ for some input $A = [(a, b, c), (x, y, z)]$ of B. We write: $B \text{ —} A \text{→ } B'$

The graph obtained from a block B by following exhaustively the possible arcs as defined above (always assuming this block is in a valid context) is called the *behavior graph* of B. Figure 3 illustrates the activation of a valid variable A.

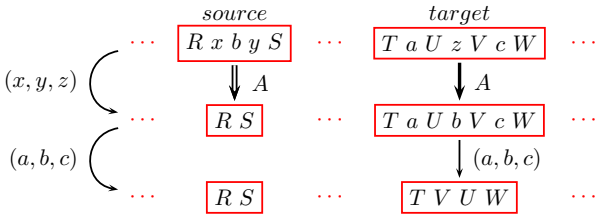


Fig. 3. Activation of a valid variable $A = [(a, b, c), (x, y, z)]$. It can be followed by the 3DT-step $\xrightarrow{(a,b,c)}$, impacting only the target block of A. Dot symbols ($\cdot$) are omitted. We denote by R, S, T, U, V, W some factors of the source and target blocks of A: the consequence of activating A is to allow U and V to be swapped in the target of A.

3.2 Basic Blocks

We now define four basic blocks: copy, and, or, and var. They are studied independently in this section, before being assembled in Section 3.3. Each of these blocks is defined by a word and a set of triples. We distinguish internal triples, for which all three elements appear in a single block, from external triples, which are part of an input/output variable, and for which only one or two elements appear in the block. Note that each external triple is part of an input (resp. output) variable, which itself must be an output (resp. input) of another block, the other block containing the remaining elements of the triple.

We then compute the behavior graph of each of these blocks (it is given here for the block copy, see Figure 4, and in the full version for the other blocks): in each case, we assume that the block is in a valid context, and follow exhaustively the 3DT-steps that can be applied to it. It must be kept in mind that for any variable, it is the state of the source block which determines whether it can be activated, whereas the activation itself affects mostly the target block. It can be verified that each output (resp. input) variable of these blocks satisfy the constraints (i) and (iii) (resp. (ii) and (iv)) of Definition 8.

The block copy. This block aims at duplicating a variable: any of the two output variables can only be activated after the input variable has been activated. See Figure 4 for the behavior graph of this block.
Input variable: $A = [(a, b, c), (x, y, z)]$.
Output variables: $A_1 = [(a_1, b_1, c_1), (x_1, y_1, z_1)]$ and $A_2 = [(a_2, b_2, c_2), (x_2, y_2, z_2)]$.
Internal triple: (d, e, f).
Definition:

$$[A_1, A_2] = \mathsf{copy}(A) \quad = \quad a\ y_1\ e\ z\ d\ y_2\ x_1\ b_1\ c\ x_2\ b_2\ f$$

Property 2. In a block $[A_1, A_2] = \mathsf{copy}(A)$ in a valid context, the possible orders in which A, A_1 and A_2 can be activated are (A, A_1, A_2) and (A, A_2, A_1).

The block and. This block aims at simulating a conjunction: the output variable can only be activated after both input variables have been activated.

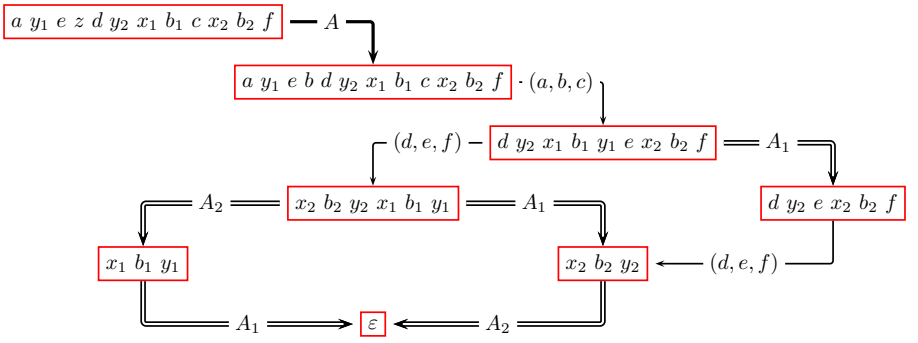


Fig. 4. Behavior graph of the block $[A_1, A_2] = \mathsf{copy}(A)$

Input variables: $A_1 = [(a_1, b_1, c_1), (x_1, y_1, z_1)]$ and $A_2 = [(a_2, b_2, c_2), (x_2, y_2, z_2)]$.
Output variable: $A = [(a, b, c), (x, y, z)]$.
Internal triple: (d, e, f).
Definition:

$$A = \mathsf{and}(A_1, A_2) \quad = \quad a_1\ e\ z_1\ a_2\ c_1\ z_2\ d\ y\ c_2\ x\ b\ f$$

Property 3. In a block $A = \mathsf{and}(A_1, A_2)$ in a valid context, the possible orders in which A, A_1 and A_2 can be activated are (A_1, A_2, A) and (A_2, A_1, A).

The block or. This block aims at simulating a disjunction: the output variable can be activated as soon as any of the two input variables is activated.
Input variables: $A_1 = [(a_1, b_1, c_1), (x_1, y_1, z_1)]$ and $A_2 = [(a_2, b_2, c_2), (x_2, y_2, z_2)]$.
Output variable: $A = [(a, b, c), (x, y, z)]$.
Internal triples: (a', b', c') and (d, e, f).
Definition:

$$A = \mathsf{or}(A_1, A_2) \quad = \quad a_1\ b'\ z_1\ a_2\ d\ y\ a'\ x\ b\ f\ z_2\ c_1\ e\ c'\ c_2$$

Property 4. In a block $A = \mathsf{or}(A_1, A_2)$ in a valid context, the possible orders in which A, A_1 and A_2 can be activated are (A_1, A, A_2), (A_2, A, A_1), (A_1, A_2, A) and (A_2, A_1, A).

The block var. This block aims at simulating a boolean variable: in a first stage, only one of the two output variables can be activated. The other needs the activation of the input variable to be activated.
Input variable: $A = [(a, b, c), (x, y, z)]$.
Output variables: $A_1 = [(a_1, b_1, c_1), (x_1, y_1, z_1)]$, $A_2 = [(a_2, b_2, c_2), (x_2, y_2, z_2)]$.
Internal triples: (d_1, e_1, f_1), (d_2, e_2, f_2) and (a', b', c').
Definition:

$$[A_1, A_2] = \mathsf{var}(A) \quad = \quad d_1\ y_1\ a\ d_2\ y_2\ e_1\ a'\ e_2\ x_1\ b_1\ f_1\ c'\ z\ b'\ c\ x_2\ b_2\ f_2$$

Property 5. In a block $[A_1, A_2] = \mathsf{var}(A)$ in a valid context, the possible orders in which A, A_1 and A_2 can be activated are (A_1, A, A_2), (A_2, A, A_1), (A, A_1, A_2) and (A, A_2, A_1).

With such a block, if A is not activated first, one needs to make a choice between activating A_1 or A_2. Once A is activated, however, all remaining output variables are activable.

Assembling the blocks copy, and, or, var

Definition 10 (Assembling of basic blocks). *An* assembling of basic blocks $(I, \mathcal{B})$ *is composed of a 3DT-instance I and an l-block-decomposition $\mathcal{B}$ obtained by the following process. Create a set of variables $\mathcal{A}$. Define $I = \langle \Sigma, T, \psi \rangle$ by its word representation, as a concatenation of l factors $\mathcal{B}_1\ \mathcal{B}_2\ \ldots\ \mathcal{B}_l$ and a set of triples T, where each $\mathcal{B}_h$ is one of the blocks $[A_1, A_2] = \mathsf{copy}(A)$, $A = \mathsf{and}(A_1, A_2)$, $A = \mathsf{or}(A_1, A_2)$ or $[A_1, A_2] = \mathsf{var}(A)$, with $A_1, A_2, A \in \mathcal{A}$ (such that each $X \in \mathcal{A}$ appears in the input of exactly one block, and in the output of exactly one other block); and where T is the union of the set of internal triples needed in each block, and the set of external triples defined by the variables of $\mathcal{A}$.*

Lemma 1. *Let I' be a 3DT-instance with an l-block-decomposition $\mathcal{B}'$, such that $(I', \mathcal{B}')$ is obtained from an assembling of basic blocks $(I, \mathcal{B})$ after any number of 3DT-steps. Then $(I', \mathcal{B}')$ is a valid context. Moreover, if the set of variables of $(I', \mathcal{B}')$ is empty, then I' is 3DT-collapsible.*

The above lemma justifies the assumption that each block is in a valid context to derive Properties 2 to 5. An assembling of basic blocks is 3DT-collapsible iff there exists a total order, satisfying these properties, in which all its variables can be activated.

3.3 Construction of I_ϕ

Let ϕ be a boolean formula, over the boolean variables $x_1, \ldots, x_m$, given in conjunctive normal form: $\phi = C_1 \wedge C_2 \wedge \ldots \wedge C_\gamma$. Each clause C_c ($c \in [\![1\,;\,\gamma]\!]$) is the disjunction of a number of literals, x_i or $\neg x_i$, $i \in [\![1\,;\,m]\!]$. We write q_i (resp. $\bar{q}_i$) for the number of occurrences of the literal x_i (resp. $\neg x_i$) in ϕ, $i \in [\![1\,;\,m]\!]$. We also write $k(C_c)$ for the number of literals appearing in the clause C_c, $c \in [\![1\,;\,\gamma]\!]$. We can assume that $\gamma \geq 2$, that for each $c \in [\![1\,;\,\gamma]\!]$, we have $k(C_c) \geq 2$, and that for each $i \in [\![1\,;\,m]\!]$, $q_i \geq 2$ and $\bar{q}_i \geq 2$ (otherwise, we can always add clauses of the form $(x_i \vee \neg x_i)$ to ϕ, or duplicate the literals appearing in the clauses C_c such that $k(C_c) = 1$). In order to distinguish variables of an l-block-decomposition from $x_1, \ldots, x_m$, we always use the term *boolean variable* for the latter.

The 3DT-instance I_ϕ is defined as an assembling of basic blocks: we first define a set of variables, then we list the blocks of which the word representation of I_ϕ is the concatenation. It is necessary that each variable is part of the input (resp. the output) of exactly one block. Note that the relative order of the blocks is of no importance. We simply try, for readability reasons, to ensure that the source of a variable appears before its target, whenever possible. We say that a variable *represents* a term, i.e. a literal, clause or formula, if it can be activated only if this term is true (for some fixed

assignment of the boolean variables), or if ϕ is satisfied by this assignment. We also say that a block *defines* a variable if it is its source block.

The construction of I_ϕ is done as follows (an example is given in the full version [7]): Create a set of variables:

- For each $i \in [\![1\,;\,m]\!]$, create $q_i + 1$ variables representing x_i: X_i and X_i^j, $j \in [\![1\,;\,q_i]\!]$, and $\bar{q}_i + 1$ variables representing $\neg x_i$: $\bar{X}_i$ and $\bar{X}_i^j$, $j \in [\![1\,;\,\bar{q}_i]\!]$.
- For each $c \in [\![1\,;\,\gamma]\!]$, create a variable Γ_c representing the clause C_c.
- Create $m + 1$ variables, A_ϕ and A_ϕ^i, $i \in [\![1\,;\,m]\!]$, representing the formula ϕ.
- We also use a number of intermediate variables, with names $U_i^j, \bar{U}_i^j, V_c^p, W_c$ and Y_i.

Start with an empty 3DT-instance ε, and add blocks successively:

- For each $i \in [\![1\,;\,m]\!]$, add the following $q_i + \bar{q}_i - 1$ blocks defining the variables X_i, X_i^j ($j \in [\![1\,;\,q_i]\!]$), and $\bar{X}_i$, $\bar{X}_i^j$ ($j \in [\![1\,;\,\bar{q}_i]\!]$):

$$[X_i, \bar{X}_i] = \mathsf{var}(A_\phi^i);$$
$$[X_i^1, U_i^2]{=}\mathsf{copy}(X_i);\ [X_i^2, U_i^3]{=}\mathsf{copy}(U_i^2);$$
$$\dots\ [X_i^{q_i-2}, U_i^{q_i-1}]{=}\mathsf{copy}(U_i^{q_i-2});\ [X_i^{q_i-1}, X_i^{q_i}]{=}\mathsf{copy}(U_i^{q_i-1});$$
$$[\bar{X}_i^1, \bar{U}_i^2]{=}\mathsf{copy}(\bar{X}_i);\ [\bar{X}_i^2, \bar{U}_i^3]{=}\mathsf{copy}(\bar{U}_i^2);$$
$$\dots\ [\bar{X}_i^{\bar{q}_i-2}, \bar{U}_i^{\bar{q}_i-1}]{=}\mathsf{copy}(\bar{U}_i^{\bar{q}_i-2});\ [\bar{X}_i^{\bar{q}_i-1}, \bar{X}_i^{\bar{q}_i}]{=}\mathsf{copy}(\bar{U}_i^{\bar{q}_i-1}).$$

- For each $c \in [\![1\,;\,\gamma]\!]$, let $C_c = \lambda_1 \vee \lambda_2 \vee \dots \vee \lambda_k$, with $k = k(C_c)$. Let each λ_p, $p \in [\![1\,;\,k]\!]$, be the j-th occurrence of a literal x_i or $\neg x_i$, for some $i \in [\![1\,;\,m]\!]$ and $j \in [\![1\,;\,q_i]\!]$ (resp. $j \in [\![1\,;\,\bar{q}_i]\!]$). We respectively write $L_p = X_i^j$ or $L_p = \bar{X}_i^j$. Add the following $k - 1$ blocks defining Γ_c:

$$V_c^2 = \mathsf{or}(L_1, L_2);\ V_c^3 = \mathsf{or}(V_c^2, L_3);$$
$$\dots\ V_c^{k-1} = \mathsf{or}(V_c^{k-2}, L_{k-1});\ \Gamma_c = \mathsf{or}(V_c^{k-1}, L_k).$$

- Since $\phi = C_1 \wedge C_2 \wedge \dots \wedge C_l$, add the following $l - 1$ blocks:

$$W_2{=}\mathsf{and}(\Gamma_1, \Gamma_2);\ W_3{=}\mathsf{and}(W_2, \Gamma_3);$$
$$\dots\ W_{l-1}{=}\mathsf{and}(W_{l-2}, \Gamma_{l-1});\ A_\phi{=}\mathsf{and}(W_{l-1}, \Gamma_l).$$

- The m copies $A_\phi^1, \dots, A_\phi^m$ of A_ϕ are defined with the following $m - 1$ blocks:

$$[A_\phi^1, Y_2] = \mathsf{copy}(A_\phi);\ [A_\phi^2, Y_3] = \mathsf{copy}(Y_2);$$
$$\dots [A_\phi^{m-2}, Y_{m-1}] = \mathsf{copy}(Y_{m-2});\ [A_\phi^{m-1}, A_\phi^m] = \mathsf{copy}(Y_{m-1}).$$

Theorem 1. *Let ϕ be a boolean formula, and I_ϕ the 3DT-instance defined above. The construction of I_ϕ is polynomial in the size of ϕ, and ϕ is satisfiable iff I_ϕ is 3DT-collapsible.*

4 Sorting by Transpositions Is NP-Hard

In order to transfer our result from 3DT-collapsibility to SORTING BY TRANSPOSITIONS, we need a notion of equivalence between 3DT-instances and permutations, which is introduced here.

Definition 11. *Let $I = \langle \Sigma, T, \psi \rangle$ be a 3DT-instance of span n with domain L, and π be a permutation of $[\![0\,;\,n]\!]$. We say that I and π are equivalent, and we write $I \sim \pi$, if:*

$$\begin{array}{ll} & \pi(0) = 0, \\ \forall v \in [\![1\,;\,n]\!] - L, & \pi(v) = \pi(v-1) + 1, \\ \forall v \in L, & \pi(v) = \pi(succ_I^{-1}(v) - 1) + 1. \end{array}$$

There is no guarantee that any 3DT-instance I has an equivalent permutation π (for example, no permutation is equivalent to $I = a_1\ a_2\ b_1\ b_2\ c_1\ c_2$). However, coming back to our example in Figure 2, we have $I \sim \pi = (0\,5\,2\,1\,4\,3\,6)$, and $I' \sim \pi' = (0\,1\,4\,5\,2\,3\,6)$. More generally, with the following theorem, we show that such a permutation can always be found in the special case of assemblings of basic blocks, which is the case we are interested in.

Theorem 2. *Let I be a 3DT-instance of span n with $\mathcal{B}$ an l-block-decomposition such that $(I, \mathcal{B})$ is an assembling of basic blocks. Then there exists a permutation π_I, computable in polynomial time in n, such that $I \sim \pi_I$.*

With an equivalence $I \sim \pi$, each breakpoint of π can be associated to an element of Σ via ψ, and the triples of breakpoints that may be resolved by a single transposition correspond to the well-ordered triples of T. Moreover, applying such a transposition on π corresponds to operating a 3DT-step on I. These properties, which lead to the following theorem, can be seen on the previous example as summarized below:

$$\begin{array}{ccccc} I & \xrightarrow{(a_1,b_1,c_1)} & I' & \xrightarrow{(a_2,b_2,c_2)} & \varepsilon \\ \pi & \xrightarrow{\circ\,\tau_{1,3,5}} & \pi' & \xrightarrow{\circ\,\tau_{2,4,6}} & Id_6 \\ d_b(\pi) = 6 & & d_b(\pi') = 3 & & d_b(Id_6) = 0 \end{array}$$

Theorem 3. *Let $I = \langle \Sigma, T, \psi \rangle$ be a 3DT-instance of span n with domain L, and π be a permutation of $[\![0\,;\,n]\!]$, such that $I \sim \pi$. Then I is 3DT-collapsible iff $d_t(\pi) = |T| = d_b(\pi)/3$.*

With the previous theorem, we now have all the necessary ingredients to prove the main result of this paper.

Theorem 4. *The* SORTING BY TRANSPOSITIONS *problem is NP-hard.*

Proof. The reduction from SAT is as follows: given any instance ϕ of SAT, create a 3DT-instance I_ϕ, being an assembling of basic blocks, which is 3DT-collapsible iff ϕ is satisfiable (Theorem 1). Then create a permutation π_{I_ϕ} equivalent to I_ϕ (Theorem 2). The above two steps can be achieved in polynomial time.

Finally, set $k = d_b(\pi_{I_\phi})/3 = n/3$: ϕ is satisfiable iff $d_t(\pi_{I_\phi}) = k$ (Theorem 3).

Corollary 1. *The following decision problem from [10] is also NP-complete: given a permutation π of $[\![0\,;\,n]\!]$, is the equality $d_t(\pi) = d_b(\pi)/3$ satisfied?*

5 Conclusion

In this paper we have proved that the SORTING BY TRANSPOSITIONS problem is NP-hard, thus answering a long-standing question. However, a number of questions remain open. For instance, does this problem admit a polynomial-time approximation scheme? We note that the reduction we have provided does not answer this question, since it is not a linear reduction. Indeed, by our reduction, if a formula ϕ is not satisfiable, it can be seen that we have $d_t(\pi_{I_\phi}) = d_b(\pi_{I_\phi})/3 + 1$.

Also, do there exist some relevant parameters for which the problem is fixed parameter tractable? A parameter that comes to mind when dealing with the transposition distance is the size of the factors exchanged (e.g., the value $\max\{j-i, k-j\}$ for a transposition $\tau_{i,j,k}$). Does the problem become tractable if we bound this parameter? In fact, the answer to this question is no if we bound only the size of the smallest factor, $\min\{j-i, k-j\}$: in our reduction, this parameter is upper bounded by 6 for every transposition needed to sort π_{I_ϕ}, independently of the formula ϕ.

References

1. Amir, A., Aumann, Y., Benson, G., Levy, A., Lipsky, O., Porat, E., Skiena, S., Vishne, U.: Pattern matching with address errors: Rearrangement distances. J. Comput. Syst. Sci. 75(6), 359–370 (2009)
2. Amir, A., Aumann, Y., Indyk, P., Levy, A., Porat, E.: Efficient computations of ℓ_1 and $\ell_{?8?}$ rearrangement distances. In: Ziviani, N., Baeza-Yates, R.A. (eds.) SPIRE 2007. LNCS, vol. 4726, pp. 39–49. Springer, Heidelberg (2007)
3. Bafna, V., Pevzner, P.A.: Sorting permutations by transpositions. In: SODA, pp. 614–623 (1995)
4. Bafna, V., Pevzner, P.A.: Sorting by transpositions. SIAM J. Discrete Math. 11(2), 224–240 (1998)
5. Benoît-Gagné, M., Hamel, S.: A new and faster method of sorting by transpositions. In: Ma, B., Zhang, K. (eds.) CPM 2007. LNCS, vol. 4580, pp. 131–141. Springer, Heidelberg (2007)
6. Bongartz, D.: Algorithmic Aspects of Some Combinatorial Problems in Bioinformatics. PhD thesis, RWTH Aachen University, Germany (2006)
7. Bulteau, L., Fertin, G., Rusu, I.: Sorting by Transpositions is Difficult. CoRR abs/1011.1157 (2010)
8. Chitturi, B., Sudborough, I.H.: Bounding prefix transposition distance for strings and permutations. In: HICSS, p. 468. IEEE Computer Society, Los Alamitos (2008)
9. Christie, D.A.: Sorting permutations by block-interchanges. Inf. Process. Lett. 60(4), 165–169 (1996)

10. Christie, D.A.: Genome Rearrangement Problems. PhD thesis, University of Glasgow, Scotland (1998)
11. Christie, D.A., Irving, R.W.: Sorting strings by reversals and by transpositions. SIAM J. Discrete Math. 14(2), 193–206 (2001)
12. Cormode, G., Muthukrishnan, S.: The string edit distance matching problem with moves. In: SODA, pp. 667–676 (2002)
13. Dias, Z., Meidanis, J.: Sorting by prefix transpositions. In: Laender, A.H.F., Oliveira, A.L. (eds.) SPIRE 2002. LNCS, vol. 2476, pp. 65–76. Springer, Heidelberg (2002)
14. Elias, I., Hartman, T.: A 1.375-approximation algorithm for sorting by transpositions. IEEE/ACM Trans. Comput. Biology Bioinform. 3(4), 369–379 (2006)
15. Eriksson, H., Eriksson, K., Karlander, J., Svensson, L.J., Wästlund, J.: Sorting a bridge hand. Discrete Mathematics 241(1-3), 289–300 (2001)
16. Feng, J., Zhu, D.: Faster algorithms for sorting by transpositions and sorting by block interchanges. ACM Transactions on Algorithms 3(3) (2007)
17. Fertin, G., Labarre, A., Rusu, I., Tannier, É., Vialette, S.: Combinatorics of genome rearrangements. The MIT Press, Cambridge (2009)
18. Gu, Q.-P., Peng, S., Chen, Q.M.: Sorting permutations and its applications in genome analysis. Lectures on Mathematics in the Life Science, vol. 26, pp. 191–201 (1999)
19. Guyer, S.A., Heath, L.S., Vergara, J.P.: Subsequence and run heuristics for sorting by transpositions. Technical report, Virginia State University (1997)
20. Hartman, T., Shamir, R.: A simpler and faster 1.5-approximation algorithm for sorting by transpositions. Inf. Comput. 204(2), 275–290 (2006)
21. Kolman, P., Waleń, T.: Reversal distance for strings with duplicates: Linear time approximation using hitting set. In: Erlebach, T., Kaklamanis, C. (eds.) WAOA 2006. LNCS, vol. 4368, pp. 279–289. Springer, Heidelberg (2007)
22. Labarre, A.: New bounds and tractable instances for the transposition distance. IEEE/ACM Trans. Comput. Biology Bioinform. 3(4), 380–394 (2006)
23. Labarre, A.: Edit distances and factorisations of even permutations. In: Halperin, D., Mehlhorn, K. (eds.) ESA 2008. LNCS, vol. 5193, pp. 635–646. Springer, Heidelberg (2008)
24. Qi, X.-Q.: Combinatorial Algorithms of Genome Rearrangements in Bioinformatics. PhD thesis, University of Shandong, China (2006)
25. Radcliffe, A.J., Scott, A.D., Wilmer, A.L.: Reversals and transpositions over finite alphabets. SIAM J. Discret. Math. 19, 224–244 (2005)
26. Shapira, D., Storer, J.A.: Edit distance with move operations. In: Apostolico, A., Takeda, M. (eds.) CPM 2002. LNCS, vol. 2373, pp. 85–98. Springer, Heidelberg (2002)

Popular Matchings in the Stable Marriage Problem*

Chien-Chung Huang[1] and Telikepalli Kavitha[2]

[1] Humboldt-Universität zu Berlin, Germany
villars@informatik.hu-berlin.de
[2] Tata Institute of Fundamental Research, India
kavitha@tcs.tifr.res.in

Abstract. The input is a bipartite graph $G = (\mathcal{A} \cup \mathcal{B}, E)$ where each vertex $u \in \mathcal{A} \cup \mathcal{B}$ ranks its neighbors in a strict order of preference. A matching M^* is said to be popular if there is no matching M such that more vertices are better off in M than in M^*. We consider the problem of computing a *maximum* cardinality popular matching in G. It is known that popular matchings always exist in such an instance G, however the complexity of computing a maximum cardinality popular matching was not known so far. In this paper we give a simple characterization of popular matchings when preference lists are strict and a sufficient condition for a maximum cardinality popular matching. We then show an $O(mn_0)$ algorithm for computing a maximum cardinality popular matching in G, where $m = |E|$ and $n_0 = \min(|\mathcal{A}|, |\mathcal{B}|)$.

1 Introduction

Our input is a bipartite graph $G = (\mathcal{A} \cup \mathcal{B}, E)$ where each vertex ranks its neighbors in a strict order of preference. Each vertex $u \in \mathcal{A} \cup \mathcal{B}$ seeks to be assigned to one of its neighbors and u's preference is given by the ordering in u's preference list. Preference lists can be incomplete, which means that a vertex may be adjacent to only some of the vertices on the other side. Note that this is the same as an instance of the *stable marriage* problem with incomplete lists. Let V denote the entire vertex set $\mathcal{A} \cup \mathcal{B}$ and let $|V| = n$ and $|E| = m$. We assume that no vertex is isolated, so $m \geq n/2$.

A matching M is a set of edges no two of which share an endpoint. An edge (u, v) is said to be a *blocking edge* for a matching M if by being matched to each other, both u and v are *better off* than their respective assignments in M: that is, u is either unmatched in M or prefers v to $M(u)$ and similarly, v is either unmatched in M or prefers u to $M(v)$. A matching that admits no blocking edges is called a stable matching. It is known that every instance G admits a stable matching [7] and such a matching can be computed in linear time by a straightforward generalization [5] of the Gale/Shapley algorithm [3] for complete lists.

* Work done when C.-C. Huang was at MPI Saarbrücken and visited TIFR Mumbai under the IMPECS program.

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 666–677, 2011.

Popular Matchings. For any two matchings M and M', we say that vertex u prefers M to M' if u is better off in M than in M' (i.e., u is either matched in M and unmatched in M' or matched in both and prefers $M(u)$ to $M'(u)$). We say that M is more popular than M', denoted by $M \succ M'$, if the number of vertices that prefer M to M' is more than the number of vertices that prefer M' to M.

Definition 1. *A matching M is popular if there is no matching that is more popular than M.*

Popularity is an attractive notion of optimality as a majority vote cannot force a migration from a popular matching. Using the fact that a stable matching has no blocking edges, it can be shown that every stable matching is popular. But not all popular matchings are stable as shown by this simple example: let $\mathcal{A} = \{a_1, a_2\}$ and $\mathcal{B} = \{b_1, b_2\}$ and let the preference lists be as follows: a_1's top choice is b_1 and second choice is b_2 while a_2 has a single neighbor b_1. The vertex b_1's top choice is a_1 and second choice is a_2 while b_2 has a single neighbor a_1. In this instance, the matching $\{(a_1, b_1)\}$ is the only stable matching, while $\{(a_1, b_2), (a_2, b_1)\}$ is popular but unstable.

Our problem. Given $G = (\mathcal{A} \cup \mathcal{B}, E)$ with two-sided preference lists, a stable matching has usually been considered the optimal way of matching the vertices. The fact that there can be *no* blocking edge in a stable matching is a very strong condition and it is known that all stable matchings in G have the same size and match exactly the same set of vertices ([5], Section 4.5.2), let U denote this subset of V. There are many problems, where it is desirable to match more than just the vertices in U, for instance, in allocating training positions to trainees or projects to students, where the total absence of blocking edges is not necessary and a more relaxed definition of stability suffices. The notion of popularity captures this slightly weakened notion of stability: blocking edges are permitted in a popular matching M, nevertheless M has *overall stability* since there is no matching where more vertices are better off than in M. Hence in problems where we are ready to substitute stability with popularity, for the sake of increasing the size of the resulting matching, what we seek is a *maximum* cardinality popular matching. There are instances, as in our example above, where a maximum cardinality popular matching can be twice as large as a stable matching. Our main result is that a maximum cardinality popular matching in $G = (\mathcal{A} \cup \mathcal{B}, E)$ can be computed in $O(mn_0)$ time, where $m = |E|$ and $n_0 = \min(|\mathcal{A}|, |\mathcal{B}|)$.

Related work. Abraham et al. [1] considered the popular matchings problem in the domain of *one-sided* preference lists (only vertices in $\mathcal{A}$ have preferences here and ties are allowed). They gave a structural characterisation of instances that admit popular matchings and described efficient algorithms to determine if a given instance admits a popular matching or not and if so, to compute one with maximum cardinality. The work in [1] on one-sided popular matchings was generalized to the capacitated version by Manlove and Sng [9], the weighted version by Mestre [11], and Mahdian studied random popular matchings [8]. For

instances that do not admit popular matchings, McCutchen [10] considered the problem of computing a least unpopular matching and showed this problem to be NP-hard, while Kavitha, Mestre, and Nasre [6] showed the existence of popular mixed matchings and gave efficient algorithms for computing them.

Gärdenfors [4], who originated the notion of popular matchings, considered this problem in the domain of two-sided preference lists. When ties are allowed in preference lists here, it has recently been shown by Biró, Irving, and Manlove [2] that the problem of computing an arbitrary popular matching in the stable marriage problem is NP-hard. The complexity of the maximum cardinality popular matching problem in the stable marriage problem when preference lists are strict (recall that the popular matchings always exist here) was not known so far and we answer this question here.

2 Structural Results

Theorem 1 gives a simple characterization of popular matchings in an instance $G = (V, E)$ with strictly ordered preference lists. Note that the theorems in this section also hold for non-bipartite graphs with strictly ordered preference lists, however popular matchings need not always exist in the non-bipartite case.

For any vertex u in G and neighbors v and w of u, let $\mathsf{vote}_u(v, w)$ be 1 if u prefers v to w, it is -1 if u prefers w to v, and it is 0 otherwise (i.e., $v = w$). Let M be any matching in G. Mark every edge $e = (u, v)$ in $E \setminus M$ by the pair (α_e, β_e), where $\alpha_e = \mathsf{vote}_u(v, M(u))$ and $\beta_e = \mathsf{vote}_v(u, M(v))$, i.e., α_e is u's vote for v vs. $M(u)$ and β_e is v's vote for u vs. $M(v)$. Also, if M leaves u unmatched, then $\mathsf{vote}_u(v, M(u)) = 1$ where v is any neighbor of u.

Theorem 1. *Let G_M denote the subgraph of G obtained by deleting all edges from G that are marked $(-1, -1)$ wrt M. The matching M is popular in G if and only if the following conditions hold in G_M:*

(i) There is no alternating cycle with respect to M that contains a $(1, 1)$ edge.
(ii) There is no alternating path starting from an unmatched vertex wrt M that contains a $(1, 1)$ edge.
(iii) There is no alternating path with respect to M that contains two or more $(1, 1)$ edges.

Proof. Suppose M is any matching in G that satisfies conditions (i)-(iii). Let M' be any matching in G. Define $\Delta(M', M) = \sum_{u \in V} \mathsf{vote}_u(M'(u), M(u))$. Thus $\Delta(M', M)$ is the difference between the votes that M' gets vs. M and the votes that M gets vs. M'. Note that $M(u)$ or $M'(u)$ can also be the state of being unmatched, which is the least preferred state for any u. We have $M' \succ M$ if and only if $\Delta(M', M) > 0$. We will now show that $\Delta(M', M) \leq 0$ for all matchings M'. This will prove that M is popular.

We need to compute $\sum_u \mathsf{vote}_u(M'(u), M(u))$ now. Mark each edge $e = (u, v)$ of M' by the pair (α_e, β_e) where $\alpha_e = \mathsf{vote}_u(v, M(u))$ and $\beta_e = \mathsf{vote}_v(u, M(v))$. Suppose $\alpha_e = \beta_e = -1$. That is, both u and v are happier with their partners

in M than with each other. Then we can as well assume that M' leaves u and v unmatched, i.e., we can delete the edge (u, v) from M' since this makes no difference to $\mathsf{vote}_u(M'(u), M(u))$ or $\mathsf{vote}_v(M'(v), M(v))$ because both these values were -1 to begin with and they both remain -1 after assuming that u and v are unmatched in M'. Thus in order to evaluate $\sum_u \mathsf{vote}_u(M'(u), M(u))$, we can assume that M' is a matching in the subgraph G_M. Recall that G_M is the subgraph of G obtained by deleting all edges marked $(-1, -1)$ wrt M.

Let ρ be any connected component in $M \oplus M'$. We have $\Delta(M', M) = \sum_\rho \sum_{u \in \rho} \mathsf{vote}_u(M'(u), M(u))$, where the sum is over all the components $\rho \in M \oplus M'$. For vertices u that are isolated in $M \oplus M'$, $M(u) = M'(u)$, so we need to consider only those components ρ that contain two or more vertices. Each such ρ in $M \oplus M'$ is either a cycle or a path; also $\mathsf{vote}_u(M'(u), M(u)) = \pm 1$ for each vertex u in ρ.

Let ρ be a cycle. Since every vertex in ρ is matched by M', we have

$$\sum_{u \in \rho} \mathsf{vote}_u(M'(u), M(u)) = \sum_{e=(u,v) \in \rho \cap M'} \alpha_e + \beta_e \tag{1}$$

where $\alpha_e = \mathsf{vote}_u(v, M(u))$ and $\beta_e = \mathsf{vote}_v(u, M(v))$. Note that for every edge $e \in \rho$, (α_e, β_e) is either $(1, 1)$ or $(-1, 1)$ or $(1, -1)$. But we are given that M satisfies condition (i) of Theorem 1. Hence there is no $(1, 1)$ edge in ρ. Thus for each edge $e \in \rho \cap M'$, $\alpha_e + \beta_e = 0$ and hence $\sum_{e \in \rho \cap M'} \alpha_e + \beta_e = 0$.

Let ρ be a path. Suppose both the endpoints of ρ are matched in M'. Then Eqn. (1) holds here. Since an endpoint of ρ is free in M, by condition (ii) of Theorem 1, we have no $(1, 1)$ edge wrt M in ρ. Thus for each edge $e \in \rho \cap M'$, $\alpha + \beta_e = 0$ and hence $\sum_{e \in \rho \cap M'} \alpha_e + \beta_e = 0$.

Suppose exactly one endpoint of ρ is matched in M'. Then

$$\sum_{u \in \rho} \mathsf{vote}_u(M'(u), M(u)) = -1 + \sum_{e=(u,v) \in \rho \cap M'} \alpha_e + \beta_e$$

since there is one vertex that is matched in M but not in M' and that vertex prefers M to M'. Here too an endpoint of ρ is free in M, and so by condition (ii), we have no $(1, 1)$ edge wrt M in ρ. Thus for each edge $e \in \rho \cap M'$, $\alpha + \beta_e = 0$ and hence $\sum_{u \in \rho} \mathsf{vote}_u(M'(u), M(u)) = -1$ here.

Suppose neither endpoint of ρ is matched in M'. Then

$$\sum_{u \in \rho} \mathsf{vote}_u(M'(u), M(u)) = -2 + \sum_{e=(u,v) \in \rho \cap M'} \alpha_e + \beta_e,$$

since there are two vertices that are matched in M but not in M' and those two vertices prefer M to M'. We use condition (iii) here. There can be at most one $(1, 1)$ edge wrt M in ρ. Thus except for at most one edge e in $\rho \cap M'$, we have $\alpha_e + \beta_e = 0$. So $\sum_{e=(u,v) \in \rho \cap M'} \alpha_e + \beta_e \leq 2$, thus $\sum_{u \in \rho} \mathsf{vote}_u(M'(u), M(u)) \leq 0$.

Hence for each component ρ of $M \oplus M'$, $\sum_{u \in \rho} \mathsf{vote}_u(M'(u), M(u)) \leq 0$. Thus it follows that $\Delta(M', M) \leq 0$. In other words, if M satisfies properties (i)-(iii), then M is popular.

The proof of the converse is straightforward and is omitted here. □

Recall that an augmenting path ρ wrt M is an alternating path where both the endpoints of ρ are unmatched in M. Along with conditions (i)-(iii), let us introduce condition (iv): *There is no augmenting path wrt M in G_M.*

Theorem 2. *If a popular matching M satisfies condition (iv), then M is a maximum cardinality popular matching in G.*

Proof. Since M is a popular matching, we know that M satisfies conditions (i)-(iii) of Theorem 1. Let Q be another matching in G and let $|Q| > |M|$. So $Q \oplus M$ contains an augmenting path p wrt M. We will show using condition (iv) that $Q \oplus p$ is more popular than Q. Thus no matching of size larger than $|M|$ can be popular. Hence M is a maximum cardinality popular matching in G.

Condition (iv) states that there is no augmenting path with respect to M in G_M. So the path p has to use edges outside G_M, i.e., p contains $(-1,-1)$ edges wrt M. Split the path p into subpaths $p_1, p_2, \ldots, p_t$ by removing the $(-1,-1)$ edges from p. Each of the subpaths p_i belongs to G_M.

Each of the paths $p_2, \ldots, p_{t-1}$ can have at most one $(1,1)$ edge wrt M by condition (iii). By condition (ii), neither p_1 nor p_t can contain a $(1,1)$ edge. Thus we have $\sum_{u \in p} \mathsf{vote}_u(Q(u), M(u)) \leq 2(t-2) - 2(t-1)$, where the first term $2(t-2)$ counts the total number of $(1,1)$ edges possible over $p_1, \ldots, p_t$ and the second term $2(t-1)$ counts all the $(-1,-1)$ edges in p (one such edge between p_i and p_{i+1}, for $i = 1, \ldots t-1$). Thus $\sum_{u \in p} \mathsf{vote}_u(Q(u), M(u)) \leq -2$. In other words, $Q \oplus p$ is more popular than Q. □

Lemma 1. *If S is a stable matching in G, then S is a minimum cardinality popular matching in G.*

If M_0 is a matching such that $|M_0| < |S|$, then there has to be an augmenting path ρ wrt M_0 in $M_0 \oplus S$. Using the fact that S has no blocking edges, we can show that $M_0 \oplus \rho$ is more popular than M_0. Thus Lemma 1 follows.

2.1 Good Matchings

Our goal now is to construct a matching M in G that obeys conditions (i)-(iv) (as given in Theorems 1 and 2). Our approach is as follows: suppose we partition the vertex set V into L and R, i.e., $L \dot\cup R = V$, and reorganize the graph G by placing all the vertices of L on the left and all the vertices of R on the right. Note that L and R need not be independent sets. Let M be a matching in $L \times R$, i.e., every edge of M has one endpoint in L and the other endpoint in R.

Definition 2. *Call a matching $M \subseteq L \times R$* good *with respect to (L,R) if the following two properties are satisfied:*

(1) There is no edge marked $(1,1)$ in $L \times R$.
(2) Every edge in $L \times L$ is marked $(-1,-1)$.

Theorem 3. *If M is a matching that is good with respect to some partition (L,R) of V and M is R-perfect, then M satisfies conditions (i)-(iv).*

Proof. Let M be a matching that is good with respect to some partition (L, R) of V and suppose M is R-perfect. Consider the graph G_M. By property (2) of goodness, the set L of vertices is independent in G_M. We now show that conditions (i)-(iv) are obeyed by M.

Condition (i). Let C be an alternating cycle with respect to M in G_M. Since $M \subseteq L \times R$, every edge in $C \cap M$ is an edge of $L \times R$. Thus the number of vertices of L that are in C equals the number of vertices of R that are in C. Since there is no edge in G_M between any pair of vertices in L, the only way an alternating cycle C can exist in G_M is that $C \subseteq L \times R$. By property (1) of goodness of M, there is no $(1, 1)$ edge in $L \times R$. Hence C has no $(1, 1)$ edge wrt M. Thus condition (i) is satisfied.

Condition (ii). Let $p = \langle u_0, u_1, \ldots, u_k \rangle$ be an alternating path with respect to M in G_M such that u_0 is unmatched in M. Since M is R-perfect, the vertex $u_0 \in L$. Since there are no $L \times L$ edges in G_M, the next vertex u_1 in p is in R. Since M uses only $L \times R$ edges, it follows that $u_2 = M(u_1)$ has to be in L, and u_3 is in R since u_2 has no neighbor in L and so on. Thus $p \subseteq L \times R$. Hence by property (1) of goodness of M, condition (ii) is satisfied.

Condition (iii). Let $p = \langle u_0, u_1, \ldots, u_k \rangle$ be any alternating path with respect to M in G_M. We need to show that p has at most one $(1, 1)$ edge wrt M in G_M. Since it is only edges outside M that get marked, we can assume without loss of generality that $(u_0, u_1) \notin M$. If $u_0 \in L$, then the same argument as in the earlier case (which showed that condition (ii) is satisfied) shows that $p \subseteq L \times R$ and so there is *no* $(1, 1)$ edge in p.

So let us assume that $u_0 \in R$. Since there are $R \times R$ edges in G_M, there are two cases.

Case 1: Every odd indexed vertex (i.e., for every i, the vertex u_{2i+1}) is in L. Then the entire path uses only $L \times R$ edges, hence there is no $(1, 1)$ edge in p.

Case 2: Not every odd indexed vertex is in L. Let u_{2j+1} be the first odd indexed vertex that is in R. That is, the edge $(u_{2j}, u_{2j+1}) \in R \times R$. Then u_{2j+2}, which is $M(u_{2j+1})$, has to be in L. Since there are no $L \times L$ edges in G_M, thereafter every odd indexed vertex u_{2k-1} of p is in R and $u_{2k} = M(u_{2k-1})$ has to be in L, so every even indexed vertex in p after u_{2j+1} is in L. Hence there can be only one $R \times R$ edge, which is (u_{2j}, u_{2j+1}), in p. Thus p has at most one $(1, 1)$ edge and condition (iii) is satisfied.

Condition (iv). Suppose there exists an augmenting path $p = \langle u_0, u_1, \ldots, u_{2k+1} \rangle$ wrt M in G_M, that is, the vertices u_0 and u_{2k+1} are unmatched in M. Since M is R-perfect, $u_0 \in L$ and since there are no $L \times L$ edges in G_M, the vertex u_1, which is u_0's neighbor, has to be in R. So the vertex $u_2 = M(u_1)$ is in L, and the vertex u_3, which is u_2's neighbor, has to be in R, and $u_4 = M(u_3)$ has to be in L, and so on. That is, every *even* indexed vertex u_{2i} is in L and every *odd* indexed vertex u_{2i+1} is in R. Thus u_{2k+1} (the other endpoint of p) has to be in R, which contradicts that M is R-perfect, since u_{2k+1} is unmatched in M. Hence there exists no augmenting path wrt M in G_M.

Thus if M is a matching that is good wrt a partition (L, R) and M is also R-perfect, then M has to satisfy conditions (i)-(iv). □

3 The Algorithm

Our input is a bipartite graph $G = (\mathcal{A} \cup \mathcal{B}, E)$. Now we want to find a partition (L, R) and a matching M that is good wrt this partition and which is R-perfect. The vertices in R can be viewed as the "sought-after" vertices and the vertices in L are the vertices that *seek* partners in R. Our algorithm is given below. For convenience, we will refer to the elements of $\mathcal{A}$ and $\mathcal{B}$ as *men* and *women*, respectively. We assume without loss of generality that $|\mathcal{B}| \leq |\mathcal{A}|$.

Algorithm 1. *Input:* $G = (\mathcal{A} \cup \mathcal{B}, E)$ *with strict preference lists*

1. Let S be the stable matching returned by the Gale/Shapley proposal-disposal algorithm on $(\mathcal{A}, \mathcal{B})$. {*That is, men propose and women dispose.*}
2. Let L_1 = set of vertices left unmatched in S; let $R_1 = V \setminus L_1$.
3. $i = 1$.
4. **while** true **do**
5. compute a matching M_i by the proposal-disposal algorithm on (L_i, R_i).
6. **if** M_i is R_i-perfect **then** return M_i.
7. let $A_i \subset \mathcal{A}$ be the set of men in R_i who are unmatched in M_i.
8. set $L'_i = L_i \cup A_i$ and $R'_i = V \setminus L'_i$.
9. compute a matching M'_i by the proposal-disposal algorithm on (L'_i, R'_i).
10. **if** M'_i is R'_i-perfect **then** return M'_i.
11. let B_i be the set of vertices in R'_i left unmatched by M'_i. {*we will show that all these vertices have to be women*}
12. set $L_{i+1} = L_i \cup B_i$ and $R_{i+1} = V \setminus L_{i+1}$.
13. $i = i + 1$.
14. **end while**

We use the Gale/Shapley proposal-disposal algorithm several times in Algorithm 1: for any X, Y such that $X \dot{\cup} Y = V$, when the vertices of X propose to those in Y (i.e., the edge set is restricted to $E \cap (X \times Y)$) and the vertices of Y dispose, every unmatched $x \in X$ proposes in decreasing order of preference and every $y \in Y$ improves in the choice of its partner whenever $M(y)$ gets reassigned. Hence Claim 1 stated below is straightforward. This will be used in our analysis.

Claim 1. *If M is returned by the Gale/Shapley proposal-disposal algorithm on (X, Y), then there is no edge (x, y) in $X \times Y$ such that $\mathsf{vote}_x(y, M(x))$ is 1 and $\mathsf{vote}_y(x, M(y))$ is 1.*

Let $A_0 \subset \mathcal{A}$ and $B_0 \subset \mathcal{B}$ be the sets of those men and women respectively, that are unmatched in any stable matching of $G = (\mathcal{A} \cup \mathcal{B}, E)$. Our initial left side L_1

is $A_0 \cup B_0$. It is easy to see that M_1 is *good* with respect to the partition (L_1, R_1). Property (1) of goodness holds by Claim 1 and property (2) of the goodness of any matching $M_1 \subseteq L_1 \times R_1$ is vacuously true, since L_1 is an independent set in G, and hence in G_M. If every vertex of R_1 receives a proposal, then we have our desired matching. Otherwise, we enter the *second stage* of the first iteration. In the second stage, we move all the unmatched *men* from R_1 to L_1 and run the proposal-disposal algorithm between the new L_1 (call this set L_1') and the new R_1 (call this set R_1') to compute M_1'. We will show that M_1' is good with respect to the new left-right partition.

If M_1' matches all the vertices on the right, then this is the desired matching. Otherwise, let B_1 denote the set of vertices (women) [as is proved below] on the right who are not matched by M_1'. We set $L_2 = L_1 \cup B_1$ (our old L_1 along with B_1) and $R_2 = R_1 \setminus B_1$ (our old R_1 with B_1 deleted) and move to the next iteration of the algorithm. The unmatched men who moved from right to left in the second stage of the first iteration are back on the right now. Their purpose was to identify the set B_1.

At the start of the i-th iteration, we have a partition (L_i, R_i) of V.

- If the matching M_i that results from the proposal-disposal algorithm on (L_i, R_i) is R_i-perfect, then M_i is the desired matching.
- Else let A_i be the set of *men* in R_i who are unmatched in M_i. We run the proposal-disposal algorithm on $(L_i \cup A_i, R_i \setminus A_i)$. If the resulting matching M_i' matches all the vertices of $R_i \setminus A_i$, then M_i' is the desired matching.
- Else let B_i be the set of unmatched vertices (women) [as is proved below] on the right. We set $L_{i+1} = L_i \cup B_i$ and $R_{i+1} = R_i \setminus B_i$; the next iteration begins.

Lemma 2. *For every i, the set $B_i \subseteq \mathcal{B}$.*

Proof. The set B_i is the set of vertices of $R_i' = R_i \setminus A_i$ that are unmatched in M_i'. The matching M_i' is the result of vertices in $L_i' = L_i \cup A_i$ proposing and vertices in R_i' disposing. Note that every vertex of R_i' that was matched in M_i with vertices in L_i proposing, will remain matched in M_i' with $L_i' = L_i \cup A_i$ proposing to $R_i' = R_i \setminus A_i$.

Thus every *man* in R_i who was matched in M_i will remain matched in M_i'. Since we moved all the unmatched men of R_i (this is the set A_i) away from R_i to form $R_i' = R_i \setminus A_i$, every vertex of R_i' that is unmatched in M_i' has to be a *woman*. That is, the set B_i of vertices of R_i' that are unmatched in M_i', is a subset of $\mathcal{B}$. □

Termination of the algorithm. It is easy to see that every iteration takes $O(m+n)$ time, which is $O(m)$. We now show that the while loop in Algorithm 1 runs for at most $|\mathcal{B}|$ iterations.

Lemma 3. *The number of while-loop iterations in Algorithm 1 is at most $|\mathcal{B}|$.*

Proof. To show that termination has to happen within the first $|\mathcal{B}|$ iterations is simple. This is because, if termination does not happen in the i-th iteration,

then $L_{i+1} \supset L_i$ because $B_i \neq \emptyset$ (otherwise termination would have happened in the i-th iteration). Once a woman moves to the left side of the graph, she never moves back to the right side again. Thus there is an iteration k, for some $1 \leq k \leq |\mathcal{B}|$, where either M_k is R_k-perfect or M'_k matches all the *women* in R'_k (in other words, M'_k will be R'_k-perfect), i.e., the termination condition gets satisfied. So the algorithm terminates in the k-th iteration, for some $k \leq |\mathcal{B}|$. □

Correctness of Algorithm 1. We will show in this section that our algorithm maintains the following invariants:

- M_i is good with respect to (L_i, R_i).
- M'_i is good with respect to (L'_i, R'_i).

Note that for all the matchings M_i and M'_i computed in our algorithm, property (1) of goodness is obvious since these matchings are obtained by the proposal-disposal algorithm between the left side and the right side (see Claim 1). What we need to show now is that property (2) of goodness is also obeyed by them.

Recall that M_1 obeys property (2) of goodness. The next lemma shows that M'_1 obeys property (2) of goodness.

Lemma 4. *If* $(a, b) \in L'_1 \times L'_1$, *then* $\mathsf{vote}_a(b, M'_1(a))$ *and* $\mathsf{vote}_b(a, M'_1(b))$ *are* -1.

Proof. Let $e = (a, b)$ be any edge in $L'_1 \times L'_1$, where $a \in \mathcal{A}$ and $b \in \mathcal{B}$. Since $L'_1 = A_0 \cup B_0 \cup A_1$ where $A_0 \cup B_0$ is an independent set, the vertex a has to be in A_1. Observe that every vertex of A_1 will be matched in M'_1 by virtue of the fact that the other vertices in L'_1 comprise the set of vertices unmatched in any stable matching of G. It is easy to see that $a \in A_1$ gets a partner in M'_1 that is at least as good as $S(a)$, where S is the stable matching that results from vertices in $\mathcal{A}$ proposing to vertices in $\mathcal{B}$. Recall that B_0 is the set of women unmatched in S, so a regards $S(a)$ better than any neighbor in B_0. Thus a prefers $M'_1(a)$ to all his neighbors in B_0, hence $\mathsf{vote}_a(b, M'_1(a)) = -1$.

Now we show that $\mathsf{vote}_b(a, M'_1(b)) = -1$. Recall that each man in A_1 was left *unmatched* in M_1: so $b \in B_0$ prefers $M_1(b)$ to all her neighbors in A_1. Observe that no vertex b of B_0 gets dislodged from $M_1(b)$ (a man) by the presence of A_1 in L'_1 since vertices of A_1 propose to *women*. Thus $M'_1(b) = M_1(b)$ and so $\mathsf{vote}_b(a, M'_1(b)) = -1$. This finishes the proof of the lemma. □

This proves that M'_1 is good with respect to (L'_1, R'_1). Now consider any $i \geq 2$. We assume by induction hypothesis on i that the matching $M'_{i-1} \subseteq L'_{i-1} \times R'_{i-1}$ is good with respect to (L'_{i-1}, R'_{i-1}). Lemma 5 shows that then $M_i \subseteq L_i \times R_i$ will be good with respect to (L_i, R_i).

Lemma 5. *If* $(a, b) \in L_i \times L_i$, *then* $\mathsf{vote}_a(b, M_i(a))$ *and* $\mathsf{vote}_b(a, M_i(b))$ *are* -1.

Proof. The set $L_i = A_0 \cup B_0 \cup B_1 \cup \cdots \cup B_{i-1}$. Let $e = (a, b) \in L_i \times L_i$, where $a \in \mathcal{A}$ and $b \in \mathcal{B}$. So a has to be in A_0 and $b \in B_0 \cup \cdots \cup B_{i-1}$. We need to show that every $a \in A_0$ prefers $M_i(a)$ to his neighbors in $B_0 \cup \cdots \cup B_{i-1}$ and every $b \in B_0 \cup \cdots \cup B_{i-1}$ prefers $M_i(b)$ to her neighbors in A_0.

By induction hypothesis, we know that M'_{i-1} is good wrt (L'_{i-1}, R'_{i-1}), where the set $L'_{i-1} = A_0 \cup B_0 \cup \cdots \cup B_{i-2} \cup A_{i-1}$. So for every edge $(a, b) \in L'_{i-1} \times L'_{i-1}$, we know that a prefers $M'_{i-1}(a)$ to any neighbor b in $B_0 \cup \cdots \cup B_{i-2}$. It is easy to see that any $a \in A_0$ gets at least as good a partner in M_i as in M'_{i-1} because $L_i = (L'_{i-1} \setminus A_{i-1}) \cup B_{i-1}$.

The presence of B_{i-1} in L_i does not hurt the men in A_0 when they propose to women in R_i because B_{i-1} is the set of women who were *unmatched* on the right when the men in A_0 were proposing in (L'_{i-1}, R'_{i-1}). Also, the absence of A_{i-1} on the left helps the men in A_0 as they are the only men proposing on the left now in (L_i, R_i) in comparison with (L'_{i-1}, R'_{i-1}).

Thus for any $a \in A_0$ and a's neighbor $b \in B_0 \cup \cdots \cup B_{i-2}$, $\mathsf{vote}_a(b, M_i(a))$ is -1. Also, for any $a \in A_0$ and neighbor $b \in B_{i-1}$, we know that $\mathsf{vote}_a(b, M'_{i-1}(a))$ is -1 since the vertices of B_{i-1} were unmatched in M'_{i-1}, hence $\mathsf{vote}_a(b, M_i(a))$ is -1.

Now we show that for every $(a, b) \in L_i \times L_i$, the vertex b also votes -1 for a vs $M_i(b)$. First, there are no edges between B_0 and A_0. So $b \in B_1 \cup \cdots \cup B_{i-1}$. Each woman in $B_1 \cup \cdots \cup B_{i-1}$ is matched in any stable matching of G and recall that A_0 is the set of men left unmatched in any stable matching of G. Hence when women in $B_1 \cup \cdots \cup B_{i-1}$ propose to men in $\mathcal{A} \setminus A_0$, each woman in $B_1 \cup \cdots \cup B_{i-1}$ gets matched to a man that she considers better than her neighbors in A_0. Thus for any $b \in B_1 \cup \cdots \cup B_{i-1}$ and b's neighbor $a \in A_0$, $\mathsf{vote}_b(a, M_i(b)) = -1$. This finishes the proof of this lemma. □

Suppose M_i does not match all the vertices in R_i. Then we run the second stage of the i-th iteration, where all the men in R_i who were left unmatched by M_i (call this set A_i) are moved to the left. Thus $L'_i = L_i \cup A_i$. That is, $L'_i = A_0 \cup B_0 \cup \cdots \cup B_{i-1} \cup A_i$.

The proposal-disposal algorithm between L'_i and R'_i results in the matching M'_i. We will now show that property (2) of goodness also holds for M'_i. By induction hypothesis on i, we know that the matching M'_{i-1} is good with respect to (L'_{i-1}, R'_{i-1}). The following claim will be helpful to us.

Claim 2. *The set $A_i \subseteq A_{i-1}$, where A_{i-1} is the set of men in R_{i-1} left unmatched by M_{i-1}.*

Proof. The set A_{i-1} was the set of men in R_{i-1} left unmatched by M_{i-1}. Observe that every vertex in R_{i-1} that was matched with $L_{i-1} = A_0 \cup B_0 \cup \cdots \cup B_{i-2}$ proposing, will remain matched with $L_{i-1} \cup A_{i-1}$ proposing to $R_{i-1} \setminus A_{i-1}$. Thus every *man* in R_{i-1} who was matched in M_{i-1} will remain matched in M'_{i-1} with the women in $B_0 \cup \cdots \cup B_{i-2}$ proposing on the left. At the end of the $(i-1)$-th iteration, the set A_{i-1} goes back to the right and the set B_{i-1} moves to the left. With the women in $B_0 \cup \cdots \cup B_{i-1}$ proposing in the i-th iteration, all the men who were matched in the second stage of the previous iteration, continue to remain matched and some vertices of A_{i-1} also possibly get matched. So the set of men in R_i who are unmatched in M_i is a subset of A_{i-1}, i.e., $A_i \subseteq A_{i-1}$. □

We will now show that for any $(a, b) \in L'_i \times L'_i$, $\mathsf{vote}_a(b, M'_i(a)) = -1$ and $\mathsf{vote}_b(a, M'_i(b)) = -1$ in Lemmas 6 and 7, respectively.

Lemma 6. *If* $(a, b) \in L'_i \times L'_i$, *then* $\mathsf{vote}_a(b, M'_i(a)) = -1$.

Proof. We know from Claim 2 that $A_i \subseteq A_{i-1}$. Now B_{i-1} is the set of women in R'_{i-1} left *unmatched* when vertices of L'_{i-1}, which contains $A_0 \cup A_{i-1}$, were proposing on the left in the second stage of the $(i-1)$-th iteration. Hence each man $a \in A_0 \cup A_{i-1}$ prefers $M'_{i-1}(a)$ to any neighbor $b \in B_{i-1}$. Each man in $A_0 \cup A_i$ gets at least as good a partner in M'_i as in M'_{i-1} because there are fewer men proposing now than in the second stage of the $(i-1)$-th iteration as $A_0 \cup A_i \subseteq A_0 \cup A_{i-1}$ and it is only the unmatched women who moved away from R'_{i-1}. Hence all women who belong to $\{M'_{i-1}(a) : a \in A_0 \cup A_{i-1}\}$ are still present in R'_i for $A_0 \cup A_i$ to propose to.

We know from the induction hypothesis that M'_{i-1} is good with respect to (L'_{i-1}, R'_{i-1}). Hence each $a \in A_0 \cup A_{i-1}$ prefers $M'_{i-1}(a)$ to any neighbor in $B_0 \cup \cdots \cup B_{i-2}$. Also, we just argued that $a \in A_0 \cup A_{i-1}$ prefers $M'_{i-1}(a)$ to any neighbor in B_{i-1}. Since $A_i \subseteq A_{i-1}$ and because $M'_i(a)$ is at least as good as $M'_{i-1}(a)$ for all $a \in A_0 \cup A_i$, it follows that $\mathsf{vote}_a(b, M'_i(a)) = -1$ for any edge (a, b) where $a \in A_0 \cup A_i$ and $b \in B_0 \cup \cdots \cup B_{i-2} \cup B_{i-1}$. □

Lemma 7. *If* $(a, b) \in L'_i \times L'_i$, *then* $\mathsf{vote}_b(a, M'_i(b)) = -1$.

Proof. Since $L'_i = A_0 \cup B_0 \cup \cdots \cup B_{i-1} \cup A_i$, for any $(a, b) \in L'_i \times L'_i$, where $a \in \mathcal{A}$ and $b \in \mathcal{B}$, the vertex $a \in A_0 \cup A_i$.

Case 1: Suppose $a \in A_i$. Since A_i is the set of men in R_i who are *unmatched* by M_i, it follows that each of $b \in B_0 \cup \cdots \cup B_{i-1}$ prefers $M_i(b)$ to any neighbor in A_i. Also for any $b \in B_0 \cup \cdots \cup B_{i-1}$, we have $M'_i(b) = M_i(b)$, since it is only *unmatched men* that moved from R_i to the left side to form L'_i. Thus $\mathsf{vote}_b(a, M'_i(b)) = -1$.

Case 2: Suppose $a \in A_0$. Consider any edge between a man in A_0 and a woman $b \in B_0 \cup \cdots \cup B_{i-1}$. In the first place, b has to be in $B_1 \cup \cdots \cup B_{i-1}$ since $A_0 \cup B_0$ is an independent set. Every $b \in B_1 \cup \cdots \cup B_{i-1}$ prefers her partner $M'_i(b)$ to any neighbor in A_0, since A_0 is the set of unmatched men in any stable matching of G. Thus $\mathsf{vote}_b(a, M'_i(b)) = -1$.

Hence for any $b \in B_0 \cup \cdots \cup B_{i-1}$, and any neighbor $a \in A_0 \cup A_i$, we have $\mathsf{vote}_b(a, M'_i(b)) = -1$. □

Thus property (2) of goodness is true for M'_i. We have thus shown that for every i, where $1 \leq i \leq$ number of iterations in our algorithm, M_i is good with respect to (L_i, R_i) and M'_i is good with respect to (L'_i, R'_i). Thus as soon as we find an M_i or an M'_i that matches all the vertices on the right, we have a good matching that matches all the vertices on the right. Thus if M is returned by Algorithm 1, then we know from Theorems 1-3 that M is a maximum cardinality popular matching.

Lemma 3 tells us that within the first $|\mathcal{B}|$ iterations of the while loop, there is an iteration k such that either M_k or M'_k matches all the vertices on the right. Hence the running time of Algorithm 1 is $O(m \cdot |\mathcal{B}|)$ (recall that we assumed $|\mathcal{B}| \leq |\mathcal{A}|$). We can now conclude Theorem 4.

Theorem 4. *A maximum cardinality popular matching in a bipartite graph* $G = (\mathcal{A} \cup \mathcal{B}, E)$ *with 2-sided strict preference lists can be computed in* $O(mn_0)$ *time, where* $m = |E|$ *and* $n_0 = \min(|\mathcal{A}|, |\mathcal{B}|)$.

4 Conclusions

We gave a simple characterization of popular matchings in an instance G with strictly ordered (incomplete) preference lists. We also showed a sufficient condition for a popular matching to be one of maximum cardinality. We introduced the notion of a "good" matching wrt a partition (L, R) of the vertex set and showed that a good matching that is R-perfect has to be a maximum cardinality popular matching. We gave an efficient algorithm to compute such a matching when G is bipartite. When G is non-bipartite, the complexity of determining if a popular matching exists in G is an open problem.

Acknowledgments. We thank the reviewers for their helpful comments.

References

1. Abraham, D.J., Irving, R.W., Kavitha, T., Mehlhorn, K.: Popular matchings. SIAM Journal on Computing 37(4), 1030–1045 (2007)
2. Biró, P., Irving, R.W., Manlove, D.F.: Popular matchings in the marriage and roommates problems. In: Calamoneri, T., Diaz, J. (eds.) CIAC 2010. LNCS, vol. 6078, pp. 97–108. Springer, Heidelberg (2010)
3. Gale, D., Shapley, L.S.: College admissions and the stability of marriage. American Mathematical Monthly 69, 9–15 (1962)
4. Gärdenfors, P.: Match making: assignments based on bilateral preferences. Behavioural Sciences 20, 166–173 (1975)
5. Gusfield, D., Irving, R.W.: The Stable Marriage Problem: Structure and Algorithms. MIT Press, Cambridge (1989)
6. Kavitha, T., Mestre, J., Nasre, M.: Popular mixed matchings. In: Albers, S., Marchetti-Spaccamela, A., Matias, Y., Nikoletseas, S., Thomas, W. (eds.) ICALP 2009. LNCS, vol. 5555, pp. 574–584. Springer, Heidelberg (2009)
7. Knuth, D.E.: Mariages Stables. Les Presses de L'Université de Montreal (1976)
8. Mahdian, M.: Random popular matchings. In: Proceedings of the 7th ACM Conference on Electronic-Commerce, pp. 238–242 (2006)
9. Manlove, D.F., Sng, C.: Popular matchings in the capacitated house allocation problem. In: Azar, Y., Erlebach, T. (eds.) ESA 2006. LNCS, vol. 4168, pp. 492–503. Springer, Heidelberg (2006)
10. McCutchen, M.: The least-unpopularity-factor and least-unpopularity-margin criteria for matching problems with one-sided preferences. In: Laber, E.S., Bornstein, C., Nogueira, L.T., Faria, L. (eds.) LATIN 2008. LNCS, vol. 4957, pp. 593–604. Springer, Heidelberg (2008)
11. Mestre, J.: Weighted popular matchings. In: Bugliesi, M., Preneel, B., Sassone, V., Wegener, I. (eds.) ICALP 2006. LNCS, vol. 4051, pp. 715–726. Springer, Heidelberg (2006)

Center Stable Matchings and Centers of Cover Graphs of Distributive Lattices

Christine Cheng*, Eric McDermid**, and Ichiro Suzuki***

Department of Computer Science,
University of Wisconsin–Milwaukee, Milwaukee, WI 53211, USA
{ccheng,mcdermid,suzuki}@uwm.edu

Abstract. Let I be an instance of the stable marriage (SM) problem. In the late 1990s, Teo and Sethuraman discovered the existence of *median stable matchings*, which are stable matchings that match all participants to their (lower/upper) median stable partner. About a decade later, Cheng showed that not only are they locally-fair, but they are also globally-fair in the following sense: when $G(I)$ is the cover graph of the distributive lattice of stable matchings, these stable matchings are also *medians* of $G(I)$ – i.e., their average distance to the other stable matchings is as small as possible. Unfortunately, finding a median stable matching of I is #P-hard.

Inspired by the fairness properties of the median stable matchings, we study the *center stable matchings* which are the *centers* of $G(I)$ – i.e., the stable matchings whose maximum distance to any stable matching is as small as possible. Here are our two main results. First, we show that a center stable matching of I can be computed in $O(|I|^{2.5})$ time. Thus, center stable matchings are the first type of globally-fair stable matchings we know of that can be computed efficiently. Second, we show that in spite of the first result, there are similarities between the set of median stable matchings and the set of center stable matchings of I. The former induces a hypercube in $G(I)$ while the latter is the union of hypercubes of a fixed dimension in $G(I)$. Furthermore, center stable matchings have a property that approximates the one found by Teo and Sethuraman for median stable matchings. Finally, we note that our results extend to other variants of SM whose solutions form a distributive lattice and whose rotation posets can be constructed efficiently.

Keywords: stable matching, fairness, median, center, partially ordered set, distributive lattice, hypercube.

1 Introduction

In the *stable marriage problem* (SM), there are n men and n women each of whom has a preference list that ranks the opposite gender in a linear order.

* Supported by NSF award CCF-0830678.
** Supported by NSF award CCF-0830678 and UWM Research Growth Initiative.
*** Supported by UWM Research Growth Initiative.

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 678–689, 2011.

A *matching* is a set of n disjoint man-woman pairs; it is *stable* if there is no man-woman pair who prefer each other over their partners in the matching. The goal of the problem is to find a stable matching if one exists. A seminal result of Gale and Shapley in the 1960s [11] states that *every* SM instance has a stable matching that can be computed in $O(n^2)$ time. Today, centralized stable matching algorithms are used to match medical residents to hospitals [21] and students to schools [1,2].

In general, SM instances have many stable matchings. It turns out, however, that the Gale-Shapley algorithm outputs only two kinds: the *man-optimal/woman-pessimal* stable matching and the *woman-optimal/man-pessimal* stable matching. In the man-optimal stable matching, every man is matched to his best partner in all of the stable matchings while simultaneously every woman is matched to her worst partner in all of the stable matchings; the woman-optimal/man-pessimal stable matching has the opposite properties. Hence, in spite of the fact that the Gale-Shapley algorithm solves SM efficiently, we might not want to use the solution as one group is extremely happy while the other group is extremely unhappy. This motivates the problem of finding *fair* stable matchings.

Different notions of fair stable matchings have been considered. Loosely speaking, locally-fair stable matchings seek to ensure that all *participants* are uniformly happy. On the other hand, globally-fair stable matchings are good representatives of the entire *set* of stable matchings. Examples of locally-fair stable matchings include the *egalitarian stable matchings* which maximize the total happiness of all the individuals, the *minimum-regret stable matchings* which maximize the happiness of the unhappiest person in the matching, and the *rank-maximal stable matchings* which maximize the number of individuals matched to their first choice, and within that maximize the number of individuals matched to their second choice, etc. All three of these stable matchings can be computed in polynomial time [12,15]. On the other hand, a stable matching chosen uniformly at random is a globally-fair stable matching. Unfortunately, even in very restricted cases, uniform sampling of stable matchings is computationally hard [4,7].

Interestingly, the two categories of fair stable matchings we have just described are not mutually exclusive. In the late 1990s, Teo and Sethuraman [22] discovered the existence of the *generalized median stable matchings*, which are obtained as follows. Let I be an SM instance and $M(I)$ be its set of stable matchings with $N = |M(I)|$. Assume $M' \subseteq M(I)$. For each man m, order his multiset of partners in M' from his most preferred to his least preferred. Denote by $p_i(m)$ the ith woman in this sorted list, and set $\alpha_i = \{(m, p_i(m))\text{for each man } m\}$. Do the same for the women, and let $\beta_i = \{(p_i(w), w) \text{ for each woman } w\}$. Teo and Sethuraman proved that for $i = 1, \ldots, |M'|$, not only are α_i and β_i stable matchings, but $\alpha_i = \beta_{|M'|-i+1}$. When $M' = M(I)$, α_i is called the *ith generalized median stable matching* of I. The most notable of these are the ones in the "middle" – $\alpha_{(N+1)/2}$ when N is odd and $\alpha_{N/2}$, $\alpha_{N/2+1}$ when N is even – because all participants are matched to their (lower or upper) median stable partners and are therefore locally-fair. About a decade later, Cheng [8] showed that when a

graph structure is imposed on $M(I)$, these stable matchings are also fair in a global sense.

In facility location, when a graph G models a town, and the location of an important structure such as a hospital or a school has to be chosen, a central vertex of G is widely accepted as a good choice because everyone has "equal" access to it. One such candidate is a *median*, a vertex whose total (or average) distance to all the vertices of G is as small as possible. The *median set* of G, $Med(G)$, consists of all the medians of G. Thus, for SM instance I, if $G(I)$ is the graph that represents the global structure of $M(I)$, a median of $G(I)$ is arguably a globally-fair stable matching.

To construct $G(I)$, we define the relation $\preceq$ on $M(I)$. For any two stable matchings μ, μ' of I, let $\mu \preceq \mu'$ if, for each man m, either m has the same partner in μ and μ' or m prefers his partner in μ over his partner in μ'. Conway was the first to recognize that $(M(I), \preceq)$ forms a distributive lattice [16]. Set $G(I)$ to be the cover graph of $(M(I), \preceq)$; i.e., the undirected version of its Hasse diagram. Applying the work of Barbut [3], Cheng [8] showed that when $N = |M(I)|$ is odd, $\alpha_{(N+1)/2}$ is the unique median of $G(I)$, and when N is even, a stable matching μ is a median of $G(I)$ if and only if $\alpha_{N/2} \preceq \mu \preceq \alpha_{(N+1)/2}$. Hence, the elements of $Med(G(I))$, called the *median stable matchings of* I, are both locally *and* globally-fair. Unfortunately, such a fair set of stable matchings comes with a cost: computing a median stable matching of I is #P-hard [8].

Inspired by the nice properties of $Med(G(I))$, we study another class of stable matchings whose fairness criterion is based again on the fact that they are central vertices of $G(I)$. In a graph G, the *eccentricity* of a vertex v, $\mathcal{E}(v)$, is the maximum distance of v to any other node in G. A *center* of G is a vertex whose eccentricity is as small as possible. The *center set* of G, $Cen(G)$, consists of all the centers of G. Like a median, a center of G is also a reasonable spot to build an important facility for a town modeled by G. Call the elements of $Cen(G(I))$ the *center stable matchings of* I. Our basic question is this – how similar are the center and median stable matchings of I? For example, are the center stable matchings of I also locally-fair? Are they hard to compute as well?

To address our questions, we turn to Birkhoff's representation theorem for distributive lattices. Let $\mathcal{P} = (P, \leq)$ be a partially-ordered set or poset. A subset P' of P is a *closed-subset* (also a *down-set* or *order ideal*) of $\mathcal{P}$ if, for every element $p \in P'$, all predecessors of p are also in P'. Let $D(\mathcal{P})$ contain all the closed subsets of $\mathcal{P}$.

Theorem 1. *(Birkhoff[5]) For every distributive lattice $\mathcal{L}$, there is poset $\mathcal{P}_\mathcal{L}$ so that $(D(\mathcal{P}_\mathcal{L}), \subseteq)$ is order-isomorphic to $\mathcal{L}$.*

For some problems whose solutions form a distributive lattice $\mathcal{L}$, $\mathcal{P}_\mathcal{L}$ can be constructed far more quickly than $\mathcal{L}$ itself (see [10] for examples). This is the case for SM. One poset that corresponds to $(M(I), \preceq)$ is the *rotation poset* of I and is denoted by $\mathcal{R}(I)$. When I consists of n men and n women, $|M(I)|$ can be as large as $2^{O(n)}$ but $\mathcal{R}(I)$ has $O(n^2)$ elements and can be constructed in $O(n^2)$ time [13]. For this reason, $\mathcal{R}(I)$ has become one of the most important tools for solving computational problems in SM.

In this paper, we shall frame the problem for finding a center stable matching of I as follows. Let $\mathcal{L}$ be a distributive lattice and $G(\mathcal{L})$ its cover graph. Given $\mathcal{P}_{\mathcal{L}}$, find a center of $G(\mathcal{L})$ (expressed as a closed subset of $\mathcal{P}_{\mathcal{L}}$). Consequently, our results are expressed in terms of $\mathcal{L}$ and its related structures first and then translated to the SM context. We note that many papers have been written about finding centers of graphs (e.g., see [9] and references therein). To our knowledge, however, the graphs are always given *explicitly* so that a center can easily be identified by computing the pairwise distances of the vertices. The underlying goal is to beat this brute force algorithm. In our case, the graphs are given *implicitly*; hence, it is not obvious at all that there is an efficient algorithm for finding a center. Here now are our two main results.

Result 1: We show that the maximum matchings of the comparability graph of $\mathcal{P}_{\mathcal{L}}$, $C(\mathcal{P}_{\mathcal{L}})$, and the centers of $G(\mathcal{L})$ are related: every maximum matching F of $C(\mathcal{P}_{\mathcal{L}})$ can be transformed into a center of $G(\mathcal{L})$ whose eccentricity is $|\mathcal{P}_{\mathcal{L}}| - |F|$. This allows us to prove that when $|\mathcal{P}_{\mathcal{L}}| = k$, finding a center of $G(\mathcal{L})$ can be done in $O(f(k) + k^{2.5})$ time, where $f(k)$ is the time it takes to construct the transitive closure of $\mathcal{P}_{\mathcal{L}}$ from the input representation of $\mathcal{P}_{\mathcal{L}}$.

For SM, the result implies a surprising contrast between computing a median stable matching and a center stable matching – the former is #P-hard while the latter can be done in $O(n^5)$ time where n is the number of men and women in the instance[1]. Thus, center stable matchings are the first type of globally-fair stable matchings we know of that can be computed efficiently.

Result 2: We provide a characterization of the centers of $G(\mathcal{L})$ that essentially says that an element of $\mathcal{L}$ is a center of $G(\mathcal{L})$ if and only if it can be derived from a special kind of maximum matching of $C(\mathcal{P}_{\mathcal{L}})$ which we call a *lowest maximum matching*. An interesting corollary of this result is that the "shape" of the median and center sets of $G(\mathcal{L})$ are similar in the following sense – the median set of $G(\mathcal{L})$ induces a hypercube [19,14] while the center set of $G(\mathcal{L})$ is the union of hypercubes with dimension $|\mathcal{P}_{\mathcal{L}}| - 2|F_{\max}|$.

For SM, the characterization also implies that center stable matchings have a property that approximates the one found by Teo and Sethuraman for median stable matchings: there is a set $M' \subseteq M(I)$ whose stable matchings form a maximum-length chain in $(M(I), \preceq)$ such that $\alpha_{(|M'|+1)/2}$ is a center stable matching of I when $|M'|$ is odd, and $\alpha_{|M'|/2}$ and $\alpha_{|M'|/2+1}$ are center stable matchings of I when $|M'|$ is even.

Finally, all our results about SM extend to its variants as long as their solutions form a distributive lattice, and their rotation posets can be constructed efficiently (e.g., when the preference lists are incomplete, the hospital-residents setting, etc.). The rest of the paper is arranged as follows. Section 2 is the preliminaries section. Section 3 describes the first result while section 4 the second result. We conclude in Section 5.

[1] We note that $O(n^5)$ may seem large, but $n^5 = |I|^{2.5}$ only, where $|I|$ is the input size, because specifying the preference lists of all the participants takes $\Theta(n^2)$ time and space.

2 Preliminaries

A *distributive lattice*[2] $\mathcal{L} = (L, \leq)$ is a poset where (i) for every pair of elements x and y, their greatest lower bound or *meet*, $x \wedge y$, and their least upper bound or *join*, $x \vee y$, exist, and (ii) the meet and join operations distribute over each other. Consequently, $\mathcal{L}$ has a *bottom element* $\hat{0} = \wedge L$ and a *top element* $\hat{1} = \vee L$ so that every other element x of $\mathcal{L}$ lies between $\hat{0}$ and $\hat{1}$; i.e., $\hat{0} < x < \hat{1}$. Many objects form a distributive lattice [20,10]. For instance, when $\mathcal{F}$ is a collection of subsets of X that is closed under set union and intersection, $(\mathcal{F}, \subseteq)$ is a distributive lattice where $X_1 \wedge X_2 = X_1 \cap X_2$ and $X_1 \vee X_2 = X_1 \cup X_2$. It is easy to verify that $D(\mathcal{P})$, the set containing all the closed subsets of poset $\mathcal{P}$, satisfies this condition so $(D(\mathcal{P}), \subseteq)$ is a distributive lattice. Birkhoff's theorem states that *every* distributive lattice can be viewed in this manner. To find the poset $\mathcal{P}_\mathcal{L}$ corresponding to $\mathcal{L}$, Birkhoff showed that it is sufficient to consider the subposet induced by the *join-irreducible elements* of $\mathcal{L}$ – which are the elements with indegree 1 in the Hasse diagram of $\mathcal{L}$. See Figure 1 for an example.

For an SM instance I, however, a poset corresponding to the distributive lattice of I's stable matchings $(M(I), \preceq)$ can be derived from the preference lists of the participants itself. In the SM literature, it is called the *rotation poset of* I and is denoted by $\mathcal{R}(I)$. Its elements are referred to as *rotations*, which are the simplest moves one can make to modify one stable matching into another. When there are n men and n women in I, $\mathcal{R}(I)$ has the following nice properties: (i) it has $O(n^2)$ elements, (ii) a directed acyclic graph that represents $\mathcal{R}(I)$ (i.e., a DAG whose vertices are the elements of $\mathcal{R}(I)$ and whose transitive closure contains exactly the ordered pairs in $\mathcal{R}(I)$) can be constructed in $O(n^2)$ time, (iii) this directed acyclic graph has $O(n^2)$ nodes and $O(n^2)$ edges, and (iv) given a closed subset of $\mathcal{R}(I)$, the stable matching that corresponds to it can be obtained in $O(n^2)$ time. We refer readers to [13] for a more thorough discussion on this topic.

One of the advantages of representing the elements of $\mathcal{L}$ as closed subsets of $\mathcal{P}_\mathcal{L}$ is that their pairwise distances in $G(\mathcal{L})$ can be readily inferred from the sets.

Proposition 1. *For each element x of $\mathcal{L}$, let S_x denote the closed subset of $\mathcal{P}_\mathcal{L}$ that corresponds to x. Then for any two elements x and y of $\mathcal{L}$, the distance between x and y in $G(\mathcal{L})$ is $|S_x \triangle S_y| = |S_x - S_y| + |S_y - S_x|$.*

For example, $S_{\hat{0}} = \emptyset$ while $S_{\hat{1}}$ consists of all the elements of $\mathcal{P}_\mathcal{L}$. Thus, $d(\hat{0}, \hat{1})$ in $G(\mathcal{L})$ is $|\mathcal{P}_\mathcal{L}|$. But for any two elements x and y, $d(x, y)$ in $G(\mathcal{L})$ is at most $|\mathcal{P}_\mathcal{L}|$. Hence, the diameter of $G(\mathcal{L})$ is $|\mathcal{P}_\mathcal{L}|$. We shall make extensive use of Proposition 1 in the latter sections.

Consider the distributive lattice $\mathcal{L}$ in Figure 1. It is straightforward to check that $G(\mathcal{L})$ has only one median, which is h, and one center, which is i. That is, in general, the medians and centers of $G(\mathcal{L})$ need not be the same. In fact, by enlarging this example, one can show that the medians and centers of $G(\mathcal{L})$ can be arbitrarily far apart. Blair [6] showed that every finite distributive lattice is

[2] All lattices discussed in this paper are finite.

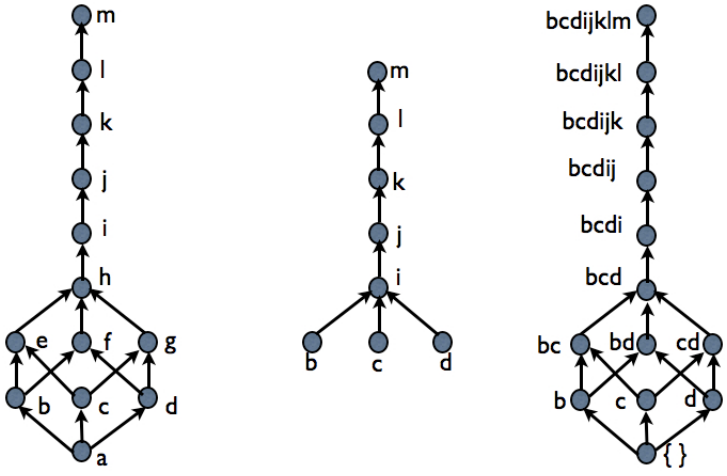


Fig. 1. Consider the distributive lattice $\mathcal{L}$ on the left. The subposet $\mathcal{P}_\mathcal{L}$ induced by the join-irreducible elements of $\mathcal{L}$ is shown in the middle. The closed subsets of $\mathcal{P}_\mathcal{L}$ (labeled without the curly braces) ordered according to the subset relation are shown on the right. Clearly, $\mathcal{L}$ and $(D(\mathcal{P}_\mathcal{L}), \subseteq)$ are isomorphic distributive lattices.

the lattice of stable matchings for some stable marriage instance – hence this example illustrates the differences between median and center stable matchings as well.

3 Finding a Center of $G(\mathcal{L})$

From hereon, when $\mathcal{L}$ is a distributive lattice, we shall refer to its elements as closed subsets of the poset $\mathcal{P}_\mathcal{L}$ and represent them using capital letters V, W, X, Y, etc. Our goal is to show that given $\mathcal{P}_\mathcal{L}$, a center of $G(\mathcal{L})$ can be computed efficiently.

Let $C(\mathcal{P}_\mathcal{L})$ denote the *comparability graph* of $\mathcal{P}_\mathcal{L}$. In this graph, the vertices are the elements of $\mathcal{P}_\mathcal{L}$, and two elements p and q are adjacent if and only if p and q are comparable in $\mathcal{P}_\mathcal{L}$. We begin by considering maximum matchings of $C(\mathcal{P}_\mathcal{L})$. Let F be one such matching with $F = \{(p_1, q_1), (p_2, q_2), \ldots, (p_f, q_f)\}$ such that $p_i < q_i$. Denote by $U(F)$ the set consisting of all the elements of $\mathcal{P}_\mathcal{L}$ not saturated by F, $B(F) = \{p_1, \ldots, p_f\}$ and $T(F) = \{q_1, \ldots, q_f\}$[3]. We shall say that F is a *low maximum matching* if $Z(F) = B(F) \cup U(F)$ is a closed subset of $\mathcal{P}_\mathcal{L}$, and F is a *lowest maximum matching* if, additionally, $B(F)$ is also a closed subset of $\mathcal{P}_\mathcal{L}$. See Figure 2 for an example of these two kinds of matchings.

[3] The letters U, B and T are meant to remind the reader what the sets contain: U for unsaturated elements, B for the bottom elements, and T for the top elements of the maximum matching F.

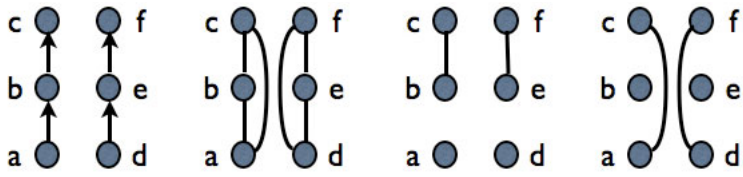


Fig. 2. Suppose $\mathcal{P}_\mathcal{L}$ is the figure on the left. Its comparability graph $C(\mathcal{P}_\mathcal{L})$ is the second figure. The third figure shows a low maximum matching F of $C(\mathcal{P}_\mathcal{L})$ with $Z(F) = \{a, b, d, e\}$. It is, however, not a lowest maximum matching because $B(F) = \{b, e\}$ is not a closed subset of $\mathcal{P}_\mathcal{L}$. The rightmost figure show a lowest maximum matching of $C(\mathcal{P}_\mathcal{L})$.

Lemma 1. *For every distributive lattice $\mathcal{L}$, $C(\mathcal{P}_\mathcal{L})$ has at least one low maximum matching and at least one lowest maximum matching.*

Proof. Assume F is a maximum matching of $C(\mathcal{P}_\mathcal{L})$ but $Z(F)$ is not a closed subset of $\mathcal{P}_\mathcal{L}$. We now show that F can be transformed into another maximum matching that satisfies this property by performing a sequence of *switches*. Suppose $x \in Z(F)$ has a predecessor that is missing from $Z(F)$. This predecessor must be some q' such that such that $(p', q') \in F$. There are two cases to consider:

Case 1: $x \in B(F)$ so that $x = p$ and $(p, q) \in F$. Hence, $p' < q' < p < q$ in $\mathcal{P}_\mathcal{L}$, and $p'p, q'q$ are edges of $C(\mathcal{P}_\mathcal{L})$. *Switch* the edges $(p', q'), (p, q)$ with $(p', p), (q', q)$; that is, replace F with $F' = F - \{(p', q'), (p, q)\} \cup \{(p', p), (q', q)\}$. Clearly, F' is another maximum matching of $C(\mathcal{P}_\mathcal{L})$ with $B(F') = B(F) - \{p\} \cup \{q'\}$ and $U(F') = U(F)$.

Case 2: $x \in U(F)$. Hence, $p' < q' < x$ in $\mathcal{P}_\mathcal{L}$, and $p'x$ is an edge of $C(\mathcal{P}_\mathcal{L})$. *Switch* the edge (p', q') with (p', x); that is, replace F with $F' = F - \{(p', q')\} \cup \{(p', x)\}$. Again, F' is a maximum matching of $C(\mathcal{P}_\mathcal{L})$ with $U(F') = U(F) - \{x\} \cup \{q'\}$ and $B(F') = B(F)$.

Observe that in both cases $Z(F')$ is obtained from $Z(F)$ by replacing the element with a missing predecessor by its missing predecessor. Hence, if we process the elements of $\mathcal{P}_\mathcal{L}$ in a top-down manner, once an element is removed from $Z(F)$, it is never added into the set again. That is, do the following:

Step 1: Topologically sort the elements of $\mathcal{P}_\mathcal{L}$, and let the result be $x_1, x_2, \ldots, x_k$; i.e., for each x_i, all its predecessors occur before it in the ordering.

Step 2: For $i = k$ to 1, if $x_i \in Z(F)$, check that all the predecessors of x_i also belong to $Z(F)$. If not, perform a switch, and update both F and $Z(F)$. At the end of the for loop, return F.

Suppose the output F is not a low maximum matching of $C(\mathcal{P}_\mathcal{L})$. This must mean that for some $x_i \in F$, a predecessor x_h, $h < i$, was part of F when x_i was processed in step 2 but was later removed. For the latter to happen, however, x_h must have a missing predecessor and was replaced by this predecessor. That

is, when x_i was processed, it also had a missing predecessor – a contradiction. Hence, F must be a low maximum matching of $C(\mathcal{P}_\mathcal{L})$.

Now, suppose that F is a low maximum matching of $C(\mathcal{P}_\mathcal{L})$ but not a lowest maximum matching. This means that some $p \in B(F)$ with $(p,q) \in F$ has a predecessor x that is missing from $B(F)$. But since $Z(F)$ is already a closed subset of $\mathcal{P}_\mathcal{L}$, x must lie in $U(F)$. Thus, $x < p < q$ and xq is an edge of $C(\mathcal{P}_\mathcal{L})$. *Switch* the edge (p,q) with (x,q); i.e., replace F with $F' = F - \{(p,q)\} \cup \{(x,q)\}$ so that $B(F') = B(F) - \{p\} \cup \{x\}$ and $Z(F') = Z(F)$. Using the same kind of reasoning as above, we note that steps 1 and 2 (where $Z(F)$ is replaced by $B(F)$) can now transform a low maximum matching of $C(\mathcal{P}_\mathcal{L})$ into a lowest maximum matching. □

In this section, we are mainly interested in the low maximum matchings of $C(\mathcal{P}_\mathcal{L})$. In the next section, the lowest maximum matchings of $C(\mathcal{P}_\mathcal{L})$ will help us characterize the centers of $G(\mathcal{L})$. The next lemma shows how the low maximum matchings of $C(\mathcal{P}_\mathcal{L})$ can be used to identify elements of $\mathcal{L}$ with small eccentricities in $G(\mathcal{L})$. We shall use $rad(G)$ to denote the *radius* of a graph G, the eccentricity of its centers.

Lemma 2. *Suppose $\mathcal{P}_\mathcal{L} = (P_L, \leq)$ has k elements, and F is a low maximum matching of $C(\mathcal{P}_\mathcal{L})$. Then $V = Z(F)$ is an element of $\mathcal{L}$ with $|V| = \mathcal{E}(V) = k - |F|$. Consequently, $rad(G(\mathcal{L})) \leq k - |F|$.*

Proof. By definition, V is a closed subset of $\mathcal{P}_\mathcal{L}$ and must therefore be an element of $\mathcal{L}$. The key property of V that we shall use is this: for every element $q \notin V$, there is a distinct element $p \in V$ such that $p < q$ in $\mathcal{P}_\mathcal{L}$. Consider any other element W of $\mathcal{L}$. Let $\bar{V} = P_L - V$. By the key property of V and the fact that W is a closed subset of $\mathcal{P}_\mathcal{L}$, it must be the case that $|W \cap \bar{V}| \leq |W \cap V|$. Thus, the distance between W and V in $G(\mathcal{L})$ is

$$\begin{aligned} d(W,V) &= |W - V| + |V - W| \\ &= |W \cap \bar{V}| + |V| - |W \cap V| \\ &\leq |V|. \end{aligned}$$

That is, $\mathcal{E}(V) \leq |V|$. But $\emptyset \in \mathcal{L}$ and $d(\emptyset, V) = |V|$ so $\mathcal{E}(V) \geq |V|$. Hence, $\mathcal{E}(V) = |V| = |B(F)| + |U(F)| = |F| + (k - 2|F|) = k - |F|$. Since $rad(G(\mathcal{L}))$ is at most $\mathcal{E}(V)$, its value is at most $k - |F|$ too. □

Next, we show how the centers of $G(\mathcal{L})$ can lead to large matchings in $C(\mathcal{P}_\mathcal{L})$. Our proof makes novel use of König's theorem [17] which states that in a bipartite graph, the size of a maximum matching is equal to the size of a smallest vertex cover.

Lemma 3. *Suppose $\mathcal{P}_\mathcal{L} = (P_L, \leq)$ has k elements. Then $C(\mathcal{P}_\mathcal{L})$ has a matching of size $k - rad(G(\mathcal{L}))$. Hence, a maximum matching of $C(\mathcal{P}_\mathcal{L})$ has size at least $k - rad(G(\mathcal{L}))$.*

Proof. Let V be a center node of $G(\mathcal{L})$ with $\mathcal{E}(V) = k'$. Create the bipartite graph $H[\bar{V}, V]$, where one side consists of elements from $\bar{V} = P_L - V$ and the

other side elements from V. For each $q \in \bar{V}, p \in V$, let qp be an edge if and only if $q > p$ in $\mathcal{P}_\mathcal{L}$. Let F be a maximum matching in $H[\bar{V}, V]$ with $|F| = f$. If $f \geq k - k'$, we are done. So suppose $f < k - k'$. By König's theorem, the smallest vertex cover of $H[\bar{V}, V]$ also has size f. Let $N_{\bar{V}} \cup N_V$ be such a cover with $N_{\bar{V}} \subseteq \bar{V}$ and $N_V \subseteq V$.

Notice that since $N_{\bar{V}} \cup N_V$ is a vertex cover of $H[\bar{V}, V]$, $H[\bar{V}, V]$ has no edges with one endpoint in $\bar{V} - N_{\bar{V}}$ and another endpoint in $V - N_V$. Furthermore, each $p \in N_V$ must cover at least one edge whose other endpoint is in $\bar{V} - N_{\bar{V}}$; otherwise, $N_{\bar{V}} \cup N_V - \{p\}$ is still a vertex cover of $H[\bar{V}, V]$, contradicting the minimality of $N_{\bar{V}} \cup N_V$. By the same reasoning, each $q \in N_{\bar{V}}$ must cover at least one edge whose other endpoint is in $V - N_V$.

Set $W = (\bar{V} - N_{\bar{V}}) \cup N_V$. Let us now argue that W is a closed subset of $\mathcal{P}_\mathcal{L}$. Suppose it is not. Either some $q \in (\bar{V} - N_{\bar{V}})$ or some $p \in N_V$ is missing a predecessor in W. Consider the first case. If q is missing a predecessor x, x either belongs to $N_{\bar{V}}$ or to $V - N_V$. If $x \in N_{\bar{V}}$, we know that x covers some edge xy such that $y \in V - N_V$; i.e., $q > x > y$. It follows that qy is an edge of $H[\bar{V}, V]$. If $x \in V - N_V$, qx is an edge of $H[\bar{V}, V]$. For both of these subcases, we conclude that there is an edge between $\bar{V} - N_{\bar{V}}$ and $V - N_V$, a contradiction. Hence, the first case is not possible. In the second case, if p is missing a predecessor x, x can only belong to $V - N_V$ because V is already a closed subset. But we know that p covers an edge qp such that $q \in \bar{V} - N_{\bar{V}}$; i.e., $q > p > x$. Hence, qx is an edge of $H[\bar{V}, V]$; again, there is an edge between $\bar{V} - N_{\bar{V}}$ and $V - N_V$ so the second case is also not possible. Thus, W is a closed subset of $\mathcal{P}_\mathcal{L}$.

Consider the distance between V and W in $G(\mathcal{L})$:

$$\begin{aligned} d(W, V) &= |W - V| + |V - W| \\ &= |W \cap \bar{V}| + |V| - |W \cap V| = |\bar{V} - N_{\bar{V}}| + |V| - |N_V| \\ &= |\bar{V}| - |N_{\bar{V}}| + |V| - |N_V| = k - (|N_{\bar{V}}| + |N_V|) \\ &= k - f > k - (k - k') = k' \end{aligned}$$

where the inequality is based on our assumption that the maximum matching F in $H[\bar{V}, V]$ has size $f < k - k'$. But $d(W, V) > k'$ is a contradiction since $\mathcal{E}(V) = k'$. It follows that $f \geq k - k'$. Finally, since $H[\bar{V}, V]$ is a subgraph of $C(\mathcal{P}_\mathcal{L})$, $C(\mathcal{P}_\mathcal{L})$ must also have a matching of size at least $k - k'$. □

Here is our main result.

Theorem 2. *When $\mathcal{P}_\mathcal{L} = (P_L, \leq)$ has k elements, and F is a low maximum matching of $C(\mathcal{P}_\mathcal{L})$, then $V = Z(F)$ is a center of $G(\mathcal{L})$ with $|V| = \mathcal{E}(V) = k - |F|$. Furthermore, given $\mathcal{P}_\mathcal{L}$, such a center V can be computed in $O(f(k) + k^{2.5})$ time, where $f(k)$ is the time it takes to construct the transitive closure of $\mathcal{P}_\mathcal{L}$ from the input representation of $\mathcal{P}_\mathcal{L}$.*

Proof. The first statement of the theorem follows directly from Lemmas 2 and 3 so let us prove the second statement. Suppose $Tr(\mathcal{P}_\mathcal{L})$ is the transitive closure of $\mathcal{P}_\mathcal{L}$. Then $C(\mathcal{P}_\mathcal{L})$ is just the undirected version of $Tr(\mathcal{P}_\mathcal{L})$; it has k vertices and $O(k^2)$ edges. Using the algorithm of Micali and Vazirani [18], a maximum

matching F of $C(\mathcal{P}_\mathcal{L})$ can be computed in $O(k^{2.5})$ time. Checking if F is a low maximum matching can be done in $O(k^2)$ time. If it is not, the proof of Lemma 1 describes how it can be transformed into one. Step 1 takes $O(k^2)$ time. In step 2, each element's predecessor is examined at most once; switching takes $O(1)$ time. Hence, step 2 also takes $O(k^2)$ time. Thus, finding a low maximum matching of $C(\mathcal{P}_\mathcal{L})$ given $Tr(\mathcal{P}_\mathcal{L})$ takes $O(k^{2.5})$. Finally, extracting V from the low maximum matching takes $O(k)$ time so the overall running time for computing V is $O(f(k) + k^{2.5})$. □

Corollary 1. *Let I be an SM instance with n men and n women. Finding a center stable matching of I takes $O(n^5)$ time.*

Proof. The poset $\mathcal{R}(I)$ corresponding to the distributive lattice of stable matchings of I can be constructed from I in $O(n^2)$ time, and its transitive closure in $O(n^4)$ time (by repeatedly using depth-first search, for example). Hence, $f(k)$ in Theorem 2 is $O(n^4)$, so that computing a center stable matching of I takes $O(n^5)$ time. □

4 The Center Set of $G(\mathcal{L})$

Applying the techniques used in the previous section, we shall now provide a characterization of the centers of $G(\mathcal{L})$.

Lemma 4. *Let F be a lowest maximum matching of $C(\mathcal{P}_\mathcal{L})$. Then every set of the form $V = B(F) \cup U'$ where $U' \subseteq U(F)$ is a center of $G(\mathcal{L})$.*

Proof. Assume that $\mathcal{P}_\mathcal{L}$ has k elements. Let $V_1 = B(F)$ and $V_2 = B(F) \cup U(F)$. According to Theorem 2, V_2 is a center of $G(\mathcal{L})$ with $|V_2| = \mathcal{E}(V_2) = k - |F|$. To prove the lemma, we need to show that for V with $V_1 \subseteq V \subset V_2$, V is an element of $\mathcal{L}$ with eccentricity $k - |F|$.

First, observe that $U(F)$ must be an antichain in $\mathcal{P}_\mathcal{L}$ for otherwise F is not a maximum matching of $C(\mathcal{P}_\mathcal{L})$. Second, since $Z(F)$ is a closed subset, any predecessor of $x \in U(F)$ must belong to $B(F)$. Hence, $V = B(F) \cup U'$ where $U' \subseteq U(F)$ must also be a closed subset of $\mathcal{P}_\mathcal{L}$ and therefore an element of $\mathcal{L}$. Let W be an element of $\mathcal{L}$. Using the same key property of V mentioned in the proof of Lemma 2 (i.e., for each $q \in T(F)$, there is a distinct $p \in B(F)$) and the fact that W is a closed subset of $\mathcal{P}_\mathcal{L}$, it must be the case that $|W \cap T(F)| \leq |W \cap B(F)|$. So consider the distance between W and V in $G(\mathcal{L})$:

$$\begin{aligned} d(W,V) &= |W - V| + |V - W| \\ &= |W \cap T(F)| + |W \cap (U(F) - U')| + |V| - |W \cap B(F)| - |W \cap U'| \\ &\leq |V| + |W \cap (U(F) - U')| - |W \cap U'| \\ &= |B(F)| + |U'| + |W \cap (U(F) - U')| - |W \cap U'| \\ &\leq |B(F)| + |U'| + |U(F) - U'| \\ &= |B(F)| + |U(F)| = |F| + (k - 2|F|) = k - |F|. \end{aligned}$$

Thus, $\mathcal{E}(V) \leq k - |F|$ so V must be a center of $G(\mathcal{L})$. □

Theorem 3. *Let V be an element of $\mathcal{L}$. Then V is a center of $G(\mathcal{L})$ if and only if there is a lowest maximum matching F of $C(\mathcal{P}_\mathcal{L})$ such that $V = B(F) \cup U'$ with $U' \subseteq U(F)$.*

Proof. The sufficiency direction of the theorem follows immediately from Lemma 4. So consider the necessity direction. Suppose V is a center of $G(\mathcal{L})$. According to Theorem 2, $\mathcal{E}(V) = k - f$ where f is the size of a maximum matching of $C(\mathcal{P}_\mathcal{L})$. In the proof of Lemma 3, we showed that the bipartite graph $H[\bar{V}, V]$ has a maximum matching F of size $k - (k - f) = f$. Thus, F is a maximum matching of $C(\mathcal{P}_\mathcal{L})$ too. Clearly, $V = B(F) \cup U'$ for some $U' \subseteq U(F)$. If F is a lowest maximum matching of $C(\mathcal{P}_\mathcal{L})$, we are done. But suppose it is not. Let us assume the worst case – that F is not even a low maximum matching of $C(\mathcal{P}_\mathcal{L})$. Using the algorithm outlined in the proof of Lemma 1, transform F into a low maximum matching F'. If F' is not a lowest maximum matching, transform it into one using the same algorithm and call it F''. Let us now show that, like F, both F' and F'' are matchings of $H[\bar{V}, V]$.

If F is not a low maximum matching, $Z(F)$ is not a closed subset of $C(\mathcal{P}_\mathcal{L})$. Since V is a closed subset of $C(\mathcal{P}_\mathcal{L})$, the only way this can happen is for some $x \in U(F) \cap \bar{V}$ to have a predecessor q with $(p, q) \in F$. To correct this, the algorithm will switch (p, q) with (p, x) in F. Possibly many more switches like this will have to be performed until $Z(F)$ is a closed subset of $C(\mathcal{P}_\mathcal{L})$. What is true though is that every such switch preserves the property that F is a matching of $H[\bar{V}, V]$. Hence, at the end of the algorithm, the resulting low maximum matching F' is still a matching of $H[\bar{V}, V]$.

Now, if F' is not a lowest maximum matching, it is because $B(F')$ is not a closed subset of $C(\mathcal{P}_\mathcal{L})$. Again, since V is a closed subset of $C(\mathcal{P}_\mathcal{L})$, this must mean that some $p \in B(F')$ with $(p, q) \in F'$ is missing a predecessor $x \in U(F') \cap V$. To correct this, the algorithm will switch (p, q) with (x, q) in F'; that is, F' remains a matching of $H[\bar{V}, V]$. Applying the same reasoning as the previous paragraph, it must be the case that the resulting lowest maximum matching F'' at the end of the algorithm is still a matching of $H[\bar{V}, V]$. As a result, $V = B(F'') \cup U'$ for some $U' \subseteq U(F'')$. □

We close by describing the "shape" of the center set of $G(\mathcal{L})$. As we noted in the introduction, the corollary is quite interesting in light of the fact that the median set of $G(\mathcal{L})$ induces a hypercube [19,14]. For space reasons, the proofs are omitted, but can be found in the full version of the paper available at http://www.cs.uwm.edu/∼ccheng/.

Corollary 2. *Let $\mathcal{L}$ be a distributive lattice such that $\mathcal{P}_\mathcal{L}$ has k elements, and a maximum matching of $C(\mathcal{P}_\mathcal{L})$ has size f. Then,*

1. *The center set of $G(\mathcal{L})$ is a union of hypercubes with dimension $k - 2f$.*
2. *There is a maximum-length chain from $\hat{0}$ to $\hat{1}$ whose middle elements are centers of $G(\mathcal{L})$. Equivalently, in $G(\mathcal{L})$, there is a path whose length is the diameter of $G(\mathcal{L})$ and whose middle nodes are centers of $G(\mathcal{L})$.*

3. *For an SM instance I, there is a set $M' \subseteq M(I)$ whose stable matchings form a maximum-length chain in $(M(I), \preceq)$ such that $\alpha_{(|M'|+1)/2}$ is a center stable matching of I when $|M'|$ is odd, and $\alpha_{|M'|/2}$ and $\alpha_{|M'|/2+1}$ are center stable matchings of I when $|M'|$ is even.*

References

1. Abdulkadiroglu, A., Pathak, P., Roth, A.: The New York City high school match. American Economic Review, Papers and Proceedings 95, 364–367 (2005)
2. Abdulkadiroglu, A., Pathak, P., Roth, A., Sönmez, T.: The Boston public school match. American Economic Review, Papers and Proceedings 95, 368–371 (2005)
3. Barbut, M.: Médiane, distributivité, éloignements, 1961. Reprinted in. Mathématiques et Sciences Humaines 70, 5–31 (1980)
4. Bhatnagar, N., Greenberg, S., Randall, D.: Sampling stable marriages: why spouse-swapping won't work. In: Proc. of SODA 2008, pp. 1223–1232 (2008)
5. Birkhoff, G.: Rings of sets. Duke Mathematical Journal 3, 443–454 (1937)
6. Blair, C.: Every finite distributive lattice is a set of stable matchings. J. Combin. Theory, Ser. A 37, 353–356 (1984)
7. Chebolu, P., Goldberg, L., Martin, R.: The complexity of approximately counting stable matchings. In: Serna, M., Shaltiel, R., Jansen, K., Rolim, J. (eds.) APPROX 2010, LNCS, vol. 6302, pp. 81–94. Springer, Heidelberg (2010)
8. Cheng, C.: Understanding the generalized median stable matchings. Algorithmica 58, 34–51 (2010)
9. Chepoi, V., Dragan, F., Vaxéx, Y.: Center and diameter problems in plane triangulations and quadrangulations. In: Proc. of SODA 2002, pp. 346–355 (2002)
10. Felsner, S.: Lattice structure from planar graphs. Electronic Journal of Combinatorics 11, R15 (2004)
11. Gale, D., Shapley, L.: College admissions and the stability of marriage. American Mathematical Monthly 69 (1962)
12. Gusfield, D.: Three fast algorithms for four problems in stable marriage. SIAM Journal on Computing 16, 111–128 (1987)
13. Gusfield, D., Irving, R.: The Stable Marriage Problem: Structure and Algorithms. The MIT Press, Cambridge (1989)
14. Imrich, W., Klavžar, S.: Product Graphs: Structure and Recognition. Wiley Interscience, Hoboken (2000)
15. Irving, R., Leather, P., Gusfield, D.: An efficient algorithm for the optimal stable marriage. Journal of the ACM 34, 532–544 (1987)
16. Knuth, D.: Mariages Stables. Les Presses de l'Université de Montréal (1976)
17. König, D.: Gráfokés mátrixok. Matematikaiés Fizikai Lapok 38, 116–119 (1931)
18. Micali, S., Vazirani, V.: An $O(\sqrt{V}E)$ algorithm for finding maximum matchings in general graphs. In: Proc. of FOCS 1980, pp. 17–27 (1980)
19. Nieminen, J.: Distance center and centroid of a median graph. Journal of the Franklin Institute 323, 89–94 (1987)
20. Propp, J.: Generating random elements of finite distributive lattices. Electronic Journal of Combinatorics 4 (1997)
21. Roth, A., Peranson, E.: The redesign of the matching market of American physicians: Some engineering aspects of economic design. American Economic Review 89, 748–780 (1999)
22. Teo, C.-P., Sethuraman, J.: The geometry of fractional stable matchings and its applications. Mathematics of Operations Research 23, 874–891 (1998)

VC-Dimension and Shortest Path Algorithms

Ittai Abraham[1], Daniel Delling[1], Amos Fiat[2,⋆],
Andrew V. Goldberg[1], and Renato F. Werneck[1]

[1] Microsoft Research Silicon Valley
{ittaia,dadellin,goldberg,renatow}@microsoft.com
[2] Tel Aviv University
fiat@tau.ac.il

Abstract. We explore the relationship between VC-dimension and graph algorithm design. In particular, we show that set systems induced by sets of vertices on shortest paths have VC-dimension at most two. This allows us to use a result from learning theory to improve time bounds on query algorithms for the point-to-point shortest path problem in networks of low highway dimension, such as road networks. We also refine the definitions of highway dimension and related concepts, making them more general and potentially more relevant to practice. In particular, we define highway dimension in terms of set systems induced by shortest paths, and give cardinality-based and average case definitions.

1 Introduction

The use of navigation software motivated recent work on point-to-point shortest path algorithms with preprocessing for road networks. This produced a large body of experimental work, including hierarchical approaches [13,18], reach-based approaches [15,14], and transit node routing algorithms [3]. (See [10] for a more detailed overview of the literature.) Abraham et al. [2] gave a theoretical justification of some of these algorithms under the assumption of low *highway dimension* (HD), which is believed to be true for road networks. In particular, they proved an $O((\Delta + h \log n \log D)(h \log n \log D))$ query bound on variants of the reach and contraction hierarchy algorithms with polynomial-time preprocessing. Here n is the number of vertices in the input graph, D is the graph diameter, Δ is its maximum vertex degree, and h is its highway dimension. In addition, Abraham et al. proposed a *labeling algorithm* (referred to as a variant of transit node routing in [2]) with a query bound of $O(\Delta + h \log n \log D)$. A very fast implementation of the labeling algorithm has been subsequently developed [1].

The contributions of our paper are as follows.

– We propose a novel application of results from learning theory, in particular those related to VC-dimension [20], to improve the above-mentioned bounds. In the above-mentioned query bounds, we replace the $\log n$ factor by a $\log h$ factor. This is achieved using a boosting-type algorithm. As bounds

⋆ This work was done while the author was visiting Microsoft Research.

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 690–699, 2011.

based on highway dimension are interesting for $h \ll n$, this is a significant improvement.

VC-dimension is heavily used in learning theory and computational geometry. Kleinberg [17] used VC-dimension to analyze failure detection sets in graphs. As far as we know, however, our results are the first application of these techniques to graph algorithm design.

- We generalize the definition of highway dimension and related concepts to set systems. This allows us to apply VC-dimension results. The new definition is also less restrictive and should give smaller highway dimension values for real-world graphs.
- We show that, if shortest paths are unique, they induce set systems of VC-dimension at most two.
- We give a cardinality-based definition of highway dimension and use it to replace the $\log D$ term in the bounds by $\log n$, thus obtaining strongly-polynomial bounds.

Although improved bounds on the shortest path algorithms are interesting in themselves, the relationship to VC-dimension and the application of learning techniques to graph algorithm design are especially exciting. This relationship may have more applications in the design and analysis of graph algorithms. The ideas may also lead to algorithms with better practical performance.

This paper is organized as follows. Section 2 gives basic definitions and notation. Section 3 defines VC-dimension and shows that set systems induced by unique shortest paths have small VC-dimension. In Section 4 we give a new definition of highway dimension and shortest path covers, and discuss the relationship between them. Section 5 combines the results of the previous two sections to improve the existing time bounds. Section 6 explores alternative definitions of highway dimension: average-case and cardinality-based. Section 7 contains concluding remarks.

2 Definitions

A *network* (G, ℓ) consists of a graph $G = (V, E)$ and a length function $\ell : E \to \mathcal{R}$. The input to the preprocessing phase of a point-to-point algorithm is the network (G, ℓ). The preprocessing phase outputs additional data that can be used in the query phase. A query takes as input a *source* vertex s and a *target* vertex t and returns the length of the shortest path $P(s, t)$ from s to t.

For this paper, we assume that the graph is undirected and connected, and that the length function is integral and nonnegative. We denote the length of a path P by $\ell(P)$. Given a nonnegative r, let $B_r(u) = \{v \in V, \ell(P(u, v)) \leq r\}$ be the *ball* of radius r centered at u. Let $D = \max_{u,v \in V} \ell(P(u, v))$ be the diameter of the network, and let Δ be the maximum degree of a vertex in G. We assume that the shortest path between any two vertices is unique. This is a common assumption that can be made without loss of generality, as one can perturb the input to ensure uniqueness.

3 VC-Dimension and USP Systems

A *set system* $(X, \mathcal{R})$ consists of a *base set* X and a collection of its subsets $\mathcal{R}$. A *hitting set* H is a subset of X that intersects every element of $\mathcal{R}$. For $Y \subseteq X$, $\mathcal{R}_{|Y} = \{S \cap Y | S \in \mathcal{R}\}$. Y is *shattered* by $\mathcal{R}$ if $\mathcal{R}_{|Y} = 2^Y$ (i.e., it includes every subset of Y). $(X, \mathcal{R})$ has *VC-dimension* d [20] if d is the smallest integer such that no element $Y \subseteq X$ with $|Y| = d + 1$ can be shattered.

We use the fact that, for set systems with low VC-dimension, there are better hitting set approximation algorithms than in the general case. More specifically, we use the following result, due to Clarkson [6,7] (randomized version) and to Brönnimann and Goodrich [5] (derandomization). This result is based on a boosting-type algorithm from learning theory. An alternative algorithm, based on linear programming relaxations, appears in [11].

Theorem 1. *For a set system with an optimum hitting set of size h and with VC-dimension d, there is an efficient algorithm to find an $O(hd \log(hd))$ hitting set.*

In this paper, we study set systems $(V, \mathcal{R})$ where V is the set of vertices of a graph and each element of $\mathcal{R}$ corresponds to the set of vertices on a shortest path. We refer to such set systems for graphs with unique shortest paths as *USP systems*. To simplify notation, we view a path P both as a path and as a set of its vertices.

The following theorem shows that USP systems have low VC-dimension. This fact is natural and may be known; since we could not find a reference, we give a proof.

Theorem 2. *A USP system $(V, \mathcal{R})$ has VC-dimension at most two.*

Proof. We need to show that no 3-vertex set $Y = \{a, b, c\}$ can be shattered.

First suppose that for some $P \in \mathcal{R}$, $Y \subseteq P$. Without loss of generality, assume that b lies between a and c on the shortest path P. Consider $Z = \{a, c\}$. By uniqueness of shortest paths, any shortest path containing a and c must contain b. Thus Y is not shattered.

Now suppose that for no $P \in \mathcal{R}$, $Y \subseteq P$. Clearly in this case Y is not shattered. □

This theorem in combination with Theorem 1 gives the following result:

Corollary 1. *For a USP-system with an optimum hitting set of size h, there is an efficient algorithm to find an $O(h \log h)$ hitting set.*

For an integer $k \geq 2$, we also study the k-UPS system $(V, \mathcal{R}^k)$, where V is the set of vertices of a graph and the elements of $\mathcal{R}^k$ correspond to sets of vertices on at most k shortest paths. More precisely, $\mathcal{R}^k = \{\bigcup_1^k S_i \mid S_i \in \mathcal{R}\}$. The following theorem shows that k-UPS systems have VC-dimension $O(k \log k)$.

Theorem 3. *A k-USP system $(V, \mathcal{R}^k)$ has VC-dimension $O(k \log k)$.*

Proof. Consider any set $B \subseteq V$ of size $|B| = b$. Due to uniqueness of shortest paths we have $|\mathcal{R}_{|B}| \leq b^2$. Therefore if $2^b > b^{2k}$ then the VC-dimension of $(V, \mathcal{R}^k)$ must be less than b (because there are not enough combinations to shatter any set of size b). For any $k \geq 3$, if we choose $b = 7k \log k$ we have $\log(2^b) = 7k \log k > 2k \log(7k \log k)$. For $k = 2$ we can get a slightly better bound: any set B of size 10 cannot be shattered. If B can be covered by two shortest paths then one of them contains at least 5 points $p_1, \ldots, p_5$ that are on a shortest path. Hence the subset p_1, p_3, p_5 cannot be formed by intersecting B with two shortest paths because any shortest path that covers more than one of these points must also include either p_2 or p_4. We conclude that the VC-dimension of S^k is $O(k \log k)$ for all integers k. □

Corollary 2. *For a k-USP-system with an optimum hitting set of size h, there is an efficient algorithm to find an $O(hk \log k \log(hk \log k))$ hitting set.*

In particular if k is a constant, the bound is $O(h \log h)$.

4 Highway Dimension and Shortest Path Covers

The concept of highway dimension was motivated by the transit node routing (TNR) algorithm [4,3]. The efficiency of TNR on road networks is based on the following observation: after a map is partitioned into regions, all significantly long shortest paths out of each region can be hit by a small number of vertices. The definition of highway dimension in [2] uses the notion of balls. In this section we give a more flexible definition using neighborhoods and set systems.

For $q > r > 0$, denote by P_q the collection of all shortest paths of length at most q, and by P_q^r the collection of all shortest paths P with $r < \ell(P) \leq q$. Define the *r-neighborhood* of v by $N_r(v) = \bigcup(P \in P_r : v \in P)$. (Here we think of P as a set of vertices, so $N_r(v)$ is also a set.) Note that $N_r(v) = B_r(v)$, but the definition of N_r is more general; in particular, it works if one defines the collection of sets P_r in an arbitrary way. Define the *r-neighborhood set system* $S_r(v) = (V, \{P \in P_r^{r/2} : P \cap N_r(v) \neq \emptyset\})$, i.e., $S_r(v)$ is induced by the set of all shortest paths of length between $r/2$ and r intersecting the r-neighborhood of v. We use S_{2r} in our new definition of highway dimension.

The *highway dimension* (HD) of a network (G, ℓ) is the smallest h such that $\forall r > 0$, $\forall v \in V$, there is a hitting set of $S_{2r}(v)$ of size at most h. A related concept is that of a *shortest path cover (SPC)*. For $r > 0$, an (h, r)-SPC is a set $C \subseteq V$ such that C hits all paths in P_{2r}^r and $\forall v \in V, |N_{2r}(v) \cap C| \leq h$ (C is sparse).

The previous definition of HD used B_{2r} instead of N_{2r} and required the hitting set to cover all paths in $B_{4r}(v) \cap P_{2r}^r$. While B_{2r} and N_{2r} are the same, the requirement to hit all the paths is stronger, as some of these do not intersect $B_{2r}(v)$. For individual instances, including real-life ones, the new definition is likely to give smaller HD values.

The basic theorem of [2] relating HD and SPC holds under the new definitions, and the proof is similar.

Theorem 4. *If the highway dimension of (G, ℓ) is h, then for any $r > 0$ the smallest hitting set for P_{2r}^r is an (h, r)-SPC.*

Proof. Let H^* be the smallest hitting set for P_{2r}^r. We prove that H^* is an (h, r)-SPC. Suppose for contradiction that for some u, $U = H^* \cap N_{2r}(u)$ and $|U| > h$. By the definition of h, there is a hitting set H for $S_{2r}(u)$ with $|H| \leq h$. By the definition of $S_{2r}(u)$, H hits all shortest paths in P_{2r}^r hit by U. Therefore $(H^* \setminus U) \cup H$ is smaller than H^* and hits all shortest paths in P_{2r}^r, contradicting the optimality of H^*. □

5 Improved Bounds

Although Theorem 4 guarantees the existence of good covers, computing the set H^* (used in the proof) is NP-hard. This motivates building approximate SPCs, i.e., (h', r)-SPCs such that h' bigger than (but close to) h. As Abraham et al. [2] show, a greedy algorithm [16] (which in each iteration adds to the solution the vertex that hits the most uncovered paths) produces $(O(h \log n), r)$-SPCs.

Next we show how to compute $(O(h \log h), r)$-SPCs efficiently. The result is more complicated than a direct application of Corollary 1 because we want to get a sparse hitting set instead of just a small one.

Theorem 5. *If the highway dimension of (G, ℓ) is h, then for any $r > 0$ we can compute an $(O(h \log h), r)$-SPC in polynomial time.*

Proof. Let c be the constant hidden by the big "O" notation in Theorem 1. By Theorem 2, for any $S_{2r}(u)$, we can efficiently compute a hitting set of size at most $h' = 2hc \log(2h)$. Our goal is to build an (h', r)-SPC.

We maintain a hitting set A for P_{2r}^r. We can start with any such A computed in polynomial time, for example by the greedy algorithm. We show that if A is not sparse, then we can replace A by a smaller hitting set for P_{2r}^r in polynomial time. Iterating this construction $O(n)$ times, we will eventually stop with a sparse A.

While A is not sparse, there exists some u such that $U = A \cap N_{2r}(u)$ and $|U| > h'$. We efficiently compute a hitting set H for $S_{2r}(u)$ with $|H| \leq h'$. $(A \setminus U) \cup H$ is a hitting set for P_{2r}^r that is smaller than A. □

The results of [2] imply the following lemma.

Lemma 1. *Suppose that one can, in polynomial time, compute (k, r)-SPCs for all r and some parameter k. Then the reach [14] and contraction hierarchy [13] algorithms have a polynomial-time preprocessing routine that adds $O(k \log D)$ edges to the graph, and the corresponding query algorithms run in $O((\Delta + k \log D)(k \log D))$ time. For the labeling algorithm [12], the preprocessing adds the same number of edges and the query runs in $O(\Delta + k \log D)$ time.*

The results of [2] use a greedy hitting set algorithm to get $k = O(h \log n)$. Theorem 5 gives $k = O(h \log h)$, and we obtain the following improved bounds:

Theorem 6. *The reach and contraction hierarchy algorithms have a polynomial-time preprocessing routine that adds $O(h \log h \log D)$ edges to the graph, and the corresponding query algorithms run in $O((\Delta + h \log h \log D)(h \log h \log D))$ time. For the labeling algorithm, the preprocessing adds the same number of edges and the query runs in $O(\Delta + h \log h \log D)$ time.*

6 Extensions

6.1 Labeling Algorithm and Average Dimension

Road networks are non-uniform. At the same scale, some regions contain structures requiring denser local covers, while most areas do not. So far we worked with the worst-case definition of highway dimension. In this section we give an average-case definition. For many graphs, this definition should give a much smaller value. We use the new definition to prove an expected-time bound on the labeling algorithm.

The labeling algorithm was introduced in [12,19] and analyzed for low highway dimension networks in [2]. An $O(\log n)$-approximation algorithm for the smallest average label size appears in [8]. The labeling algorithm works in two stages. The preprocessing stage computes, for each vertex v, a *label* $L(v)$. The label consists of a set of vertices w, together with their respective distances $\text{dist}(v, w)$ from v. The labels have the following *cover property*: For every pair of distinct vertices s and t, $L(s) \cap L(t)$ contains a vertex u on the shortest path from s to t. The query stage of the labeling algorithm is quite simple. Given s and t, find the vertex $u \in L(s) \cap L(t)$ that minimizes $\text{dist}(s, u) + \text{dist}(u, t)$ and return the length of the corresponding path.

Theorem 7. *Suppose the average label size is k. Then for random queries, the expected query time for the labeling algorithm is $O(k)$.*

Proof. For two vertices with label sizes x and y, the query time is $O(x + y)$. The result follows by the linearity of expectation. □

We do not have a similar average-case result for hierarchical and reach-based algorithms because they are asymmetric: Some vertices are used in many point-to-point computations, and they can be the ones with high degree.

We define the *average highway dimension* (AHD) of a network (G, ℓ) by taking the maximum, over all r, of the average minimum hitting set size for the $2r$-neighborhood set system of a vertex. More formally, let $h(v, r)$ be the size of the minimum hitting set for $S_{2r}(v)$. The AHD is defined by

$$\max_r \frac{\sum_v h(v, r)}{n}.$$

A related concept is the *average shortest path cover (ASPC)*. For $r > 0$, an (h, r)-ASPC is a set $C \subseteq V$ such that C hits all paths in P_{2r}^r and is sparse on average:

$$\frac{\sum_v |N_{2r}(v) \cap C|}{n} \leq h.$$

Remark: Note that the HD and AHD definitions use ∞- and 1-norm, respectively. We can generalize these definitions for k-norms. For an integer $k \geq 1$, let $h^k(v,r) = (h(v,k))^k$. We define HD^k by

$$\max_r \sqrt[k]{\frac{\sum_v h^k(v,r)}{n}}.$$

Similarly, we can define (h,r)-SPC^k to be a hitting set for P_{2r}^r such that

$$\sqrt[k]{\frac{\sum_v |N_{2r}(v) \cap C|^k}{n}} \leq h.$$

As $\lim_{k\to\infty} \sqrt[k]{\frac{\sum_v h^k(v,r)}{n}} = \max_v h(v,r)$, we have that HD$^\infty$ is the same as HD.

The following analog of Theorem 4 holds. (The proof is similar, with modifications analogous to those used in the proof of Theorem 9 below.)

Theorem 8. *If the AHD of (G,ℓ) is h, then for any $r > 0$ the smallest hitting set for P_{2r}^r is an (h,r)-ASPC.*

Next we prove an analog of Theorem 5. The proof is similar but slightly more complicated.

Theorem 9. *If the AHD of (G,ℓ) is h, then for any $r > 0$ we can compute an $(O(h \log h), r)$-ASPC in polynomial time.*

Proof. Let $V = \{v_1, \ldots, v_n\}$. Fix $r > 0$ and let $h_i = h(v_i, r)$. By definition, $h \geq \frac{\sum_i h_i}{n}$. Let c be the constant hidden by the big "O" notation in Theorem 1.

We maintain a hitting set A for P_{2r}^r. We can start with A computed by the greedy algorithm, for example. Suppose for some v_i, $U = A \cap N_{2r}(v_i)$ and $|U| > ch_i \log h_i$. We efficiently compute a hitting set H for $S_{2r}(u)$ with $|H| \leq ch_i \log h_i$. $(A \setminus U) \cup H$ is a hitting set for P_{2r}^r that is smaller than A.

By iterating the above construction at most n times, we get a hitting set A such that for all $1 \leq i \leq n$, $|A \cap N_{2r}(v_i)| \leq ch_i \log h_i$. Since the average of h_i is at most h and log is a concave function, we have

$$\sum_i ch_i \log h_i \leq c \log h \sum_i h_i \leq cnh \log h$$

and the theorem follows. □

Remark: The two theorems above also hold for HD^k and SPC^k.

For the labeling algorithm, we can state an expected time bound in terms of the average highway dimension.

Lemma 2. *If the AHD of a network is h, then we can compute labels of average size $O(h \log h \log D)$ in polynomial time.*

Proof. We build labels similarly to the worst case [2]. For $i = 0, \ldots, \log D$, let C_i be an $(O(h \log h), 2 \cdot 2^i)$-ASPC cover. Define $L(v) = \cup_i (C_i \cap N_{2 \cdot 2^i}(v))$. It is easy to see that the Ls have the labeling property. We get the size bound as follows:

$$\sum_v |L(v)| = \sum_i \sum_v |C_i \cap N_{2 \cdot 2^i}(v)| = O(nh \log h \log D),$$

as desired. □

Theorem 10. *If the average highway dimension of a network is h, the expected running time of the labeling algorithm with polynomial-time preprocessing on a random query is $O(\Delta + h \log h \log D)$.*

Note that all constructions for the average case are the same as for the worst case, so the worst-case bounds (in terms of the worst-case highway dimension) hold in addition to the average-case bounds.

6.2 Cardinality-Based Highway Dimension

We can state an alternative definition of highway dimension using path cardinality instead of path length. For the definition to be robust, we assume that the input graph has no vertices of degree two. (This can be trivially enforced during a preprocessing step, and does not affect the correctness of routing algorithms.)

To define cardinality-based highway dimension and shortest path covers, we redefine P_q and P_q^r as follows. For $q > r > 0$, denote by P_q the collection of all shortest paths P of cardinality $|P| \leq q$, and by P_q^r the collection of all shortest paths P with $r < |P| \leq q$. Our set-based definition of highway dimension and shortest path covers is general enough for all the results to go through. A major difference is that the maximum path length is D but the maximum path cardinality is n, so in the bounds we get $\log n$ instead of $\log D$, i.e., the bounds become strongly polynomial. Note that, in general, the value of h will be different for the two definitions.

Note that these results apply to the average-case bounds of Section 6.1 as well. This alternative definition of highway dimension might be useful in the analysis of methods such as *highway hierarchies* [18], which are based on the cardinality of shortest paths.

7 Concluding Remarks

Kleinberg [17] showed that VC-dimension can be useful in the analysis of graph algorithms. We extend this observation by showing that it can be used to design graph algorithms. This suggests that designers of graph algorithms can potentially benefit from relevant developments in learning theory.

Consider the cardinality-based definitions of HD and SPC. These definitions are set-theoretic. The shortest-path domain gives only the base set and the subset families. For the cardinality case, S_r, N_r, HD, and (h, r)-SPC are well-defined

for any set system. Furthermore, Theorem 4 holds. If the highway dimension is h and the VC-dimension of the set system is d, we can find an $(O(h \log n), r)$-SPC or an $(O(hd \log(hd)), r)$-SPC in polynomial time. These generalizations make it possible to apply the notion of highway dimension in other contexts.

Our results apply to undirected graphs, while road networks are directed. Extensions to the directed case, discussed in [2], apply under our new definitions. Unfortunately these extensions require additional assumptions, such as bounded asymmetry. It would be interesting to relax or eliminate these assumptions.

An important open question is that of designing practical algorithms for building good SPCs on large graphs with low highway dimension. Such algorithms will be useful in practice. Abraham et al. [1] improve contraction hierarchies by applying the greedy algorithm late in the preprocessing stage, when the graph shrinks to a sufficiently small size. A faster SPC algorithm may bring further improvements. Boosting techniques could potentially lead to practical improvements of this approach. Note that recent developments in one-to-all shortest path algorithms [9] make a wider class of algorithms potentially practical.

Another interesting experimental question is that of computing good upper and lower bounds on the highway dimension of real road networks. As we have discussed, our new definition of highway dimension is likely to yield smaller values. Experimental studies can also help to determine how much smaller the AHD of real road networks is compared to their highway dimension, and whether the cardinality-based highway dimension of these networks is smaller than the length-based one.

On the theoretical side, it would be interesting to develop better bounds on approximating SPCs, either directly or by improving the results of Theorem 1. It may also be interesting to know if Theorem 3 is tight.

Acknowledgements

We would like to thank Anupam Gupta and Kunal Talwar for helpful discussions.

References

1. Abraham, I., Delling, D., Goldberg, A.V., Werneck, R.F.: A Hub-Based Labeling Algorithm for Shortest Paths in Road Networks. In: Pardalos, P.M., Rebennack, S. (eds.) SEA 2011. LNCS, vol. 6630, pp. 230–241. Springer, Heidelberg (2011)
2. Abraham, I., Fiat, A., Goldberg, A.V., Werneck, R.F.: Highway Dimension, Shortest Paths, and Provably Efficient Algorithms. In: Proceedings of the 21st Annual ACM–SIAM Symposium on Discrete Algorithms (SODA 2010), pp. 782–793 (2010)
3. Bast, H., Funke, S., Matijevic, D., Sanders, P., Schultes, D.: In Transit to Constant Shortest-Path Queries in Road Networks. In: Proceedings of the 9th Workshop on Algorithm Engineering and Experiments (ALENEX 2007), pp. 46–59. SIAM, Philadelphia (2007)
4. Bast, H., Funke, S., Sanders, P., Schultes, D.: Fast Routing in Road Networks with Transit Nodes. Science 316(5824), 566 (2007)

5. Brönnimann, H., Goodrich, M.: Almost Optimal Set Covers in Finite VC-dimension. Discrete and Computational Geometry 14, 463–497 (1995)
6. Clarkson, K.L.: A Las Vegas Algorithm for Linear Programming When the Dimension is Small. In: Proc. 29th IEEE Annual Symposium on Foundations of Computer Science, pp. 452–456. IEEE, Los Alamitos (1988)
7. Clarkson, K.L.: Algorithms for Polytope Covering and Approximation. In: Proc. 3rd Workshop Algo. Data Struct., pp. 246–252. Springer, New York (1993)
8. Cohen, E., Halperin, E., Kaplan, H., Zwick, U.: Reachability and Distance Queries via 2-hop Labels. SIAM J. Comput. 32 (2003)
9. Delling, D., Goldberg, A.V., Nowatzyk, A., Werneck, R.F.: PHAST: Hardware-Accelerated Shortest Path Trees. In: 25th International Parallel and Distributed Processing Symposium (IPDPS 2011). IEEE Computer Society, Los Alamitos (2011)
10. Delling, D., Sanders, P., Schultes, D., Wagner, D.: Engineering Route Planning Algorithms. In: Lerner, J., Wagner, D., Zweig, K.A. (eds.) Algorithmics of Large and Complex Networks. LNCS, vol. 5515, pp. 117–139. Springer, Heidelberg (2009)
11. Even, G., Rawitz, D., Shahar, S.: Hitting Sets when the VC-dimension is Small. Inf. Process. Lett. 95, 358–362 (2005)
12. Gavoille, C., Peleg, D., Pérennes, S., Raz, R.: Distance Labeling in Graphs. J. Algorithms 53(1), 85–112 (2004)
13. Geisberger, R., Sanders, P., Schultes, D., Delling, D.: Contraction Hierarchies: Faster and Simpler Hierarchical Routing in Road Networks. In: McGeoch, C.C. (ed.) WEA 2008. LNCS, vol. 5038, pp. 319–333. Springer, Heidelberg (2008)
14. Goldberg, A., Kaplan, H., Werneck, R.: Reach for A*: Shortest Path Algorithms with Preprocessing. In: Demetrescu, C., Goldberg, A., Johnson, D. (eds.) The Shortest Path Problem: Ninth DIMACS Implementation Challenge, pp. 93–140. AMS, Providence (2009)
15. Gutman, R.: Reach-based Routing: A New Approach to Shortest Path Algorithms Optimized for Road Networks. In: Proc. 6th International Workshop on Algorithm Engineering and Experiments, pp. 100–111 (2004)
16. Johnson, D.: Approximation Algorithms for Combinatorial Problems. J. Comput. Syst. Sci. 9, 256–278 (1974)
17. Kleinberg, J.: Detecting a Network Failure. In: Proc. 41nd IEEE Annual Symposium on Foundations of Computer Science, pp. 231–239. IEEE, Los Alamitos (2008)
18. Sanders, P., Schultes, D.: Highway Hierarchies Hasten Exact Shortest Path Queries. In: Brodal, G.S., Leonardi, S. (eds.) ESA 2005. LNCS, vol. 3669, pp. 568–579. Springer, Heidelberg (2005)
19. Thorup, M., Zwick, U.: Approximate Distance Oracles. J. ACM 52(1), 1–24 (2005)
20. Vapnik, V., Chervonenkis, A.: On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities. Theory of Probability and its Applications 16, 264–280 (1971)

Characterizing Arithmetic Circuit Classes by Constraint Satisfaction Problems
(Extended Abstract)

Stefan Mengel*

Institute of Mathematics, University of Paderborn, D-33098 Paderborn, Germany
stefan.mengel@mail.uni-paderborn.de

Abstract. We explore the expressivity of constraint satisfaction problems (CSPs) in the arithmetic circuit model. While CSPs are known to yield VNP-complete polynomials in the general case, we show that for different restrictions of the structure of the CSPs we get characterizations of different arithmetic circuit classes. In particular we give the first natural non-circuit characterization of VP, the class of polynomial families efficiently computable by arithmetic circuits.

1 Introduction and Related Work

The complexity class VP has a very natural definition: It is the class of families of polynomials computable by arithmetic circuits efficiently, i.e., by families of arithmetic circuits of polynomial size. Despite this apparent naturality there is one irritating aspect in which VP differs from other arithmetic circuit classes: There are no known natural complete problems for VP – artificial ones can be constructed – and no known natural characterizations of VP that do not in one form or another depend on circuits. This puzzling feature of VP raises the question whether VP is indeed the right class for measuring natural efficient computability. This scepticism is further strengthened by the fact that Malod and Portier [14] have shown that many natural problems from linear algebra are complete for VP_{ws}, a subclass of VP. Thus the search for complete problems or natural characterizations of VP is an interesting and meaningful problem in algebraic complexity. In this paper we give such a natural characterization of VP and other classes by constraint satisfaction problems.

Constraint satisfaction problems (CSPs) are a classical problem in complexity theory and among the first shown to be NP-complete. In a seminal paper Schaefer [15] characterized the complexity of boolean CSPs by showing a famous dichotomy theorem: if all constraints are chosen from a small class which he completely describes, then the corresponding CSP is in P, otherwise it is NP-complete. This result has spawned several follow up results, in one of which Briquel and Koiran [5] gave a similar dichotomy result in the arithmetic circuit

* Supported by the Research Training Group GK-693 of the Paderborn Institute for Scientific Computation (PaSCo)

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 700–711, 2011.

model. To a family (Φ_n) of CSPs they assign a polynomial family $(P(\Phi_n))$. They show that there is a small set S of constraints with the following property: If a family (Φ_n) of CSPs is built of constraints in S only, then $(P(\Phi_n)) \in \mathsf{VP}$. On the other hand if CSPs may be constructed with the help of any constraint not in S, one can construct such a CSP-family (Φ_n) such that $(P(\Phi_n))$ is VNP-complete.

Because CSPs are immensely important for practical purposes, researchers especially in database theory and AI tried to circumvent Schaefer's result by finding feasible subclasses of CSPs. The key idea here is not to restrict the individual constraints but instead to restrict the structure of the CSPs built with these constraints. It was shown [1] that if one restricts the problem to so called acyclic CSPs, then the resulting CSPs are solvable in P. It was even shown that acyclic CSPs are parallelizable, but the exact complexity of the problem was open for some time. Gottlob et. al [11] solved this question by proving that acyclic CSPs are complete for the class LOGCFL. This result easily extends to CSPs of bounded treewidth.

Treewidth is a crucial graph parameter for many algorithmic problems on graphs. Often hard problems become feasible if one bounds the treewidth of the inputs by a constant. During the last years treewidth has found its way into arithmetic circuit complexity. This was started by Courcelle et al. [8] who showed that generating functions of graph problems expressible in monadic second order logic have small arithmetic circuits for graphs of bounded treewidth. This line of research was continued by Flarup et al. [13] who improved these upper bounds and showed matching lower bounds for some familes of polynomials: On the one hand the permanent and the hamilton polynomial on graphs of bounded treewidth can be computed by arithmetic formulas of polynomial size. On the other hand all arithmetic formulas can be expressed this way. Briquel, Koiran and Meer [6,12] – building on a paper by Fischer et al. [9] which deals with counting problems – considered polynomials defined by CNF formulas of bounded treewidth (see also Section 3).

In this paper we unify these different lines of work: We complement the general infeasibility results of Briquel and Koiran [5] by showing feasible subclasses of polynomials assigned to CSPs. Also, our paper can be seen as an extension of the work of Briquel, Koiran and Meer [12,6] by generalization from CNF-formulas to general CSPs. We introduce two kinds of polynomials for CSPs and show that they characterize the hierarchy $\mathsf{VP}_e \subseteq \mathsf{VP}_{ws} \subseteq \mathsf{VP} \subseteq \mathsf{VNP}$ of arithmetic circuit classes commonly considered (cf Section 2.1), respectively, for different classes of CSPs. Boolean bounded treewidth or pathwidth CSPs capture VP_e, while in the non-boolean case we get VP_{ws} for bounded pathwidth and VP for bounded treewidth. We also explain where exactly the difference in expressivity between boolean and non-boolean CSPs comes from. We prove that if each variable can take only a constant number of values in satisfying assignments of each constraint in non-boolean CSPs, then these CSPs capture VP_e again. In boolean CSPs each variable trivially takes only at most 2 values in the satisfying assignments of each constraint. This explains that non-boolean CSPs are more powerful,

simply because the variables can take more values in satisfying assignments of the constraints.

2 Preliminaries

2.1 Arithmetic Circuit Complexity

We briefly recall the relevant definitions from arithmetic circuit complexity. A more thorough introduction into arithmetic circuit classes can be found in the book by Bürgisser [7]. Newer insights into the nature of VP and especially VP_{ws} are presented in the excellent paper of Malod and Portier [14].

An *arithmetic circuit* over a field $\mathbb{F}$ is a labeled directed acyclic graph (DAG) consisting of vertices or gates with indegree or fanin 0 or 2. The gates with fanin 0 are called input gates and are labeled with constants from $\mathbb{F}$ or variables $X_1, X_2, \ldots, X_n$. The gates with fanin 2 are called computation gates and are labeled with $\times$ or $+$.

The polynomial computed by an arithmetic circuit is defined in the obvious way: An input gate computes the value of its label, a computation gate computes the product or the sum of its childrens' values, respectively. We assume that a circuit has only one sink which we call output gate. We say that the polynomial computed by the circuit is the polynomial computed by the output gate. The *size* of an arithmetic circuit is the number of gates. The *depth* of a circuit is the length of the longest path from an input gate to the output gate in the circuit.

Sometimes we also consider circuits in which the $+$-gates may have unbounded fanin. We call these circuits *semi-unbounded circuits*. Observe that in semi-unbounded circuits $\times$-gates still have fanin 2. A circuit is called *multiplicatively disjoint* if for each $\times$-gate v the subcircuits that have the children of v as output-gates are disjoint. A circuit is called *skew*, if for all of its $\times$-gates one of the children is an input gate.

We call a sequence (f_n) of multivariate polynomials a family of polynomials or *polynomial family*. We say that a polynomial family is of polynomial degree, if there is a univariate polynomial p such that $\deg(f_n) \leq p(n)$ for each n. VP is the class of polynomial families of polynomial degree computed by families of polynomial size arithmetic circuits. VP_e is defined analogously with the circuits restricted to trees. By a classical result of Brent [4], VP_e equals the class of polynomial families computed by arithmetic circuits of depth $O(\log(n))$. VP_{ws} is the class of families of polynomials computed by families of skew circuits of polynomial size. Finally, a family (f_n) of polynomials is in VNP, if there is a family $(g_n) \in \mathsf{VP}$ and a polynomial p such that $f_n(X) = \sum_{e \in \{0,1\}^{p(n)}} g_n(e, X)$ for all n where X denotes the vector $(X_1, \ldots, X_{q(n)})$ for some polynomial q.

A polynomial f is called a *projection* of g (symbol: $f \leq g$), if there are values $a_i \in \mathbb{F} \cup \{X_1, X_2, \ldots\}$ such that $f(X) = g(a_1, \ldots, a_q)$. A family (f_n) of polynomials is a p-projection of (g_n) (symbol: $(f_n) \leq_p (g_n)$), if there is a polynomial r such that $f_n \leq g_{r(n)}$ for all n. As usual we say that (g_n) is hard for an arithmetic circuit class $\mathcal{C}$ if for every $(f_n) \in \mathcal{C}$ we have $(f_n) \leq_p (g_n)$. If further $(g_n) \in \mathcal{C}$ we say that (g_n) is $\mathcal{C}$-complete.

2.2 CSPs...

Let D and X be two sets. We denote with $D^X := \{a\colon X \to D\}$ the set of functions from X to D. A *constraint* is a function $\phi\colon D^X \to \{0,1\}$ where X and D are finite sets. We call D the domain and $var(\phi) = X$ the set of variables of ϕ. We call $k = |var(\phi)|$ the arity of the constraint ϕ. If $k = 2$ we also say that ϕ is binary. An *assignment* $a\colon var(\phi) \to D$ is said to satisfy ϕ, if and only if $\phi(a) = 1$. We say that ϕ is boolean, if $D = \{0,1\}$.

A *constraint satisfaction problem (CSP)* Φ of size m in the variables $var(\Phi)$ and domain D is a set of m constraints $\{\phi_1, \ldots, \phi_m\}$ such that the domain of all ϕ_i is D and $\bigcup_{i\in[m]} var(\phi_i) = var(\Phi)$. A CSP Φ is called binary iff all constraints ϕ_i of Φ are binary. If $D = \{0,1\}$ we call the CSP boolean.

A CSP Φ is *satisfied* by an assignment $a\colon var(\Phi) \to D$ if for all $i = 1, \ldots m$ we have $\phi_i(a|_{var(\phi_i)}) = 1$, where $a|_{var(\phi_i)} \in D^{var(\phi)}$ is the restriction of a onto $var(\phi)$. We also say that a satisfies the constraints of Φ.

When we have an order on the variables of the CSP we sometimes identify assignments $a\colon var(\Phi) \to D$ and vectors of length $var(\Phi)$ in the obvious way by giving a value table of a. We sometimes also describe constraints by describing its satisfying assignments as a set of vectors.

A CSP defines a function $\Phi^*\colon D^{var(\Phi)} \to \{0,1\}$ by setting $\Phi^*(a) = 1$ if and only if a satisfies Φ. In a slight abuse of notation we will not distinguish between the CSP Φ and the function Φ^* in this paper, but use the same symbol Φ for both of them. It will always be clear from the context which one of the two we mean.

Many well known decision problems can be formulated as CSPs. For example 2-CNF-formulas are constraint satisfaction problems in which all constraint are of the form $a \vee b$ where a and b are literals of the form x_i or $\neg x_i$ for a variable x_i. This illustrates the fact that in general a CSP has many more variables than each of its individual constraints.

We will in the following consider families (Φ_n) of CSPs. Every Φ_n may have its own universe D_n and its own set of variables $var(\Phi_n)$. But we will always assume that the arity of all constraints in all of the CSPs Φ_n is bounded by a constant k independent of n. We say that (Φ_n) has *bounded arity* in this case.

We call a family (Φ_n) of CSPs *p-bounded*, if and only if (Φ_n) has bounded arity and there is a polynomial p such that $|D_n| \leq p(n)$ and $|var(\Phi_n)| \leq p(n)$ for every n. We say that a constraint ϕ is *c-assignment bounded* if for all $x \in var(\phi)$ we have $|\{a(x) \mid a\colon var(\phi) \to D \text{ with } \phi(a) = 1\}| \leq c$, i.e., in the satisfying assignments of ϕ each variable x takes at most c values. We call a CSP c-assignment bounded if all of its constraints are c-assignment bounded. Observe that all boolean CSPs are trivially 2-assignment bounded. We can normalize CSPs with the following straightforward lemma:

Lemma 1. *Let (Φ_n) be p-bounded family of CSPs. Then there is a p-bounded family of CSPs (Φ'_n) that defines the same family of functions such that Φ'_n is of polynomial size in n.*

2.3 ... and Their Polynomials

To a CSP Φ we will assign two polynomials $P(\Phi)$ and $Q(\Phi)$. However, $P(\Phi)$ is only defined for boolean CSPs. So let Φ first be a boolean CSP with the set of variables $X = \{x_1, \dots, x_n\}$. We assign a polynomial $P(\Phi)$ in the (position) variables $Y_1, \dots, Y_n$ to Φ in the following way:

$$P(\Phi) := \sum_{e:\{x_1,\dots x_n\}\to\{0,1\}^n} \Phi(e) Y^e.$$

Here Y^e stands for $Y_1^{e(x_1)} Y_2^{e(x_2)} \dots Y_n^{e(x_n)}$.

Example 1. Let the constraints in Φ be $\{x_1 \vee x_2, x_3 \neq x_2, \neg x_4 \vee x_2\}$. The satisfying assignments are then 0100, 0101, 1010, 1100 and 1101. This results in $P(\Phi) = X_2 + X_2X_4 + X_1X_3 + X_1X_2 + X_1X_2X_4$.

In contrast to $P(\Phi)$ the second polynomial $Q(\Phi)$ is also defined for non-boolean CSPs. So let Φ be a CSP with domain D. We assign to Φ the following polynomial $Q(\Phi)$ in the variables $\{X_d \mid d \in D\}$.

$$Q(\Phi) := \sum_{a:var(\Phi)\to D} \Phi(a) \prod_{x\in var(\Phi)} X_{a(x)} = \sum_{a:var(\Phi)\to D} \Phi(a) \prod_{d\in D} X_d^{\mu_d(a)},$$

where $\mu_d(a) = |\{x \in var(\Phi) \mid a(x) = d\}|$ computes number of variables mapped to d by a. Note that $Q(\Phi)$ is homogeneous of degree $|var(\Phi)|$.

Example 2. Let $D = \{1, 2, 3, 4\}$ and let the constraints in Φ be $\{x_1 + x_2 \geq 4, x_3 = 5 - x_2, x_1 < x_2\}$. The satisfying assignments are then $(1, 3, 2)$, $(2, 3, 2)$, $(1, 4, 1)$, $(2, 4, 1)$ and $(3, 4, 1)$. This results in $Q(\Phi) = X_1X_2X_3 + X_2^2X_3 + X_1^2X_4 + X_1X_2X_4 + X_1X_3X_4$.

Remark 1. The polynomial Q has a very natural algebraic interpretation: Consider the free monoid D^* consisting of finite words of the symbols in D. Furthermore consider the free commutative monoid X_D^c on the symbols $X_D := \{X_d \mid d \in D\}$ which is essentially the set of monomials in the variables in X_D. There is a natural monoid morphism $q : D^* \to X_D^c$ with $q(a_1 \dots a_s) = \prod_{i=1}^s X_{a_i}$. The morphism q drops the order of the symbols in a word and computes a commutative version of it.

Now we consider two rings: The first one is $\mathbb{Z}[D^*]$ consisting of formal integer linear combinations of words in D^*. Observe that we can think of any finite set $S \in D^*$ as an element of $\mathbb{Z}[D^*]$ by encoding it as $\sum_{a\in S} a$. The second ring we consider is $\mathbb{Z}[X_D]$ which is simply the polynomial ring over $\mathbb{Z}$ in the variables X_D. The monoid morphism q induces the ring morphism $Q : \mathbb{Z}[D^*] \to \mathbb{Z}[X_D]$ by $Q(\sum_a c_a a) = \sum_a c_a q(a)$. Given the encoding $\sum_{a\in S} a$ of a set S, Q computes a commutative version of it. This is exactly what the polynomial $Q(\Phi)$ defined above does: To a CSP Φ it computes a commutative version of the set of satisfying assignments.

In general the $P(\Phi)$ and $Q(\Phi)$ are expressive enough to characterize VNP. Thus in order to characterize subclasses of VNP we introduce structural restrictions to CSPs in the next section.

2.4 Treewidth

An excellent introduction to treewidth, its properties and algorithmic consequences can be found in [10, Chapter 11]. For the convenience of the reader we recall the definitions and facts needed in the remainder of this paper.

A *tree decomposition* of a graph $G = (V, E)$ is a pair $(\mathcal{T}, (B_t)_{t\in T})$, where $\mathcal{T} = (T, F)$ is a tree and $(B_t)_{t\in T}$ is a family of subsets of V such that:

- $\bigcup_{t\in T} B_t = V$.
- For every $v \in V$, the set $B^{-1}(v) := \{t \in T \mid v \in B_t\}$ is nonempty and connected in $\mathcal{T}$.
- For every edge $uv \in E$ there is a $t \in T$ such that $u, v \in B_t$.

The sets B_t are called the *bags* of the decomposition. The *width* of the decomposition is $\max\{|B_t| \mid t \in T\} - 1$. The *treewidth* $\mathrm{tw}(G)$ of a graph G is defined as the minimum of the widths of all tree-decompositions of G. With this definition trees have a treewidth of 1.

A *path decomposition* is a tree decomposition in which $\mathcal{T}$ is a path. The *pathwidth* $\mathrm{pw}(G)$ is defined in a completely analogous fashion to $\mathrm{tw}(G)$. Clearly for all graphs G we have $\mathrm{tw}(G) \leq \mathrm{pw}(G)$.

We state a well known property of tree-decompositions (see [10, Chapter 11]).

Proposition 1. *Let $G = (V, E)$ be a graph and $(\mathcal{T}, (B_t)_{t\in T})$ be a tree-decomposition of G. Then for each clique $C \subseteq V$ in G there is a bag B_t such that $C \subseteq B_t$.*

To a CSP Φ we assign two graphs: The *primal graph* G^P_Φ has the vertex set $var(\Phi)$ and there is an edge between two vertices x and y if and only if there is a constraint ϕ in Φ such that $\{x, y\} \subseteq var(\phi)$. Note that the constraints in Φ yield cliques in G^P_Φ. The *incidence graph* G^I_Φ has the vertex set $var(\Phi) \cup \{\phi \mid \phi$ constraint in $\Phi\}$. There is an edge between $x \in var(\Phi)$ and ϕ if and only if $x \in var(\phi)$. There are no other edges in G^I_Φ, thus G^I_Φ is bipartite.

We define the *treewidth* of a CSP G as $\mathrm{tw}(\Phi) := \mathrm{tw}(G^P_\Phi)$. We say that a family of CSPs (Φ_n) has bounded treewidth, if and only if $\mathrm{tw}(G_{\Phi_n}) \leq d$ for a constant d independent of n. We could also have defined the treewidth of Φ as the treewidth of the incidence graph G^I_Φ, but the following folklore lemma tells us that there is not much difference if we consider CSPs with bounded arity.

Lemma 2. *For every CSP Φ we have:*

a) $tw(G^I_\Phi) \leq tw(G^P_\Phi) + 1$.
b) If all constraints in Φ have arity at most k, then $tw(G^P_\Phi) \leq k(tw(G^I_\Phi)+1)-1$.

The following lemma relates the expressivity of P and Q.

Lemma 3. *For every boolean CSP Φ in s variables there is a 2-assignment bounded CSP Ψ with domain size $|D| = 2s$ such that $P(\Phi) \leq Q(\Psi)$ and $G^P_\Phi \simeq G^P_\Psi$.*

3 Statement of the Results

Having introduced all necessary definitions we will now formulate our results in this section. Our first theorem characterizes the expressive power of boolean and non-boolean CSPs of bounded path- and treewidth.

Theorem 1 (Characterization of VP_e)

a) Let (Φ_n) be a p-bounded family of boolean CSPs with bounded treewidth. Then $(P(\Phi_n)) \in \mathsf{VP}_e$. Moreover, any family in VP_e is a p-projection of such a $(P(\Phi_n))$. The same statement also holds with pathwidth instead of treewidth.

b) Let (Φ_n) be a p-bounded family of c-assignment bounded CSPs with bounded treewidth. Then $(Q(\Phi_n)) \in \mathsf{VP}_e$. Moreover, any family in VP_e is a p-projection of such $(Q(\Phi_n))$. The same statement also holds with pathwidth instead of treewidth.

Observe that Theorem 1 implies that bounded pathwidth and bounded treewidth have the same computational power in this setting, although pathwidth is a far more restrictive measure.

Our next Theorem shows that general non-boolean CSPs with bounded *treewidth* characterize VP.

Theorem 2 (Characterization of VP) *Let (Φ_n) be a p-bounded family of CSPs with bounded treewidth. Then $(Q(\Phi_n)) \in \mathsf{VP}$. Moreover, any family in VP is a p-projection of such a $(Q(\Phi_n))$.*

Finally we show that for general non-boolean CSPs pathwidth and treewidth differ in expressivity. With bounded *pathwidth* we get a characterization of VP_{ws}.

Theorem 3 (Characterization of VP_{ws}) *Let (Φ_n) be a p-bounded family of CSPs with bounded pathwidth. Then $(Q(\Phi_n)) \in \mathsf{VP}_{ws}$. Moreover, any family in VP_{ws} is a p-projection of such a $(Q(\Phi_n))$.*

Observe that the only difference between Theorem 1 b) and Theorem 2/3 is the c-assigment boundedness. This means that the difference between VP_e and $\mathsf{VP}/\mathsf{VP}_{ws}$ in this setting is simply that for VP and VP_{ws} the variables in the constraints may take more different values in satisfying assignments.

We will prove Theorem 1, Theorem 2 and Theorem 3 in several individual lemmas. Because of space restrictions most of the proofs are omitted. They can be found in the upcoming full version of this paper.

We now relate our results to known results. Fischer, Makowsky and Ravve [9] consider the problem of counting solutions to boolean CSPs and achieve the following results:

Theorem 4 ([9])

a) There is an algorithm that given a CNF-Formula Φ of size n and a tree decomposition of G^I_Φ of width k counts the number of satisfying assignments of Φ using at most $4^k n$ operations.

b) Given a boolean CSP Φ of size n and a tree decomposition of G^P_Φ of width k, the number of satisfying assignments of Φ can be computed with $4^k n^2$ arithmetic operations.

Observe that CNF formulas are special forms of CSPs in which the constraints are disjunctive clauses. For CNF-formulas the size of the clauses need not have bounded arity to guarantee feasibility in part a) of Theorem 4. In b) there is an implicit bound on the arity of the constraints, because the treewidth of the primal graph is bounded, so the setting is more comparable to ours. Thus Theorem 2 can be seen as an extension of b) to non-boolean CSPs also adding a matching lower bound.

Briquel, Koiran and Meer [12,6] give the following result.

Theorem 5 ([12,6]). *For every family (Φ_n) of p-bounded CNF-formulas with bounded treewidth of $G^I_{\Phi_n}$ we have $(P(\Phi_n)) \in \mathsf{VP}_e$. Moreover, any family in VP_e is a p-projection of such a $(P(\Phi_n))$.*

Again the size of the CNF-clauses is not restricted and the treewidth of the incidence graph is considered. Theorem 5 can be interpreted as translation of Theorem 4 a) into the arithmetic circuit model with a matching hardness result. Theorem 1 is an extension of Theorem 5 to general CSPs instead of CNF-formulas. Moreover, the lower bound is shown to already hold for bounded pathwidth. But in contrast to Briquel, Koiran and Meer we require a bound on the arity of the constraints to show feasibility in our setting.

4 Lower Bounds

In this section we show the lower bounds on the expressivity of polynomials defined by CSPs.

Lemma 4. *There is a constant $c \leq 26$ such that the following holds: For every $(f_n) \in \mathsf{VP}_e$ there is a p-bounded family (Φ_n) of boolean CSPs with pathwidth at most c such that $(f_n) \leq_p (P(\Phi_n))$.*

A proof of Lemma follows by using an encoding of iterated 3×3-matrix multiplication (see [2]) into CSPs. Combining Lemma 4 and Lemma 3 directly yields the following corollary.

Corollary 1. *There is a constant $c \leq 26$ such that the following holds: For every $(f_n) \in \mathsf{VP}_e$ there is a p-bounded family (Ψ_n) of 2-assignment bounded CSPs with pathwidth at most c such that $(f_n) \leq_p (Q(\Psi_n))$.*

Next we will show the lower bounds for the characterizations of VP and VP_{ws}. For the proofs we use so called parse tree arguments (see e.g. [14, Section 4]). A parse tree T of a multiplicatively disjoint circuit C is a subgraph of C that is constructed in the following way:

– Add the output gate of C to T.
– For every gate v added to T do the following;
  • If v is a $+$-gate, add exactly one of its children to T.
  • If v is a $\times$-gate, add both of its children to T.

Observe that parse trees are binary trees. The weight $w(T)$ of a parse tree T is the product of the labels of its leaves. The polynomial computed by C is the sum of the weights of all of C's parse trees.

Lemma 5. *Let $(f_n) \in \mathsf{VP}$, then there is a p-bounded family Φ_n of binary CSPs such that $(f_n) \leq_p (Q(\Phi_n))$ and $G^P_{\Phi_n}$ is a tree for every n.*

In the proof we will use the following result.

Lemma 6. *$(f_n) \in \mathsf{VP}$ if and only if (f_n) is computed by a family of multiplicatively disjoint semi-unbounded logarithmic depth circuits.*

The proof of Lemma 6 easily follows by applying the techniques of Malod and Portier [14, Lemma 2] on the classical characterization of VP by logarithmic depth semi-unbounded arithmetic circuits found by Valiant et al. [16].

Proof (of Lemma 5). The idea of the proof is the following: We use the characterization in Lemma 6 which yields that the polynomials f_n have logarithmic depth parse trees. We encode these parse trees into polynomial size CSPs whose primal graphs are trees isomorphic to the parse trees of the f_n. Summing up over all possible encodings of parse trees we get polynomials whose projection are the f_n. We now describe the construction in more detail.

So consider a polynomial $f = f_n$ from our family. By Lemma 6 we know that f_n is computed by a logarithmic depth semi-unbounded circuit C of polynomial size. By adding dummies we can make sure that C has the following "layered" form:

– All operation gates at the same depth have the same operation.
– All leaves are at the same depth level.

This implies that all parse trees of C are isomorphic binary trees. Let the children of the $\times$-gates in Φ be ordered, i.e., we call one of them the left child and the other one the right child. Let T be a binary tree isomorphic to the parse trees of C. The children of vertices in T that correspond to $\times$-gates in C are also ordered.

We now build a CSP Φ with $var(\Phi) = V(T)$ and $G^P_\Phi = T$. The domain is the vertex set $V(C)$ of C. To distinguish the vertices of T from the gates of C we write the vertices of T with a hat, e.g. $\hat{v} \in V(T)$. For each edge $\hat{u}\hat{v}$ in T we define a constraint $\phi_{\hat{u}\hat{v}}$ on the variables $\hat{u}$ and $\hat{v}$ in the following way: If $\hat{u}$ corresponds to a $+$-gate in C, then the satisfying assignments of $\phi_{\hat{u}\hat{v}}$ (where u and v denote the images of $\hat{u}$ and $\hat{v}$, respectively) are described by

$$\{(u, v) \mid u, v \in V(C), u \text{ is a } +\text{-gate}, v \text{ is a child of } u\}.$$

If $\hat{u}$ corresponds to a $\times$-gate and $\hat{v}$ is the left child of $\hat{u}$, then $\phi_{\hat{u}\hat{v}}$ is described by

$$\{(u, v) \mid u, v \in V(C), u \text{ is a } \times\text{-gate}, v \text{ is the left child of } u\}.$$

For right children we add constraints in an analog fashion.

It is easy to see, that Φ is satisfied by an assignment $a : V(T) \to V(C)$ if and only if a maps T onto a parse tree T_a of C. Also for each satisfying assignment a the resulting monomial $\prod_{\hat{u} \in V(T)} X_{a(\hat{u})}$ can be projected to $w(T_a)$ by doing the following: If v is an operation gate of C, then substitute X_v by 1. If v is an input gate of C with label l, then substitute X_v by l. Because each v is either an operation gate or an input gate but never both, these settings do not contradict for different satisfying assignments of Φ. It follows that $f \leq Q(\Phi)$. The primal graph G^P_Φ of Φ is by construction the tree T. The observation that the size of Φ and its domain $V(C)$ are polynomial completes the proof. □

We use a similar parse tree argument for VP_{ws}.

Lemma 7. *Let $(f_n) \in \mathsf{VP}_{ws}$, then there is a p-bounded family Φ_n of binary CSPs such that $(f_n) \leq_p (Q(\Phi_n))$. Furthermore $pw(\Phi_n) = 1$ for every n.*

The key insight for the proof for Lemma 7 is that parse trees of skew circuits have a very restricted form that allows encoding them into CSPs of bounded pathwidth.

5 Upper Bounds on the Complexity

Lemma 8. *For every family (Φ_n) of p-bounded and c-assignment bounded CSPs of bounded treewidth we have $(Q(\Phi_n)) \in \mathsf{VP}_e$.*

Proof. Consider a family (Φ_n) of CSPs with the desired properties. To ease notation we fix n and set $\Phi = \Phi_n$ and $D = D_n$.

Fix a tree-decomposition $(\mathcal{T}, (B_t)_{t \in T})$ of the primal graph G^P_Φ with minimal width. W.l.o.g. the tree $\mathcal{T} = (T, F)$ is a rooted, binary tree and has depth $O(\log(n))$ (see [3]). We give $\mathcal{T}$ a direction from the leaves to the root and will later make an induction along this direction. We use teh following helpful claim:

Claim 6. *We may assume that there is a bijection from the vertices in T to the constraints in Φ such that $t \in T$ is mapped to a constraint ϕ_t with $var(\phi_t) = B_t$.*

It follows that $|B_t| \leq k$ for every $t \in T$, where k is the upper bound on arity of the constraints in Φ.

For each vertex t of $\mathcal{T}$ let $\mathcal{T}_t$ be the subtree of $\mathcal{T}$ that has t as its root. Let $T_t = V(\mathcal{T}_t)$ be the vertex set of $\mathcal{T}_t$. Furthermore let Φ_t be the CSP with the set of constraints $\{\phi_{t'} \mid t' \in T_t\}$ and the set of variables $var(\Phi_t) = \bigcup_{t' \in T_t} B_{t'}$.

We say that an assignment $a\colon B_t \to D$ is *good* for t or ϕ_t if it satisfies ϕ_t. Similarly we call a partial assignment to B_t good for t or ϕ_t if it can be extended to a good assignment. We are only interested in good assignments for individual constraints ϕ_t, because bad assignments do not contribute to $Q(\Phi)$ anyway.

Let $a\colon V \to D$ and $b\colon W \to D$ be assignments. We say that a and b are consistent (symbol: $a \sim b$), if $a|_{V\cap W} = b|_{V\cap W}$, i.e. they assign the same values to variables they share.

For each vertex $t \in T$ we will compute polynomials

$$f_{t,a,e} := \sum_{\substack{\alpha\colon var(\Phi_t)\to D,\\ a\sim\alpha}} \Phi_t(\alpha) \prod_{x\in var(\Phi_t)\setminus e} X_{\alpha(x)},$$

where a is a good assignment for ϕ_t and $e \subseteq B_t$. The sets $e \subseteq B_t$ will later in the construction prevent that variables $X_{a(x)}$ appear more than once in a monomial for $x \in var(\Phi)$.

Observe that for each t there are only $O(1)$ polynomials $f_{t,a,e}$: Because Φ is c-assignment bounded and its constraints ϕ have at most arity k, each ϕ has at most c^k satisfying assignments. Also there are at most $2^{|B_t|} \leq 2^k$ sets e. It follows that there are at most $c^k 2^k = O(1)$ polynomials $f_{t,a,e}$ for each t.

The *depth* of a vertex $t \in T$, denoted by $depth(t)$, is the length of the longest path from a leaf to t in $\mathcal{T}_t$.

Claim 7. *For each $t \in T$ we can compute all $f_{t,a,e}$ with a circuit of depth $O(depth(t))$.*

Lemma 8 follows easily with Claim 7: Let t^* be the root of $\mathcal{T}$. By definition

$$\begin{aligned} Q(\Phi) &= \sum_{\alpha\colon var(\Phi)\to D} \Phi(\alpha) \prod_{x\in var(\Phi)} X_{\alpha(x)} \\ &= \sum_{a\colon B_{t^*}\to D} \sum_{\alpha\colon var(\Phi)\to D, a\sim\alpha} \Phi(\alpha) \prod_{x\in var(\Phi)} X_{\alpha(x)} \\ &= \sum_{a\colon B_{t^*}\to D, \Phi_{t^*}(a)=1} f_{t^*,a,\emptyset} \end{aligned}$$

The tree $\mathcal{T}$ has depth $O(\log(n))$, so with Claim 7 we can compute $Q(\Phi)$ with a circuit of depth $O(\log(n))$. It follows that $(\Phi_n) \in \mathsf{VP}_e$. Thus all that is left to do is to prove Claim 7. □

Corollary 2. *For every family (Φ_n) of boolean p-bounded CSPs of bounded treewidth we have $(P(\Phi_n)) \in \mathsf{VP}_e$.*

The proof of Lemma 8 can be adapted to show the following lemmas:

Lemma 9. *For every family (Φ_n) of p-bounded CSPs of bounded treewidth we have $(Q(\Phi_n)) \in \mathsf{VP}$.*

Lemma 10. *For every family (Φ_n) of p-bounded CSPs of bounded pathwidth we have $(Q(\Phi_n)) \in \mathsf{VP}_{ws}$.*

Acknowledgements. I am very grateful to my supervisor Peter Bürgisser for many helpful discussions and his support in making the presentation of this paper much clearer. I would also like to thank the organizers of the Dagstuhl Seminar 10481 "Computational Counting" where some of the results in this paper were conceived.

References

1. Beeri, C., Fagin, R., Maier, D., Yannakakis, M.: On the desirability of acyclic database schemes. J. ACM 30(3), 479–513 (1983)
2. Ben-Or, M., Cleve, R.: Computing algebraic formulas using a constant number of registers. SIAM J. Comput. 21(1), 54–58 (1992)
3. Bodlaender, H.L.: NC-algorithms for graphs with small treewidth. In: van Leeuwen, J. (ed.) WG 1988. LNCS, vol. 344, pp. 1–10. Springer, Heidelberg (1989)
4. Brent, R.P.: The complexity of multiple-precision arithmetic. In: Brent, R.P., Andersson, R.S. (eds.) The Complexity of Computational Problem Solving, pp. 126–165. Univ. of Queensland Press (1976)
5. Briquel, I., Koiran, P.: A dichotomy theorem for polynomial evaluation. In: Královič, R., Niwiński, D. (eds.) MFCS 2009. LNCS, vol. 5734, pp. 187–198. Springer, Heidelberg (2009)
6. Briquel, I., Koiran, P., Meer, K.: On the expressive power of cnf formulas of bounded tree- and clique-width. Discrete Applied Mathematics 159(1), 1–14 (2011)
7. Bürgisser, P.: Completeness and reduction in algebraic complexity theory. Springer, Heidelberg (2000)
8. Courcelle, B., Makowsky, J.A., Rotics, U.: On the fixed parameter complexity of graph enumeration problems definable in monadic second-order logic. Discrete Applied Mathematics 108(1-2), 23–52 (2001)
9. Fischer, E., Makowsky, J.A., Ravve, E.V.: Counting truth assignments of formulas of bounded tree-width or clique-width. Discrete Applied Mathematics 156(4), 511–529 (2008)
10. Flum, J., Grohe, M.: Parameterized complexity theory. Springer-Verlag New York Inc., New York (2006)
11. Gottlob, G., Leone, N., Scarcello, F.: The complexity of acyclic conjunctive queries. J. ACM 48(3), 431–498 (2001)
12. Koiran, P., Meer, K.: On the expressive power of CNF formulas of bounded tree- and clique-width. In: Broersma, H., Erlebach, T., Friedetzky, T., Paulusma, D. (eds.) WG 2008. LNCS, vol. 5344, pp. 252–263. Springer, Heidelberg (2008)
13. Flarup, U., Koiran, P., Lyaudet, L.: On the Expressive Power of Planar Perfect Matching and Permanents of Bounded Treewidth Matrices. In: Tokuyama, T. (ed.) ISAAC 2007. LNCS, vol. 4835, pp. 124–136. Springer, Heidelberg (2007), http://dx.doi.org/10.1007/978-3-540-77120-3_13
14. Malod, G., Portier, N.: Characterizing Valiant's algebraic complexity classes. J. Complexity 24(1), 16–38 (2008)
15. Schaefer, T.J.: The complexity of satisfiability problems. In: STOC 1978, pp. 216–226 (1978)
16. Valiant, L.G., Skyum, S., Berkowitz, S., Rackoff, C.: Fast parallel computation of polynomials using few processors. SIAM J. Comput. 12(4), 641–644 (1983)

The Complexity of Symmetric Boolean Parity Holant Problems
(Extended Abstract)

Heng Guo[1], Pinyan Lu[2], and Leslie G. Valiant[3,⋆]

[1] University of Wisconsin-Madison
hguo@cs.wisc.edu
[2] Microsoft Research Asia
pinyanl@microsoft.com
[3] Harvard University
valiant@seas.harvard.edu

Abstract. For certain subclasses of NP, ⊕P or #P characterized by local constraints, it is known that if there exist any problems that are not polynomial time computable within that subclass, then those problems are NP-, ⊕P- or #P-complete. Such dichotomy results have been proved for characterizations such as Constraint Satisfaction Problems, and directed and undirected Graph Homomorphism Problems, often with additional restrictions. Here we give a dichotomy result for the more expressive framework of Holant Problems. These additionally allow for the expression of matching problems, which have had pivotal roles in complexity theory. As our main result we prove the dichotomy theorem that, for the class ⊕P, every set of boolean symmetric Holant signatures of any arities that is not polynomial time computable is ⊕P-complete. The result exploits some special properties of the class ⊕P and characterizes four distinct tractable subclasses within ⊕P. It leaves open the corresponding questions for NP, #P and $\#_k$P for $k \neq 2$.

1 Introduction

The complexity class ⊕P is the class of languages L such that there is a polynomial time nondeterministic Turing machine that on input $x \in L$ has an odd number of accepting computations, and on input $x \notin L$ has an even number of accepting computations [29,25]. It is known that ⊕P is at least as powerful as NP, since NP is reducible to ⊕P via (one-sided) randomized reduction [28]. Also, the polynomial hierarchy is reducible to ⊕P via two sided randomized reduction [27]. There exist decision problems, such as graph isomorphism, that are not known to be in P but are known to be in ⊕P [1]. The class ⊕P has also been related to other complexity classes via relativization [2]. Further, while the class ⊕P lies between NP and #P, it is known that there are several natural

⋆ This research was supported in part by grants NSF-CCF-04-27129 and NSF-CCF-09-64401.

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 712–723, 2011.

problems such as 2SAT that are $\oplus$P-complete where the corresponding existence problem is in P [31], and a range of others, including graph matchings and some coloring problems, for which the parity problem is in P but exact counting is #P-complete [33].

As with the classes NP and #P it is an open question whether $\oplus$P strictly extends P. For certain restrictions of these classes, however, dichotomy theorems are known. For NP a dichotomy theorem would state that any problem in the restricted subclass of NP is either in P or is NP-complete (or both, in the eventuality that NP equals P.) Ladner [24] proved that without any restrictions this situation does not hold: if P $\neq$ NP then there is an infinite hierarchy of intermediate problems that are not polynomial time interreducible.

The restrictions for which dichotomy theorems are known can be framed in terms of local constraints, most importantly, Constraint Satisfaction Problems (CSP) [26,15,3,4,17,20,14,19], and Graph Homomorphism Problems [18,21,5]. Explicit dichotomy results, where available, manifest a total understanding of the class of computations in question, to within polynomial time reduction, and modulo the collapse of the class.

In this paper we consider dichotomies in a framework for characterizing local properties that is more general than those mentioned in the previous paragraph, and is called the Holant framework [9,11]. A particular problem in this framework is characterized by a set of signatures as defined in the theory of Holographic Algorithms [32,31]. The CSP framework can be viewed as the special case of the Holant framework in which equality relations of any arity are always available [11]. The addition of equality relations in CSP makes many sets of constraints complete that are not otherwise.

A brief description of the Holant framework is as follows. A *signature grid* $\Omega = (G, \mathcal{F}, \pi)$ is a triple, where $G = (V, E)$ is an undirected graph, $\mathcal{F}$ is a set of functions on variables from a domain D, and π labels each $v \in V$ with a function $f_v \in \mathcal{F}$. An assignment σ maps each edge $e \in E$ to an element of D and determines a value $\prod_{v \in V} f_v(\sigma \mid_{E(v)})$, where $E(v)$ denotes the incident edges of v, and $\sigma \mid_{E(v)}$ denotes the restriction of σ to $E(v)$. The counting problem on the instance Ω is the problem of computing the following sum over all possible assignments σ

$$\text{Holant}_\Omega = \sum_\sigma \prod_{v \in V} f_v(\sigma \mid_{E(v)}).$$

For example, consider the Perfect Matching problem on G. This corresponds to $D = \{0, 1\}$ and f_v the Exactly-One function at every vertex of G. Then σ corresponds to a subset of the edges, and Holant_Ω counts the number of perfect matchings in G. If we use the At-Most-One function at every vertex, then we count all (not necessarily perfect) matchings. We use the notation Holant($\mathcal{F}$) to denote the class of Holant problems where the functions f_v are chosen from the set $\mathcal{F}$. If all functions take integer values and we only need to compute the parity of the Holant value, it is called a *parity Holant problem*, and is denoted by $\oplus$Holant($\mathcal{F}$).

In this paper we consider symmetric boolean parity Holant problems, that is, in the definition of Holant$_\Omega$, $D = \{0, 1\}$, $\mathcal{F}$ is a set of symmetric functions with variables in D and range in D, and summation is modulo two. Out main theorem is a dichotomy regarding the class $\oplus$P.

Theorem 1. *Let $\mathcal{F}$ be a set of symmetric signatures. The parity problem* $\oplus$Holant $(\mathcal{F})$ *is either computable in polynomial time, or $\oplus$P-complete.*

This is the first such dichotomy for the Holant family. No dichotomy theorem is known for comparable restrictions of #P, NP or #$_k$P for $k \neq 2$. For #P dichotomy results are known only for Holant^c problems, where Holant^c denotes that the unary constant signatures 0 and 1 are assumed to be available. The known results for Holant^c are for the symmetric case over the real numbers [11], and over the complex numbers [6], and for planar graphs in the former case [12]. For NP, Cook and Bruck [13] gave a dichotomy theorem for singleton sets of constraints of arity up to three in the general nonsymmetric case.

Our main dichotomy theorem exhibits four classes of signature sets that are polynomial time computable. The first class is composed by affine signatures, for which the Holant problem is solvable by Gaussian elimination. The second corresponds to signature sets that include perfect and partial matching gates. The third corresponds to Fibonacci signatures with the addition of the binary negation signature [0, 1, 0]. The fourth is what we call *vanishing signature sets*, which always give zero solutions modulo two. We do not have an explicit characterization of this fourth class. We show that any set of symmetric signatures that is not a subset of one of these four classes is $\oplus$P-complete.

Similar results have been obtained for the #CSP problem modulo k. In Faben's dichotomy theorem for boolean #CSP modulo k [19], the affine ones form the only positive class for general k, and for our case of $k = 2$ there is the second class of those that vanish for the simple reason that they are closed under complement. In the dichotomy of weighted boolean #CSP modulo k [22], the tractable classes have no immediate counterpart as it is of no meaning to discuss weights here in the parity setting.

Along the way to proving our main result we prove dichotomy theorems for both the planar and general case of $\oplus$Holant^c, that is for signature sets including both of the unary constants 0 and 1. We also prove a dichotomy theorem for 2-3 regular bipartite graphs with singleton signature sets, which is the simplest nontrivial setting, and previously investigated in the Holant framework [9,10,23,6] for #P.

Finding analogs of our main result for NP, #P or #$_k$P for $k \neq 2$ remain challenges for the future, as is also the same question for $\oplus$P for nonsymmetric signatures.

2 Preliminaries

The framework of Holant problems for #P is usually defined for functions mapping any $[q]^k \to \mathbb{C}$ for a finite q. Our results in this paper for $\oplus$P are for the

Boolean case $q = 2$ of functions $[2]^k \to \{0, 1\}$. We shall therefore assume throughout that $q = 2$.

A *signature grid* $\Omega = (H, \mathcal{F}, \pi)$ consists of a graph $H = (V, E)$ with each vertex labeled by a function $f_v \in \mathcal{F}$, where π is the labelling. The Holant problem on instance Ω is that of evaluating $\text{Holant}_\Omega = \sum_\sigma \prod_{v \in V} f_v(\sigma \mid_{E(v)})$, a sum over all edge assignments $\sigma : E \to \{0, 1\}$. A function f_v can be represented as a truth table, or as a tensor in $(\mathbb{C}^2)^{\otimes \deg(v)}$. We also use f^α to denote the value $f(\alpha)$, where α is a $\{0, 1\}$ string. A function $f \in \mathcal{F}$ is also called a *signature*. A symmetric function f on k Boolean variables can be expressed as $[f_0, f_1, \ldots, f_k]$, where f_i is the value of f on inputs of Hamming weight i. For any $0 \le l < h \le k$, we call $[f_l, f_{l+1}, \ldots, f_h]$ a *subsignature* of $[f_0, f_1, \ldots, f_k]$. Note that with the help of the two unary signatures $[0, 1]$ and $[1, 0]$, any subsignature of a given signature is realizable.

Definition 1. *A signature is degenerate iff it is a tensor product of unary signatures.*

A Holant problem is parameterized by a set of signatures.

Definition 2. *Given a set of signatures $\mathcal{F}$, we define the following counting problem as* $\text{Holant}(\mathcal{F})$*:*

Input: A signature grid $\Omega = (G, \mathcal{F}, \pi)$;

Output: Holant_Ω*.*

The following family Holant^c of Holant problems is important [11,6,12]. This is the class of all Holant Problems (on boolean variables) where one can set any particular edge (variable) to 0 or 1 in an input to the graph, or in other words, where the unary constant functions 0 and 1 are always available for use.

Definition 3. *Given a set of signatures $\mathcal{F}$,* $\text{Holant}^c(\mathcal{F})$ *denotes* $\text{Holant}(\mathcal{F} \cup \{[1, 0], [0, 1]\})$.

In this paper, we consider the parity version of Holant problems $\oplus\text{Holant}(\mathcal{F})$, where each signature in $\mathcal{F}$ takes values from $\mathbb{Z}_2 = \{0, 1\}$. We also define $\oplus\text{Holant}^c$ problems analogously. Planar (parity) Holant problems are (parity) Holant problems on planar graphs.

To introduce the idea of holographic reductions, it is convenient first to consider bipartite graphs. We note that for any general graph we can make it bipartite by adding an additional vertex on each edge, and giving each new vertex the EQUALITY function $=_2$ on 2 inputs.

We use $\text{Holant}(\mathscr{G}|\mathscr{R})$ to denote all counting problems, expressed as Holant problems on bipartite graphs $H = (U, V, E)$, where each signature for a vertex in U or V is from $\mathscr{G}$ or $\mathscr{R}$, respectively. An input instance for the bipartite Holant problem is a bipartite signature grid and is denoted as $\Omega = (H, \mathscr{G}|\mathscr{R}, \pi)$. Signatures in $\mathscr{G}$ are denoted by column vectors (or contravariant tensors); signatures in $\mathscr{R}$ are denoted by row vectors (or covariant tensors) [16].

One can perform (contravariant and covariant) tensor transformations on the signatures. We define a simple version of holographic reductions that are invertible. Suppose $\text{Holant}(\mathscr{G}|\mathscr{R})$ and $\text{Holant}(\mathscr{G}'|\mathscr{R}')$ are two Holant problems defined

for the same family of graphs, and $T \in \mathbf{GL}_2(\mathbb{C})$ is a basis transformation. We say that there is an (invertible) holographic reduction from Holant($\mathscr{G}|\mathscr{R}$) to Holant($\mathscr{G}'|\mathscr{R}'$), if the *contravariant* transformation $G' = T^{\otimes g}G$ and the *covariant* transformation $R = R'T^{\otimes r}$ map $G \in \mathscr{G}$ to $G' \in \mathscr{G}'$ and $R \in \mathscr{R}$ to $R' \in \mathscr{R}'$, and vice versa, where G and R have arity g and r respectively. (Notice the reversal of directions when the transformation $T^{\otimes n}$ is applied. This is the meaning of *contravariance* and *covariance*.) Suppose there is a holographic reduction from $\#\mathscr{G}|\mathscr{R}$ to $\#\mathscr{G}'|\mathscr{R}'$ mapping signature grid Ω to Ω', then $\text{Holant}_{\Omega} = \text{Holant}_{\Omega'}$.

In particular, for invertible holographic reductions from Holant($\mathscr{G}|\mathscr{R}$) to Holant($\mathscr{G}'|\mathscr{R}'$), one problem is in P iff the other one is in P, and similarly one problem is #P-hard ($\oplus$P-hard) iff the other one is also #P-hard ($\oplus$P-hard).

In the study of Holant problems, we will often move between bipartite and non-bipartite settings. When this does not cause confusion, we do not distinguish between signatures that are column vectors (or contravariant tensors) and row vectors (or covariant tensors). Whenever we write a transformation as $T^{\otimes n}F$ or $T\mathscr{F}$, we view the signatures as column vectors (or contravariant tensors); whenever we write a transformation as $FT^{\otimes n}$ or $\mathscr{F}T$, we view the signatures as row vectors (or covariant tensors).

All signatures we consider are in the boolean domain. If we flip the 0 and 1 in the domain, a symmetric signature will be changed into its reverse, and the Holant values are the same. That is, the complexity of Holant problems for a set of signatures is the same as the complexity of Holant problems for the set composed by those signatures reversed. In this paper this operation will be performed repeatedly.

3 Tractable Families

We shall identify three tractable families for $\oplus\text{Holant}^c$ problems. The first family, *Affine Signatures*, is adopted directly from the corresponding family for #CSP, where it is the sole tractable class [15,14]. The second family we derive from the *Fibonacci Signatures*. For general counting problems, we also have a tractable family of Fibonacci signatures, but for parity problems, as we shall show, the family remains tractable even with the addition of the inversion signature $[0, 1, 0]$. This addition for general counting problems would give rise to #P-hardness. The third tractable family, *Matchgate Signatures*, is special to parity problems.

3.1 Affine Signatures

Definition 4. *A signature is affine iff its support is an affine space. We denote the set of all affine signatures by* $\mathscr{A}$.

By definition, an affine signature can be viewed as a constraint defined by a set of linear equations. Viewing the edges as variables in $\mathbb{Z}_2$, every assignment which contributes 1 in the summation corresponds to a solution which satisfies all the linear equations. Then the Holant value is exactly the number of solutions of the linear system, which can be computed in polynomial time.

Theorem 2. *If* $\mathcal{F} \subseteq \mathscr{A}$, $\oplus\text{Holant}^c(\mathcal{F})$ *is polynomial time computable.*

3.2 Fibonacci Signatures and $[0, 1, 0]$

Definition 5. *A symmetric signature* $[f_0, f_1, \ldots, f_n]$ *is called a Fibonacci signature iff for* $1 \leq k \leq n-2$*, it is the case that* $x_k + x_{k+1} = x_{k+2}$*. We denote the set of all Fibonacci signatures by* $\mathscr{F}$*.*

The family of Fibonacci signatures was introduced in [9] to characterize a new family of holographic algorithms. It has played an important role in some previous dichotomy theorems [9,11]. The Holant of a grid composed of Fibonacci signatures can be computed in polynomial time [9]. Its parity version is therefore also tractable. But here we shall show that the tractability still holds even if we extend the set to contain the signature $[0, 1, 0]$, which is not a Fibonacci signature. This proof of tractability is based on the properties of Fibonacci signatures and a new observation on $[0, 1, 0]$ as a parity signature.

Since we only care about the parity of the solutions, $[0, 1, 0]$ can be replaced by the unsymmetrical signature $(0, 1, -1, 0)$ in $\mathbb{R}$. (Note that here $(0, 1, -1, 0)$ is not a symmetric signature. It is in fact in the vector form, rather than the abbreviated form of symmetric signatures.) This $(0, 1, -1, 0)$ is a so-called *2-realizable signature*, which is invariant up to a constant under holographic transformations [3,1,7,8]. Polynomial time computability follows from the facts that every Fibonacci signature can be transformed into a form similar to equality signatures while leaving invariant the signature $(0, 1, -1, 0)$.

Theorem 3. *If* $\mathcal{F} \subseteq \mathscr{F} \cup \{[0, 1, 0]\}$*,* $\oplus\text{Holant}^c(\mathcal{F})$ *is polynomial time computable.*

3.3 Matchgate Signatures

Definition 6. *A signature is called a matchgate signature iff it can be realized by a gadget, where each signature used in the gadget is a perfect matching signature* $([0, 1, 0, 0, \ldots, 0])$ *or a partial matching signature* $([1, 1, 0, 0, \ldots, 0])$*. We denote the set of all matchgate signatures by* $\mathscr{M}$*.*

Matchgates were introduced to simulate classically certain subclasses of quantum computations [30] and to be the basis of a class of holographic algorithms [32]. We remark that the notion of matchgate we use here is in its most general sense: the graph can be either planar or non-planar and for each node we can insist or not on whether it has to be saturated by a matching edge.

As $\mathcal{F} \subseteq \mathscr{M}$ and we also have $[1, 0], [0, 1] \in \mathscr{M}$, the problem of $\oplus\text{Holant}^c(\mathcal{F})$ is essentially that of computing the parity of the number of matchings in a graph where some specified nodes must be saturated while the remainder need not be. We show that the parity of the number of matchings equals the Pfaffian of a certain matrix of even arity in $\mathbb{Z}_2$. If the parity of the perfect matchings only is needed then such a result is immediate. What we show is that it is true also in the more general case.

Theorem 4. *If* $\mathcal{F} \subseteq \mathscr{M}$*,* $\oplus\text{Holant}^c(\mathcal{F})$ *is polynomial time computable.*

4 Hardness Results and Dichotomy for $\oplus\text{Holant}^c$

In this section, we prove several hardness results. These results, together with the tractable results in previous section, lead to the dichotomy theorem for $\oplus\text{Holant}^c$.

4.1 An Initial Hard Problem

The following hardness result from [31] is the starting point for all the hardness results in this paper.

Theorem 5. *$\oplus$Pl-Rtw-Mon-3CNF is $\oplus$P-complete. In the Holant language, Planar $\oplus\text{Holant}([0,1,1,1])$ is $\oplus$P-complete.*

Remark: All the hardness results in this paper for $\oplus\text{Holant}^c$, but not for $\oplus$Holant, will hold even if we restrict the input to planar graphs. This is because the above starting point is true for planar graphs, and all the gadgets used in those reductions are also planar. In the following, for brevity, we will not explicitly refer to this.

This $\oplus\text{Holant}([0,1,1,1])$ can also be viewed as $\oplus\text{Holant}([1,0,1]|[0,1,1,1])$. Under the holographic transformation $\begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}$, the Holant value of $\oplus$Holant $([1,0,1]\,|[0,1,1,1])$ is the same as that of $\oplus$Holant $([1,1,0]|[1,0,0,1])$. This gives the following hardness result for vertex covers:

Corollary 1. *$\oplus\text{Holant}([0,1,1]|[1,0,0,1])$, and $\oplus\text{Holant}([1,1,0]|[1,0,0,1])$ are $\oplus$P-complete.*

Corollary 2. *$\oplus\text{Holant}^c([0,1,1],[1,0,\ldots,0,1])$ is $\oplus$P-complete, as long as the number of 0s is at least 2.*

4.2 More Hardness Results and the Dichotomy

We establish some further hardness results for other signatures. These results will be used to obtain subsequent hardness results for certain longer signatures and sets of signatures.

Lemma 1. *$\oplus\text{Holant}^c([0,1,0,1,0],[0,1,1,1,0])$ is $\oplus$P-complete.*

The proof of this lemma utilizes some novel gadget and holographic transformation. It can be further generalized because of certain realizability properties of matchgates and Fibonacci signatures.

Corollary 3. *If $\mathcal{F}$ contains a non-degenerate symmetric signature in $\mathscr{M}$ and a non-degenerate Fibonacci signature, both of which have arity at least 3, then $\oplus\text{Holant}^c(\mathcal{F})$ is $\oplus$P-complete.*

This result implies that simultaneous occurrences of matchgates and Fibonacci signatures lead to ⊕P-completeness. Similarly, we have the following lemma, which shows that the simultaneous occurrences of matching signatures and equality signatures also lead to ⊕P-completeness.

Lemma 2. *The parity problems* $\oplus\text{Holant}^c$ $([0,0,1,0],\ [1,0,0,\ldots,0,1])$, $\oplus\text{Holant}^c$ $([0,1,0,0],\ [1,0,0,\ldots,0,1])$, $\oplus\text{Holant}^c$ $([0,0,1,1],\ [1,0,0,\ldots,0,1])$ *and* $\oplus\text{Holant}^c$ $([1,1,0,0],\ [1,0,0,\ldots,0,1])$ *are all* $\oplus$*P-complete if the arity of the equality signature is at least 3.*

This lemma entails the following direct corollary for signatures that contain both equality and matching signatures as subsignatures.

Corollary 4. $\oplus\text{Holant}^c([1,0,\ldots,0,1,0])$ *and* $\oplus\text{Holant}^c([1,0,\ldots,0,1,1])$ *are* $\oplus$*P-complete, as long as the number of 0s is at least 2.*

There are still two special patterns of signatures that we need to take care of.

Lemma 3. $\oplus\text{Holant}^c([0,0,1,0,0])$ *and* $\oplus\text{Holant}^c([0,0,1,0,1])$ *are* $\oplus$*P-complete.*

Based on the algorithms in Section 3 and the hardness results above, we show a dichotomy theorem for parity Holant^c problems. The proof is basically a case-by-case study based on the number of consecutive 0s or 1s.

Theorem 6. *If* $\mathcal{F} \subseteq \mathscr{A}$, $\mathcal{F} \subseteq \mathscr{M}$ *or* $\mathcal{F} \subseteq \mathscr{F} \cup \{[0,1,0]\}$ *then the parity problem* $\oplus\text{Holant}^c(\mathcal{F})$ *is computable in polynomial time. Otherwise it is* $\oplus$*P-complete. The same statement also holds for planar graphs.*

5 Vanishing Signature Sets

In the remaining two sections we extend our results to obtain the dichotomy result for ⊕Holant without any assumptions. In order to formulate the dichotomy we shall need a fourth family of tractable signature sets, which we call *Vanishing Signature Sets*.

Definition 7. *A set of signatures* $\mathcal{F}$ *is called vanishing iff the value of* $\oplus\text{Holant}_\Omega(\mathcal{F})$ *is zero for every* Ω*. We denote the class of all vanishing signature sets by* $\mathscr{O}$*.*

First we show some general properties of vanishing signature sets. For two signatures f and g of the same arity, $f + g$ denotes the bitwise addition in $\mathbb{Z}_2$, i.e. $[f_0 + g_0, f_1 + g_1, \ldots]$.

Lemma 4. *Let* $\mathcal{F}$ *be a vanishing signature set. If a signature* f *can be realized by a gadget using signatures in* $\mathcal{F}$*, then* $\mathcal{F} \cup \{f\} \in \mathscr{O}$*. If* g_0 *and* g_1 *are two signatures in* $\mathcal{F}$ *with the same arity, then* $\mathcal{F} \cup \{g_0 + g_1\} \in \mathscr{O}$*.*

There are several classes of vanishing signatures, e.g. complement invariant signatures and matchgate-based vanishing signatures. Here we introduce a concept called *self-vanishable signatures* which plays an important role in the proof of the general dichotomy. First, we introduce an extended version of the inner product for two signatures of not necessarily the same arity.

Definition 8. *Let f and g be two signatures with arities n and m ($n \geq m$) respectively. Their inner product $h = \langle f, g \rangle$ is a signature with arity $n - m$ defined as follows:*

$$h^{\alpha} = \sum_{\beta \in \{0,1\}^m} f^{\beta,\alpha} g^{\beta},$$

where $\alpha \in \{0,1\}^{n-m}$.

If f is symmetric, the final $h = \langle f, g \rangle$ is also symmetric. If both f and g are symmetric, their inner product $h = [h_0, h_1, \ldots, h_{n-m}]$ has the following form: $h_i = \sum_{j=0}^{m} \binom{m}{j} f_{j+i} g_j$ for $0 \leq i \leq n - m$.

Definition 9. *A signature f is called self-vanishable of degree k iff $\langle f, [1,1]^{\otimes k} \rangle = \mathbf{0}$ and $\langle f, [1,1]^{\otimes k-1} \rangle \neq \mathbf{0}$. We denote this by $v(f) = k$. If such a k does not exist, the signature f is not self-vanishable.*

We note that for the trivial signature $\mathbf{0}$, we have $v(\mathbf{0}) = 0$. Also, $f = [1,1]$ is self-vanishable with $v(f) = 1$ since $\langle [1,1], [1,1] \rangle = 0$.

For a symmetric signature $f = [f_0, f_1, \ldots, f_n]$, we call f_0 the first entry of f and $f_0, f_1, ..., f_{k-1}$ the first k entries of f. It follows from the definition that for a symmetric signature $f = [f_0, f_1, \ldots, f_n]$, we have

$$\langle f, [1,1] \rangle = [f_0 + f_1, f_1 + f_2, \ldots, f_{n-1} + f_n].$$

Hence the only symmetric signature of arity n with $v(f) = 1$ is $[1,1]^{\otimes n}$. There are two symmetric signatures of arity $n \geq 3$ with $v(f) = 2$, which are the parity signatures $[1, 0, 1, 0, \ldots, 0/1]$ and $[0, 1, 0, 1, \ldots, 0/1]$. Inductively, we have the following lemma:

Lemma 5. *For any $k \geq 2$, there are 2^{k-1} symmetric signatures of arity $n \geq k$ with $v(f) = k$, whose first $k - 1$ entries are arbitrary and the remaining entries are determined by them.*

To be self-vanishable is a necessary condition for a signature to be a member of a vanishing signature set. This lemma also explains the intuition for why we define this notion of self-vanishable and why we define it in this way. The proof is a direct construction of a grid with Holant value 1.

Lemma 6. *If $\mathcal{F}$ contains a signature f which is not self-vanishable then $\mathcal{F}$ is not a vanishing set.*

However, it is not sufficient for a signature to be self-vanishable for it to form a vanishing set. One condition that is sufficient, called strong self-vanishable, is defined below. There exist some weak self-vanishable signatures that do not form vanishing sets, e.g. $\{[1, 0, 0, 0, 1, 0]\}$.

Definition 10. *Let f be self-vanishable of degree $k \geq 0$ with arity n. It is called strong self-vanishable if $k \leq \lfloor \frac{n}{2} \rfloor + 1$ and weak self-vanishable if $\lfloor \frac{n}{2} \rfloor + 2 \leq k \leq n$.*

Theorem 7. *Let $\mathcal{F}$ be a set of symmetric strong self-vanishable signatures. Then $\mathcal{F}$ is a vanishing set, i.e. $\mathcal{F} \in \mathscr{O}$.*

As a final remark we note that the family $\mathscr{O}$ of vanishing signature sets has the following difference from the previous tractable families $\mathscr{A}$, $\mathscr{M}$ and $\mathscr{F} \cup \{[0,1,0]\}$. The union of two sets in $\mathscr{O}$ is not necessarily in $\mathscr{O}$.

6 Dichotomy for the Whole Holant Family

In this final section, we prove our main theorem, the dichotomy for all parity Holant problems with symmetric signatures, without assuming any freely available signatures. This improves on our dichotomy theorem for parity Holant^c problems given in Section 4, which we use, however, as our starting point. The main idea is to construct gadgets for the two signatures $[0,1]$ and $[1,0]$. We will first show that realizing either one of these is enough. Where one of these unary signatures is realizable, we reduce the Holant problem to the corresponding Holant^c problem and apply the Holant^c dichotomy result. However, for some signature sets it is impossible to realize $[0,1]$ or $[1,0]$. We show that those signature sets must be vanishing, in the sense defined in the previous section.

First we show that it is enough to realize just one of $[0,1]$ or $[1,0]$. We remark that the gadgets used in the proof are not all planar, and hence the dichotomy for planar graphs does not follow.

Lemma 7. *Let $\mathcal{F}$ be a set of symmetric signatures. If $\mathcal{F} \subseteq \mathscr{A}$, $\mathcal{F} \subseteq \mathscr{M}$, or $\mathcal{F} \subseteq \mathscr{F} \cup \{[0,1,0]\}$ then the parity problems $\oplus\text{Holant}(\mathcal{F} \cup \{[1,0]\})$, $\oplus\text{Holant}(\mathcal{F} \cup \{[1,0,0]\})$, $\oplus\text{Holant}(\mathcal{F} \cup \{[0,1]\})$ and $\oplus\text{Holant}(\mathcal{F} \cup \{[0,0,1]\})$ are computable in polynomial time. Otherwise these parity problems are $\oplus$P-complete.*

The proof of the first part of Lemma 7 is a case-by-case study which shows that we can always realize $[0,1]$ via some signature in $\mathcal{F}$ and $[1,0]$. The second part is shown using reductions from $\oplus\text{Holant}(\mathcal{F} \cup \{[1,0]\})$ to $\oplus\text{Holant}(\mathcal{F} \cup \{[1,0,0]\})$. We duplicate an instance of $\oplus\text{Holant}(\mathcal{F} \cup \{[1,0]\})$, and replace every two corresponding occurrences of $[1,0]$ with a binary signature $[1,0,0]$. The Holant value remains the same due to properties of $\mathbb{Z}_2$.

An important fact is that, as long as the signature set $\mathcal{F}$ is not vanishing, one of the above unary or binary signatures is realizable. In particular, we can always construct such a signature from a not self-vanishable signature, or realize either $[0,1]$ or $[1,0]$ from a weak self-vanishable signature. This gives our main theorem.

Theorem 8. *Let $\mathcal{F}$ be a set of symmetric signatures. If $\mathcal{F} \subseteq \mathscr{A}$, $\mathcal{F} \subseteq \mathscr{M}$, $\mathcal{F} \subseteq \mathscr{F} \cup \{[0,1,0]\}$, or $\mathcal{F} \in \mathscr{O}$ then the parity problem $\oplus\text{Holant}(\mathcal{F})$ is computable in polynomial time. Otherwise it is $\oplus$P-complete.*

Acknowledgements. We would like to thank anonymous referees for their helpful comments. Part of the work was done when Heng Guo was a master's student in Peking University.

References

1. Arvind, V., Kurur, P.P.: Graph isomorphism is in spp. Inf. Comput. 204(5), 835–852 (2006)
2. Beigel, R., Buhrman, H., Fortnow, L.: Np might not be as easy as detecting unique solutions. In: STOC 1998: Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing, pp. 203–208 (1998)
3. Bulatov, A.A.: A dichotomy theorem for constraint satisfaction problems on a 3-element set. J. ACM 53(1), 66–120 (2006)
4. Bulatov, A.A.: The complexity of the counting constraint satisfaction problem. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part I. LNCS, vol. 5125, pp. 646–661. Springer, Heidelberg (2008)
5. Cai, J.-Y., Chen, X., Lu, P.: Graph homomorphisms with complex values: A dichotomy theorem. In: Abramsky, S., Gavoille, C., Kirchner, C., Meyer auf der Heide, F., Spirakis, P.G. (eds.) ICALP 2010, Part I. LNCS, vol. 6198, pp. 275–286. Springer, Heidelberg (2010)
6. Cai, J.Y., Huang, S., Lu, P.: From holant to #CSP and back: Dichotomy for holant^c problems. arXiv 1004.0803 (2010)
7. Cai, J.Y., Lu, P.: Holographic algorithms: from art to science. In: STOC 2007: Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing, pp. 401–410. ACM, New York (2007)
8. Cai, J.-Y., Lu, P.: Signature theory in holographic algorithms. In: Hong, S.H., Nagamochi, H., Fukunaga, T. (eds.) ISAAC 2008. LNCS, vol. 5369, pp. 568–579. Springer, Heidelberg (2008)
9. Cai, J.Y., Lu, P., Xia, M.: Holographic algorithms by fibonacci gates and holographic reductions for hardness. In: FOCS 2008: Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science. IEEE Computer Society Press, Washington, DC, USA (2008)
10. Cai, J.Y., Lu, P., Xia, M.: A computational proof of complexity of some restricted counting problems. In: Chen, J., Cooper, S.B. (eds.) TAMC 2009. LNCS, vol. 5532, pp. 138–149. Springer, Heidelberg (2009)
11. Cai, J.Y., Lu, P., Xia, M.: Holant problems and counting CSP. In: Mitzenmacher, M. (ed.) STOC, pp. 715–724. ACM, New York (2009)
12. Cai, J.Y., Lu, P., Xia, M.: Holographic algorithms with matchgates capture precisely tractable planar #CSP. In: FOCS 2010: Proceedings of the 49th Annual IEEE Symposium on Foundations of Computer Science, pp. 427–436 (2010)
13. Cook, M., Bruck, J.: Implementability among predicates. Tech. rep., California Institute of Technology (2005)
14. Creignou, N., Khanna, S., Sudan, M.: Complexity classifications of boolean constraint satisfaction problems. SIAM Monographs on Discrete Mathematics and Applications (2001)
15. Creignou, N., Hermann, M.: Complexity of generalized satisfiability counting problems. Inf. Comput. 125(1), 1–12 (1996)
16. Dodson, C.T.J., Poston, T.: Tensor Geometry. Graduate Texts in Mathematics, vol. 130. Springer, New York (1991)
17. Dyer, M.E., Goldberg, L.A., Jerrum, M.: The complexity of weighted boolean #CSP. SIAM J. Comput. 38(5), 1970–1986 (2009)
18. Dyer, M.E., Goldberg, L.A., Paterson, M.: On counting homomorphisms to directed acyclic graphs. J. ACM 54(6) (2007)

19. Faben, J.: The complexity of counting solutions to generalised satisfiability problems modulo k. CoRR abs/0809.1836 (2008)
20. Feder, T., Vardi, M.: The computational structure of monotone monadic snp and constraint satisfaction: A study through datalog and group theory. SIAM Journal on Computing 28(1), 57–104 (1999)
21. Goldberg, L.A., Grohe, M., Jerrum, M., Thurley, M.: A complexity dichotomy for partition functions with mixed signs. In: Albers, S., Marion, J.Y. (eds.) STACS. LIPIcs, vol. 3, pp. 493–504. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany (2009)
22. Guo, H., Huang, S., Lu, P., Xia, M.: The complexity of weighted boolean #csp modulo k. In: Schwentick, T., Dürr, C. (eds.) STACS. LIPIcs, vol. 9, pp. 249–260. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2011)
23. Kowalczyk, M., Cai, J.Y.: Holant problems for regular graphs with complex edge functions. In: The Proceeding of STACS (2010)
24. Ladner, R.E.: On the structure of polynomial time reducibility. J. ACM 22(1), 155–171 (1975)
25. Papadimitriou, C.H., Zachos, S.: Two remarks on the power of counting. In: Proceedings of the 6th GI-Conference on Theoretical Computer Science, pp. 269–276 (1982)
26. Schaefer, T.: The complexity of satisfiability problems. In: Proceedings of the Tenth Annual ACM Symposium on Theory of Computing, p. 226. ACM, New York (1978)
27. Toda, S., Ogiwara, M.: Counting classes are at least as hard as the polynomial-time hierarchy. SIAM J. Comput. 21(2), 316–328 (1992)
28. Valiant, L.G., Vazirani, V.V.: NP is as easy as detecting unique solutions. Theor. Comput. Sci. 47(1), 85–93 (1986)
29. Valiant, L.G.: The complexity of computing the permanent. Theor. Comput. Sci. 8, 189–201 (1979)
30. Valiant, L.G.: Quantum circuits that can be simulated classically in polynomial time. SIAM J. Comput. 31(4), 1229–1254 (2002)
31. Valiant, L.G.: Accidental algorthims. In: FOCS 2006: Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science, pp. 509–517. IEEE Computer Society Press, Washington, DC, USA (2006)
32. Valiant, L.G.: Holographic algorithms. SIAM J. Comput. 37(5), 1565–1594 (2008)
33. Valiant, L.G.: Some observations on holographic algorithms. In: López-Ortiz, A. (ed.) LATIN 2010. LNCS, vol. 6034, pp. 577–590. Springer, Heidelberg (2010)

Permanent Does Not Have Succinct Polynomial Size Arithmetic Circuits of Constant Depth

Maurice Jansen and Rahul Santhanam

School of Informatics, The University of Edinburgh
maurice.julien.jansen@gmail.com, rsanthan@inf.ed.ac.uk

Abstract. We show that over fields of characteristic zero there does not exist a polynomial $p(n)$ and a constant-free[1] *succinct* arithmetic circuit family $\{\Phi_n\}$, where Φ_n has size at most $p(n)$ and depth $O(1)$, such that Φ_n computes the $n \times n$ permanent. A circuit family $\{\Phi_n\}$ is succinct if there exists a *nonuniform* Boolean circuit family $\{C_n\}$ with $O(\log n)$ many inputs and size $n^{o(1)}$ such that that C_n can correctly answer direct connection language queries about Φ_n - succinctness is a relaxation of uniformity.

To obtain this result we develop a novel technique that further strengthens the connection between black-box derandomization of polynomial identity testing and lower bounds for arithmetic circuits. From this we obtain the lower bound by explicitly constructing a hitting set against arithmetic circuits in the polynomial hierarchy.

1 Introduction

Proving super-polynomial arithmetic circuit lower bounds for explicit polynomials is one of the hardest challenges in theoretical computer science. For unrestricted circuits the best-known lower bounds are $\Omega(n \log r)$, for polynomials in n variables and degree r, due to Baur and Strassen [6]. To make further progress one popular approach has been to aim at proving lower bounds for small depth circuits first. This restriction is well-motivated by a recent result of Agrawal and Vinay [3], who show that proving a $2^{\Omega(n)}$ lower bound for a multilinear polynomial in n variables for depth four arithmetic circuits translates into a $2^{\Omega(n)}$ lower bound for the unrestricted model. Over finite fields, Grigoriev and Karpinski [9] show that depth three arithmetic formulas that are sums-of-products-of-sums require size $2^{\Omega(n)}$ to compute the determinant polynomial $\det_n = \sum_{\sigma \in S_n} \operatorname{sgn}(\sigma) \prod_{j=1}^{n} x_{i\sigma(j)}$. Over fields of characteristic zero, e.g. $\mathbb{Q}$ or $\mathbb{R}$, super-polynomial lower bounds have proven to be especially difficult to obtain. The best-known lower bound for depth three sums-of-products-of-sums formulas for $\det_n$, and also for the permanent polynomial $\operatorname{per}_n = \sum_{\sigma \in S_n} \prod_{j=1}^{n} x_{i\sigma(j)}$ is $\Omega(n^4/\log n)$, due to Shpilka and Wigderson [21]. Over fields of characteristic

[1] In the constant-free model the only allowed constants in the circuit are $-1, 0, 1$, cf. [7,16]. Our result holds for a slightly more general setting where Φ_n is allowed to use integer constants in the set $\{-1, 0, 1\} \cup \{2^{2^0}, 2^{2^1}, \ldots, 2^{2^{p(n)}}\}$.

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 724–735, 2011.

zero, no lower bounds for this depth three model are known beyond $\Omega(n^2)$, for explicit polynomials in n variables of degree at most $poly(n)$. For higher depth, the best-known bounds are due to Raz [18], who constructs a polynomial with coefficients in $\{0,1\}$ of degree $O(d)$ in n variables that requires depth d circuits of size $n^{1+\Omega(1/d)}$ over any field. Adding the restriction of multilinearity[2] Raz and Yehudayoff [19] prove that $\det_n$ and per_n require size $2^{n^{\Omega(1/d)}}$ for multilinear circuits of product depth d. For constant-free uniform[3] arithmetic circuits it is known that depth $o(\log\log n)$ circuits for per_n must have super-polynomial size, as proved by Koiran and Perifel [17]. For such circuits of constant depth, earlier work of Allender [4] implies a super-polynomial lower bound.

We interpret the result of Ref. [17] as a lower bound for circuits that are *succinct* in some extreme sense. In order to make progress we will prove lower bounds for a far less restrictive notion of succinctness. To make this precise, we give some definitions. Consider a family $\{\Phi_n\}$ of constant-free arithmetic circuits. We say that $\{\Phi_n\}$ is $(a(n), b(n))$-succinct[3], if there exists a family of *nonuniform* Boolean circuits $\{C_n\}$, where C_n has at most $a(n)$ inputs and is of size at most $b(n)$, such that C_n can answer direct connection language queries about Φ_n. This means that given names of gates in Φ_n, the circuit C_n should be able to answer queries like whether two gates are connected, what the type of a gate is: $+$, $\times$, or 'input', and in the latter case whether the variable or constant label equals some given string.

Succinctness interpolates between uniformity and non-uniformity. DLOGTIME-uniform circuits of polynomial size are $(O(\log n), O(\log n))$-succinct. On the other hand, if a sequence $\{\Phi_n\}$ of circuits is $(a(n), b(n))$-succinct, the circuit Φ_n can be constructed in time $2^{O(a(n))}b(n)^{O(1)}$, given $b(n)$ bits of advice.

In what follows we take $a(n) = O(\log n)$, which limits the size of the arithmetic circuit to be $n^{O(1)}$. By convention, whenever $a(n) = O(\log n)$, we will drop it from the notation, and just write "$b(n)$-succinct".

Within this setting there is a spectrum of scenarios to study. At one extreme, $poly(n)$-succinctness forms no restriction at all. At the other end, for $polylog(n)$-succinctness, a lower bound can be derived[4] using the known uniform TC^0 lower bound for permanent by Allender [4]. Our main result is to push the envelope towards the high end by proving the following theorem[5]:

Theorem 1. *Let $d \geq 0$ be an integer. Over fields of characteristic zero, there does not exist a polynomial $p(n)$ such that $\{\mathrm{per}_n\}$ has $n^{o(1)}$-succinct size $p(n)$ depth d arithmetic circuits using constants in the set $\Gamma_{p(n)} = \{-1, 0, 1\} \cup \{2^{2^0}, 2^{2^1}, \ldots, 2^{2^{p(n)}}\}$.*

[2] This means that at any gate the computed polynomial must be multilinear.

[3] See Section 3 for a formal definition. .

[4] This will appear in the full version of the paper.

[5] We have strengthened this result in several respects in a recent paper [13] by going through derandomization of univariate polynomial identity testing. Moreover, the resulting proof is conceptually cleaner.

Thus either the permanent does not have polynomial size constant depth constant-free arithmetic circuits, or in the odd case that it does, the corresponding direct connection language L that is hard in the sense that for deciding inputs of size $k = O(\log n)$, for some $\epsilon > 0$ one requires nonuniform Boolean circuits of size $n^{\epsilon} = 2^{\Omega(k)}$, i.e. exponential in the input size[6].

Why are we interested in lower bounds against succinct circuits? If we're interested in actually computing the permanent, the mere fact of there being polynomial-size circuits for it doesn't seem to be sufficient, for there is the question of how we can find these circuits. Succinctness is a measure of how easy the circuits themselves are.

Also, ultimately, much of the motivation for arithmetic complexity comes from classical problems like NP vs P and PSPACE vs P, which have resisted solution, or even progress, for many decades. Often the nonuniform versions of these questions are studied because we have few ways of taking advantage of uniformity apart from diagonalization. One aspect of our proof is that we do not make use of hierarchy theorems[7], and this methodology might prove useful in other contexts. We next describe the actual techniques used in the proof.

1.1 Techniques

At a high level, we exploit the well-known connection between lower bounds and derandomization of arithmetic circuit identity testing (ACIT). In the ACIT problem one is given an arithmetic circuit, and the question is to decide whether the polynomial computed by the circuit is identically zero or not. Using the Schwartz-Zippel-deMillo-Lipton Lemma [8,20,28], Ibarra and Moran [12] show this problem is in coRP. It is known that non-trivial *deterministic* algorithms would lead to circuit lower bounds against Boolean or arithmetic circuits [10,14,1]. This is often interpreted as evidence for the difficulty of derandomizing ACIT. However Agrawal turns this interpretation on its head and advocates derandomization of ACIT as an *approach* toward proving circuit lower bounds [1]. Our result is a step in this direction.

There are two major parts to our argument. The first is to show that the lower bound of Theorem 1 follows from a black-box derandomization hypothesis (Working Hypothesis 1) for ACIT. Our Working Hypothesis 1 poses the existence of integer sequences definable by $n^{o(1)}$-succinct TC^0 circuits that form a hitting set against small constant-free arithmetic circuits. We prove the implication by combining ideas of Agrawal [1], Bürgisser [7] and Koiran [15], and making critical use of our succinctness assumption to indirectly perform computations which we cannot afford to do directly.

We do not know how to prove Working Hypothesis 1. Instead what we will establish for the second part of our argument, is a weaker derandomization of

[6] It is possible to give a uniform upper of $\mathrm{E}^{\mathrm{NP}^{\mathrm{RP}}}$ for L.

[7] One of the referees of this paper sketched a potential alternative route to our lower bounds using 'out of the box' diagonalization. We have not fully explored this argument as of yet.

ACIT using integer sequences so-called *weakly-definable* in the polynomial hierarchy. However, this statement appears not to be strong enough to get the lower bound of Theorem 1 directly. To resolve this, we argue by contradiction as follows. Assuming that the conclusion of Theorem 1 fails, then due to Toda's Theorem [22] and an improvement by Zankó [27], cf. [4], of Valiant's [24] completeness result for per_n, this induces a collapse of the polynomial hierarchy, which makes the integer sequence we explicitly construct good enough to satisfy the requirements of Working Hypothesis 1. This way we obtain a lower bound for per_n after all, but this contradicts the assumption that the conclusion of Theorem 1 does not hold.

2 Preliminaries

An arithmetic circuit Φ over variables $X = \{x_1, x_2, \ldots, x_n\}$ and a field $\mathbb{F}$ is given by a directed acyclic graph such that nodes with in-degree $= 0$ are labeled with elements of $X \cup \mathbb{F}$. Nodes with higher in-degree are labeled by $+$ or $\times$. To each vertex in Φ we can associate an element of the polynomial ring $\mathbb{F}[X]$ in the obvious way. A polynomial $f \in \mathbb{F}[X]$ is said to be computed by Φ, if there exists a node in Φ where the associated polynomial equals f. The size $s(\Phi)$ of the circuit is defined to be the number of edges in Φ. We will also write $|\Phi|$ for $s(\Phi)$. For a polynomial $f \in \mathbb{F}[X]$, its arithmetic circuit complexity $s(f)$ is defined by $s(f) = \min\{s(\Phi) : \Phi \text{ computes } f\}$. If the underlying graph of Φ is a tree, then Φ is called a formula. Arithmetic formula size of f is denoted by $e(f)$.

Next we list several of the important complexity classes used in the paper. #P is the class of all function $f : \{0,1\}^* \to \{0,1\}^*$ such that there exists a language $A \in \mathrm{P}$ and a polynomial $p(n)$ such that $f(x) = |\{w \in \{0,1\}^{p(|x|)} : (x,w) \in A\}|$. GapP is the class of all function $f - g$, where $f, g \in \#\mathrm{P}$. We will make combined use of Valiant's [24] result that computing $\mathrm{per}_n(M)$ for M with entries in $\{0,1\}$ over $\mathbb{Z}$ is complete for #P, and Toda's [22] Theorem, which states that $\mathrm{PH} \subseteq \mathrm{P}^{\#\mathrm{P}[1]}$. We define the majority operator **C.** acting on a complexity class as follows. Given a complexity class $\mathcal{C}$, $\mathbf{C}.\mathcal{C}$ is the class of all languages L for which there exists $L' \in \mathcal{C}$ and a polynomial $p(n)$ such that $x \in L \Leftrightarrow |\{w \in \{0,1\}^{p(|x|)} : (x,w) \in L'\}| > 2^{p(|x|)-1}$. The counting hierarchy, introduced by Wagner [26], is defined to be $\bigcup_{i \geq 0} \mathrm{C}_i\mathrm{P}$, where $\mathrm{C}_0\mathrm{P} = \mathrm{P}$, and for all $i \geq 1$, $\mathrm{C}_i\mathrm{P} = \mathbf{C}.\mathrm{C}_{i-1}\mathrm{P}$. The first level $\mathrm{C}_1\mathrm{P}$ of this hierarchy corresponds to the standard complexity class PP. We will use Torán's [23] characterization of the counting hierarchy which states that $\mathrm{C}_{i+1}\mathrm{P} = \mathrm{PP}^{\mathrm{C}_i\mathrm{P}}$, for all $i \geq 0$. Next we define nonuniform versions of complexity classes. An advice function is a function of type $h : \mathbb{N} \to \{0,1\}^*$. For a complexity class $\mathcal{C}$, we define $\mathcal{C}/poly$ to be the class of languages for which there exists $L' \in \mathcal{C}$, and advice function h with $|h(n)| = n^{O(1)}$, such that $x \in L \Leftrightarrow (x, h(|x|)) \in L'$. We use the Boolean circuit complexity classes AC^0, TC^0 and NC^1. AC^0 is the class of all Boolean functions computable by polynomial size constant depth circuits with unbounded fan-in gates in $\{\vee, \wedge, \neg\}$. TC^0 is the class of all Boolean function that can be decided by polynomial size constant depth unbounded fan-in threshold circuits. A threshold

circuit is a Boolean circuit in which all gates either compute the negation, or the majority function of their inputs. NC^1 is the class of all Boolean functions that can be decided by polynomial size $O(\log n)$ depth circuits of bounded fan-in. For these classes we have that $\mathrm{AC}^0 \subseteq \mathrm{TC}^0 \subseteq \mathrm{NC}^1$. For a Boolean circuit family $\{C_n\}$, if there are no requirements on constructability, we call the family nonuniform. For the uniform versions of Boolean complexity classes we will always be using the notion of DLOGTIME-uniformity. We define this below.

We will use the notion of definability of Ref. [16]. To distinguish this from definability as in Ref. [7], we use the term *weakly-definable*. An integer sequence of bit size $q(n)$ is given by a function $a(n, k_1, k_2, \ldots, k_t)$, for some fixed number t, such that there exist polynomials $p(n)$ and so that $a(n, k_1, k_2, \ldots, k_t) \in \mathbb{Z}$ is defined for all $n \geq 0$, and all $0 \leq k_1, k_2, \ldots, k_t < 2^{p(n)}$, and where the bit size of $a(n, k_1, k_2, \ldots, k_t)$ is bounded by $q(n)$. We will often write $a_n(k_1, k_2, \ldots, k_t)$ instead of $a(n, k_1, k_2, \ldots, k_t)$. We define the language

$$uBit(a) = \{(1^n, k_1, k_2, \ldots, k_t, j, b) : \text{the } j\text{th bit of } a(n, k_1, k_2, \ldots, k_t) \text{ equals } b\}.$$

Here $k_1, k_2, \ldots, k_t$ and j are encoded in binary. For a sequence $a(n, k_1, k_2, \ldots, k_t)$ and a complexity class $\mathcal{C}$, if $uBit(a) \in \mathcal{C}$, then we say that the sequence $a(n, k_1, k_2, \ldots, k_t)$ is weakly-definable in $\mathcal{C}$. Finally, we require using the Schwartz-Zippel-deMillo-Lipton Lemma.

Lemma 1 ([8,20,28]). *Let A be a nonempty subset of the field $\mathbb{F}$. Then for any nonzero polynomial $f \in \mathbb{F}[X]$ of degree d, $\Pr[f(a_1, a_2, \ldots, a_n) = 0] \leq \frac{d}{|A|}$, where the a_is are picked independently and uniformly at random from A.*

3 Representing Arithmetic Circuits over $\mathbb{Z}$ by Boolean Circuits Succinctly

For *constant-free* arithmetic circuits the only field constants that are allowed for labels are $\in \{-1, 0, 1\}$, cf. [7,16]. We work over a slightly more general model. Define for $n \geq 0$, the set integer constants $\Gamma_n = \{2^{2^0}, 2^{2^1}, \ldots, 2^{2^n}\} \cup \{-1, 0, 1\}$. For the set $\{x_1, x_2, \ldots, x_n\} \cup \Gamma_n \cup \{+, \times\}$, we assume we have fixed some naming scheme that assigns to each element an $O(\log n)$ bit binary string, which is called a *type*. We assume that all gates in a circuit have been labeled by unique binary strings that also specify the type.

A *representation* of an arithmetic circuit Φ with constants $\in \Gamma_k$ is given by a Boolean circuit C_n that accepts precisely all tuples (t, a, b, q) such that 1) In case $q = 1$ (connection query), a and b are numbers of gates in Φ, b is a child of a, and a has type t. 2) In case $q = 0$ (type query only), a is a number of a gate in Φ, and a is of type t.

Let $a(n), b(n)$ be two functions $\mathbb{N} \to \mathbb{N}$. For a family of arithmetic circuits $\{\Phi_n\}$, we say it is $(a(n), b(n))$-*succinct*, if there exists a non-uniform family of Boolean $\{\vee, \wedge, \neg\}$-circuits $\{C_n\}$, such that C_n represents Φ_n, where for all large enough n, C_n has $\leq a(n)$ inputs and is of size $\leq b(n)$. As a matter of convention, if $a(n) = O(\log n)$, we drop it from the notation, and just write $b(n)$-succinct.

We want to study the notion of $n^{o(1)}$-succinctness. We will fix some arbitrarily slow growing function $\gamma(n)$ and consider $n^{1/\gamma(n)}$-succinctness instead. A typical example to think of would be $\gamma(n) = \log^* n$. For the rest of the paper we let $\gamma(n) : \mathbb{N} \to \mathbb{N}$ be an unbounded monotone function, such that $\forall n, \gamma(n) < \log\log n$.

Similarly to the above, we define the notion of $(a(n), b(n))$-succinct Boolean circuits. In this case type names are assumed to form a naming scheme for the elements of $\{x_1, x_2, \ldots, x_n\} \cup \{0, 1\} \cup \{\vee, \wedge, \neg, \mathrm{MAJ}\}$.

Regarding DLOGTIME-uniformity we refer the reader to Ref. [5] for an extensive treatment. In our set-up this can be defined as follows. A *poly* size Boolean circuit family $\{C_n\}$ is DLOGTIME-uniform, if given (n, t, a, b, q) with n in binary, we can answer the representation queries as defined above in time $O(\log n)$ on a Turing machine. Using standard conversions from Turing machines to Boolean circuits, observe that if a Boolean circuit family $\{C_n\}$ is DLOGTIME-uniform, then it is $O(\log n)$-succinct, but that a converse of this does not generally hold. We use the following results. For ITERATED INTEGER MULTIPLICATION one is given n integers $A_1, A_2, \ldots, A_n$ of n bits each, and the problem is to compute the bits of $A_1 A_2 \ldots A_n$. Hesse, Allender and Barrington [11] show this can be done with DLOGTIME-uniform TC^0 circuits. For the analogous problem of ITERATED INTEGER ADDITION it is well-known it is in DLOGTIME-uniform TC^0, cf. [25]. Next we state several technical lemmas that deduce consequences from the assumption that $\{\mathrm{per}_n\}$ has $n^{1/\gamma(n)}$-succinct arithmetic circuits. Proofs will appear in the full version of the paper.

Lemma 2. *Assume* $\{\mathrm{per}_n\}$ *can be computed by* $n^{1/\gamma(n)}$*-succinct size* n^{c_0} *depth* d *arithmetic circuits that use constants from* $\Gamma_{n^{c_0}}$*, for some constant* $c_0 > 0$*. Then for some constant* d'*, we can compute* $\mathrm{per}_n(M)$ *over* $\mathbb{Z}$*, where entries of* M *are in* $\{0, 1\}$ *by* $2n^{1/\gamma(n)}$*-succinct* TC^0 *circuits of depth* $d' \cdot d$*.*

Using an improvement of Valiant's completeness result by Zankó [27], cf. [4], who shows that $0, 1$-per_n over $\mathbb{Z}$ is complete for #P under DLOGTIME uniform-AC^0 reductions, one can now easily verify the following statement:

Lemma 3. *Assume* $\{\mathrm{per}_n\}$ *can be computed by* $n^{1/\gamma(n)}$*-succinct size* n^{c_0} *depth* d *arithmetic circuits that use constants from* $\Gamma_{n^{c_0}}$*, for some constant* $c_0 > 0$*. Then for any* $F \in \mathrm{GapP}$ *there exists constants* d', d'' *and* $c' \geq 1$ *such that* F *can be computed by* $n^{c'/\gamma(n)}$*-succinct depth* $d' \cdot d + d''$ TC^0 *circuits.*

Corollary 1. *If* $\{\mathrm{per}_n\}$ *can be computed by* $n^{1/\gamma(n)}$*-succinct size* n^{c_0} *depth* d *arithmetic circuits that use constants from* $\Gamma_{n^{c_0}}$*, for some constant* $c_0 > 0$*, then* $\mathrm{CH}/poly \subseteq nonuniform\text{-}\mathrm{TC}^0$*.*

Lemma 4. *Assume* $\{\mathrm{per}_n\}$ *can be computed by* $n^{1/\gamma(n)}$*-succinct size* n^{c_0} *depth* d *arithmetic circuits that use constants from* $\Gamma_{n^{c_0}}$*, for some integer constant* $c_0 > 0$*. Let* $F : \{0, 1\}^* \to \{0, 1\}^*$ *be a* GapP *function, and let* c' *and* d', d'' *be the constants provided by Lemma 3 for this* F*. Let* (A_n) *be an integer sequence of bit size at most* $\ell(n)$ *that is weakly-definable in* $\mathrm{CH}/poly$*. If it holds that* $\ell(n)^{c'/\gamma(\ell(n))} = n^{O(1)}$ *and* $\log \ell(n) \leq n^{O(1)}$*, then* $b_n := F(A_n)$ *is an integer sequence of bit size* $poly(\ell(n))$ *weakly-definable in* $\mathrm{CH}/poly$*.*

4 Conditional Lower Bound for Permanent

Given an integer sequence $a_n(i,j)$ with $0 \leq i < n$, $0 \leq j < p(n)$ for some function $p(n)$, we think of this as encoding a collection $\{\mathcal{H}_n\}$ of subsets $\mathcal{H}_n \subseteq \mathbb{Z}^n$ with $|\mathcal{H}_n| = p(n)$, where $\mathcal{H}_n = \{(a_n(0,j), a_n(1,j), \ldots, a_n(n-1,j)) : 0 \leq j < p(n)\}$.

Working Hypothesis 1. *Let d be a constant. There exists an integer sequence $a_n(i,j)$ of polynomial bit size with $0 \leq i < n$, $0 \leq j < p(n)$, with p polynomially bounded, such that $uBit(a_n(i,j))$ can be decided by $n^{1/\gamma(n)}$-succinct TC^0 circuits, and for which the following holds:*

- *For any arithmetic circuit Φ of size n over $m \leq n$ variables using constants in $\Gamma_n = \{2^{2^0}, 2^{2^1}, \ldots, 2^{2^n}\} \cup \{-1, 0, 1\}$ of depth at most d, if $\Phi(x_1, x_2, \ldots, x_m)$ computes a nonzero polynomial, then there exist $0 \leq j < p(n)$ such that $\Phi(a_n(0,j), a_n(1,j), \ldots, a_n(m-1,j)) \neq 0$.*

The following is our randomness-to-hardness theorem:

Theorem 2. *If Working Hypothesis 1 is true for depth d, then there does not exist a polynomial $p(n)$ such that $\{\mathrm{per}_n\}$ has size $p(n)$ depth $d-1$ circuits using constants in the set $\Gamma_{p(n)} = \{-1, 0, 1\} \cup \{2^{2^0}, 2^{2^1}, \ldots, 2^{2^{p(n)}}\}$, where in addition these circuits are $n^{1/\gamma(n)}$-succinct.*

Proof. We will argue by contradiction, hence we start by assuming that for some constant $c_0 > 0$, $\{\mathrm{per}_n\}$ can be computed by a family $\{\Phi_n\}$ of size n^{c_0} depth $d-1$ circuits that use constants from $\Gamma_{n^{c_0}}$, and furthermore that for all large enough n, these circuits are represented by a family of Boolean circuits $\{C_n\}$ with $O(\log n)$ inputs and size $n^{1/\gamma(n)}$. Hence by Corollary 1, we have the collapse $\mathrm{CH}/poly \subseteq$ nonuniform-TC^0. We proceed as in Ref. [1] by using the black-box derandomization assumption to construct a hard polynomial by solving a system of linear equations.

Let $a_n(i,j)$ be the integer sequence given by Working Hypothesis 1. Let $\{\mathcal{H}_n\}$ be given as mentioned in the remark before Working Hypothesis 1. Let k be such that we can take $p(n) \leq n^k$ in Working Hypothesis 1, and let c be such that the bit size of any integer $a_n(i,j)$ is bounded by n^c. Choose positive $\epsilon < min(k^{-1}, c^{-1})$. Let $N = \lfloor n^{\epsilon\gamma(n)} \rfloor$, and let $m = \lceil \gamma(n) \log n \rceil$. Then $N^k < 2^m$. Let

$$f_m = \sum_{e=0}^{2^m-1} c_m(e) x_1^{e_1} x_2^{e_2} \ldots x_m^{e_m}, \tag{1}$$

where e_j denotes the jth bit of e. In other words, we sum over all strings in $\{0,1\}^m$ in the above. We want to take $c_m(e)$ to be a nonzero *integer* solution to the system (S) given by $f(b) = 0$, for all $b \in \mathcal{H}_N$. These are at most N^k linear equations in 2^m variables. Since $2^m > N^k$, we can get a nonzero solution. The system (S) is slightly too large to manipulate directly. Coding (S) as an integer sequence weakly-definable in $\mathrm{CH}/poly$ will allow us to indirectly let a solution finding procedure act on (S), due to Lemma 4. For this purpose we think of (S)

as presented by a $2^m \times 2^m$ matrix M_S (with $2^m - N^k$ zero rows). We code M_S as an integer sequence by letting A_n be the integer represented by the binary string $1^{2^m}01^r0\text{list}(M_S)$, where $r := N^c m$ is an upper bound on the maximum bit length of entries of M, and $\text{list}(M_S)$ is the concatenation of length r binary representations of the entries of M_S (say left-to-right, top-to-bottom). Define $\ell(n)$ to be the bit length of A_n.

Due to space restriction the proof of the following lemma is omitted, and will appear in the full version. The proof involves using the DLOGTIME-uniform TC^0 circuits for iterated integer multiplication from [11], and the technique of scaling up the the counting hierarchy [7].

Lemma 5. *Given that $\gamma(n)$ is an unbounded monotone nondecreasing function such that for all n, $\gamma(n) < \log\log n$, we have that (A_n) is an integer sequence of bit size $\ell(n)$, where for all but finitely many n, $\ell(n) \leq n^{4\gamma(n)}$. Furthermore, (A_n) is weakly-definable in* CH/*poly.*

Next we apply a solver to (S). For this purpose, we let $F : \{0,1\}^* \to \{0,1\}^*$ be the following poly-time computable mapping: On input x of length $\tilde{n}$, try to parse $x = 1^{2^{\tilde{m}}}01^{\tilde{r}}0y$, for some integer $\tilde{r}, \tilde{m}$ and $y \in \{0,1\}^*$ with $|y| = 2^{2\tilde{m}}\tilde{r}$. If this fails, output 0. Otherwise, construct the $2^{\tilde{m}} \times 2^{\tilde{m}}$ matrix M whose (left-to-right, top-to-bottom) entries are given by consecutive r bit blocks of y. Then using standard tools, try to compute a nonzero integer $2^{\tilde{m}}$-vector c such that $Mc = 0$. Output c if this succeeds, 0 otherwise. We define α to be the absolute integer constant such that for all large enough $\tilde{n}$, F runs within time $\tilde{n}^\alpha$. This implies that for all large enough $\tilde{n}$, for any $x \in \{0,1\}^{\tilde{n}}$, $|F(x)| \leq \tilde{n}^\alpha$.

By Lemma 5, (A_n) is weakly-definable in CH/*poly*. Clearly, since $\gamma(n)$ is assumed to be a monotone growing function, for any constant $c' \geq 1$, it is satisfied that $\ell(n)^{c'/\gamma(\ell(n))} = n^{O(1)}$, as for all but finitely many n, $n \leq \ell(n) \leq n^{4\gamma(n)}$. Hence we can apply Lemma 4. We get that $F(A_n)$ is an integer sequence weakly-definable in CH/*poly*. We have that $F(A_n)$ encodes a 2^m-vector of integers. Indexing this vector by $e \in \{0,1\}^m$, we let $c_m(e)$ be the integer encoded by the eth entry, i.e. $c_m(e) = F(A_n)_e$, and let f_m be the polynomial given by fixing these integer coefficients in Equation (1). We have the following properties:

- $\forall^\infty n$, $c_m(e)$ has bit size at most $\ell(n)^\alpha \leq 2^{4\alpha\gamma(n)\log n} \leq 2^{4\alpha m}$.
- $L := \{(1^n, e, j, b) : e \in \{0,1\}^m, j \in \{0,1\}^{4\alpha m}, \text{ and } c_m(e)_j = b, \text{ where } m = \lceil \gamma(n)\log n \rceil\} \in$ CH/*poly*.

The proof of the following claim will be included in the full version:

Claim 1. *$f_m = \text{per}_{m'}(M')$ for $m' = 2^{o(m)}$, where M' is a matrix whose entries are in $\{x_1, x_2, \ldots, x_m\} \cup \{-1, -\frac{1}{2}, 0, \frac{1}{2}, 1\} \cup \{2^{2^0}, 2^{2^1}, \ldots, 2^{2^{4\alpha m}}\}$.*

To prove the above claim one proceeds similarly as in the proof of "Valiant's Criterion", cf. Proposition 2.11 in [7]. Recall we already observed that under our assumptions we have the collapse CH/*poly* $\subseteq$ nonuniform-TC^0. Hence $L \in$ nonuniform-TC^0. One can use the fact that nonuniform-$\text{TC}^0 \subseteq$ nonuniform-NC^1,

so we get $O(\log n) = o(m)$ depth formulas computing bits of the coefficients $c_m(e)$. Then by arithmetizing these Boolean formulas and employing standard arithmetic techniques the claim follows.

Let us observe that this completes the proof of Theorem 2. Consider $f'_m = 2^{m'} f_m$. Observe that f'_m is a nonzero polynomial in $m << N$ variables that vanishes on $\mathcal{H}_N$. Hence we have that

Fact 1. *f'_m has no size N depth d arithmetic circuits using constants $\in \Gamma_N$.*

By Claim 1, we have that $f'_m = \text{per}_{m'}(2M')$. Our assumptions imply that we have a size $s := (m')^{c_0} = 2^{o(m)}$ circuits of depth $d-1$ for $\text{per}_{m'}$ that use constants from Γ_s. We can put together $2M'$ using $O((m')^2) = 2^{o(m)}$ circuitry of depth one, while only using constants in $\Gamma_{4\alpha m}$. Hence we obtain $s' := 2^{o(m)}$ size circuits for f'_m of depth d that use constants from $\Gamma_{4\alpha m} \cup \Gamma_s$. Recalling that $N = \lfloor 2^{\epsilon\gamma(n)\log n} \rfloor$ and $m = \lceil \gamma(n) \log n \rceil$, we have that for all but finitely many n, $\max(4\alpha m, s) < N$ and $s' < N$. We have arrived at a contradiction with Fact 1. □

5 Proving a Weak Derandomization Hypothesis Unconditionally

We don't know how to prove Working Hypothesis 1, but as we will see in Section 6, neither do we need to for obtaining the sought after lower bound for the permanent as in the conclusion of Theorem 2! What we *can* establish is the following theorem. Note that there is no restriction to constant depth.

Theorem 3. *There exists an integer sequence $a_n(i)$ of polynomial bit size with $0 \leq i < n$, such that $a_n(i)$ is weakly-definable in the polynomial hierarchy, and for which the following holds:*

- *For any arithmetic circuit Φ of size n over $m \leq n$ variables and constants in $\Gamma_n = \{2^{2^0}, 2^{2^1}, \ldots, 2^{2^n}\} \cup \{-1, 0, 1\}$, if $\Phi(x_1, x_2, \ldots, x_m)$ computes a nonzero polynomial, then $\Phi(a_n(0), a_n(1), \ldots, a_n(m-1)) \neq 0$.*

Proof. Let $\mathcal{C}_n$ be the set of all arithmetic circuits of size n over $m \leq n$ variables using constants in Γ_n. For some constant $c > 0$, we can bound $|\mathcal{C}_n| \leq 2^{cn^2}$, provided n is large enough. Note that circuits in $\mathcal{C}_n$ can compute polynomials with degree at most 2^n. Let $S = \{1, 2, \ldots, 2^{n^3+n}\}$. If we pick $s_1, \ldots, s_n$ independently and uniformly at random from S, then by Lemma 1, for any nonzero polynomial f computed by a circuit in $\mathcal{C}_n$, $\Pr[f(s_1, s_2, \ldots, s_n) = 0] \leq 2^n / 2^{n^3+n} = 2^{-n^3}$. Hence by the union bound, $\Pr[\exists \text{ nonzero } f \text{ computed by a circuit} \in \mathcal{C}_n, f(s_1, s_2, \ldots, s_n) = 0] \leq 2^{cn^2 - n^3} < 1$. This means there exist at least one $\boldsymbol{s} \in S^n$, such that for any nonzero polynomial f computed by a circuit from $\mathcal{C}_n$, $f(s_1, s_2, \ldots, s_n) \neq 0$. For $\boldsymbol{s} = (s_1, s_2, \ldots, s_n) \in S^n$ define the predicate $\mathcal{P}(\boldsymbol{s})$ by

$$\forall \Psi \in \mathcal{C}_n, (\text{if } \Psi(s_1, s_2, \ldots, s_n) = 0, \text{ then } \forall \boldsymbol{t} \in S^n, \Psi(t_1, t_1, \ldots, t_n) = 0). \quad (2)$$

Observe that if $\mathcal{P}(\boldsymbol{s})$ is true, then for any *nonzero* f computed by a circuit from $\mathcal{C}_n$, $f(s_1, s_2, \ldots, s_n) \neq 0$. Also observe that for both universal quantifiers in (2), they range over sets whose elements can be described by $poly(n)$ size strings, assuming encodings of circuits using type names of size $O(\log n)$ as defined in Section 3. Hence if we can argue that given $\Psi \in \mathcal{C}_n$ and $\boldsymbol{u} \in S^n$, we can decide whether $\Psi(u_1, u_2, \ldots, u_n) \stackrel{?}{=} 0$ in PH, then the predicate $\mathcal{P}(\boldsymbol{s})$ is PH-decidable. Consequently, we can define a_n to be the lexicographically least element $\boldsymbol{s}$ of S^n such that $\mathcal{P}(\boldsymbol{s})$ holds. This makes a_n computable in P^{PH} by binary search. This implies that a_n is weakly-definable in PH.

We complete the proof by showing that given $\Psi \in \mathcal{C}_n$ and $\boldsymbol{u} \in S^n$, we can decide whether $\Psi(u_1, u_2, \ldots, u_n) \stackrel{?}{=} 0$ in PH. The only problem that may arise when evaluating $\Psi(u_1, u_2, \ldots, u_n)$ is that intermediate values get too large. For this we employ the idea of Ibarra and Moran [12] by evaluating modulo a prime number in some large range. Namely, consider primes p in the range $\{1, 2, \ldots, 2^{n^2}\}$. Each such prime takes n^2 many bits. We can evaluate $\Psi(u_1, u_2, \ldots, u_n) \bmod p$ in polynomial time[8]. By the Prime Number Theorem, the number of primes in this range is at least $\frac{2^{n^2}}{\log 2^{n^2}} > 2^{2n}$, provided n is large enough. By a simple induction, it follows that the value $|\Psi(u_1, u_2, \ldots, u_n)|$ can be bounded by $2^{2^{2n}}$. Hence it cannot be that all primes $\leq 2^{n^2}$ divide $\Psi(u_1, u_2, \ldots, u_n)$, if $\Psi(u_1, u_2, \ldots, u_n) \neq 0$. In other words, $\Psi(u_1, u_2, \ldots, u_n) = 0$ iff $\forall m \in \{1, 2, \ldots, 2^{n^2}\}$, if m is prime, then $\Psi(u_1, u_2, \ldots, u_n) \bmod p = 0$. Agrawal, Kayal and Saxena [2] proved primality testing is in polynomial time. Hence we get that the above is a Π_1P-predicate. □

6 Proof of Theorem 1

It is sufficient to show the following claim:

Claim 2. *For any unbounded monotone function $\gamma(n) = o(\log \log n)$, there does not exist a polynomial $p(n)$ such that $\forall^\infty n$, per_n has size $p(n)$ depth d arithmetic circuits using constants in the set $\Gamma_{p(n)} = \{-1, 0, 1\} \cup \{2^{2^0}, 2^{2^1}, \ldots, 2^{2^{p(n)}}\}$, where in addition these circuits are $n^{1/\gamma(n)}$-succinct.*

Proof. We argue by contradiction. Let $\gamma(n)$ be as in the claim, and suppose that for some polynomial $p(n)$, for all large enough n, per_n has size $p(n)$ depth d arithmetic circuits with constants in $\Gamma_{p(n)}$, where these circuits are $n^{1/\gamma(n)}$-succinct.

By Lemma 2, we get that we can compute $\mathrm{per}_n(M)$ over $\mathbb{Z}$, where entries of M are in $\{0, 1\}$ by $2n^{1/\gamma(n)}$-succinct TC^0 circuits. By Toda's Theorem and Valiant's completeness result for per_n, any $L \in \mathrm{PH}$ can be decided in polynomial time with a single query to the 0, 1-permanent. We can think of this as a three stage process:

[8] As a technical detail, we remark that for large numbers like 2^{2^n}, which appear as a $O(\log n)$ type name in the encoding of Ψ, we can compute $(2^{2^n} \bmod p)$ by performing n repeated squarings computed modulo p.

first apply a function $Q \in \mathrm{FP}$ to the input x, then obtain $\mathrm{per}(Q(x))$, finally apply $R \in \mathrm{FP}$ to produce the output $R(\mathrm{per}(Q(x))$. Since $\mathrm{FP} \subseteq \#\mathrm{P}$, Lemma 3 implies that for some constant b, we obtain $n^{b/\gamma(n)}$-succinct TC^0-circuits for Q and R. Combining all three levels of TC^0 circuits yields TC^0-circuits for L, where for some constant c depending on L, this family is $n^{c/\gamma(n)}$-succinct. The constant c can be picked larger than b to accommodate for the increase in size of joining the three representations. Wlog. assume $c > 1$.

In particular, for the integer sequence $a_n(i)$ of Theorem 3, we get that $uBit(a_n)$ can be decided by $n^{c/\gamma(n)}$-succinct TC^0 circuits, for some constant $c > 1$. This means that Working Hypothesis 1 is satisfied regardless of the depth (so certainly for depth $d + 1$), and for $\gamma'(n) := c^{-1}\gamma(n)$ instead of $\gamma(n)$. Since Theorem 2 holds for any unbounded monotone growing function $\gamma(n)$ where $\forall^\infty \gamma(n) < \log\log n$, we can apply Theorem 2 with $\gamma'(n)$ instead, to yield that there does not exist a polynomial $q(n)$ such that, for all large enough n, per_n has size $q(n)$ depth d circuits using constants in the set $\Gamma_{q(n)} = \{-1, 0, 1\} \cup \{2^{2^0}, 2^{2^1}, \ldots, 2^{2^{q(n)}}\}$, where in addition these circuits are $n^{1/\gamma'(n)}$-succinct. Since $1/\gamma'(n) > 1/\gamma(n)$, this contradicts the assumption that for all large enough n, per_n has size $p(n)$ depth d arithmetic circuits using constants in the set $\Gamma_{p(n)} = \{-1, 0, 1\} \cup \{2^{2^0}, 2^{2^1}, \ldots, 2^{2^{p(n)}}\}$, where in addition these circuits are $n^{1/\gamma(n)}$-succinct. □

Acknowledgments. We thank the anonymous referees for their valuable comments.

References

1. Agrawal, M.: Proving lower bounds via pseudo-random generators. In: Sarukkai, S., Sen, S. (eds.) FSTTCS 2005. LNCS, vol. 3821, pp. 92–105. Springer, Heidelberg (2005)
2. Agrawal, M., Kayal, N., Saxena, N.: Primes is in P. Ann. of Math 2, 781–793 (2002)
3. Agrawal, M., Vinay, V.: Arithmetic circuits: A chasm at depth four. In: Proc. 49th Annual IEEE Symposium on Foundations of Computer Science, pp. 67–75 (2008)
4. Allender, E.: The permanent requires large uniform threshold circuits. Chicago Journal of Theoretical Computer Science, article 7 (1999)
5. Barrington, D.M., Immerman, N., Straubing, H.: On uniformity within NC^1. J. Comp. Sys. Sci. 41, 274–306 (1990)
6. Baur, W., Strassen, V.: The complexity of partial derivatives. Theor. Comp. Sci. 22, 317–330 (1982)
7. Bürgisser, P.: On defining integers and proving arithmetic circuit lower bounds. Computational Complexity 18, 81–103 (2009)
8. DeMillo, R., Lipton, R.: A probabilistic remark on algebraic program testing. Inf. Proc. Lett. 7, 193–195 (1978)
9. Grigoriev, D., Karpinski, M.: An exponential lower bound for depth 3 arithmetic circuits. In: Proc. 13th Annual ACM Symposium on the Theory of Computing, pp. 577–582 (1998)
10. Heintz, J., Schnorr, C.: Testing polynomials which are easy to compute (extended abstract). In: Proc. 12th Annual ACM Symposium on the Theory of Computing, pp. 262–272 (1980)

11. Hesse, W., Allender, E., Barrington, D.: Uniform constant-depth threshold circuits for division and iterated multiplication. J. Comp. Sys. Sci. 64(4), 695–716 (2001)
12. Ibarra, O., Moran, S.: Probabilistic algorithms for deciding equivalence of straight-line programs. J. Assn. Comp. Mach. 30, 217–228 (1983)
13. Jansen, M., Santhanam, R.: Marginal hitting sets imply super-polynomial lower bounds for permanent (2011) (manuscript)
14. Kabanets, V., Impagliazzo, R.: Derandomizing polynomial identity testing means proving circuit lower bounds. Computational Complexity 13(1-2), 1–44 (2004)
15. Koiran, P.: Shallow circuits with high powered inputs. In: Proc. 2nd Symp. on Innovations in Computer Science (2010)
16. Koiran, P., Perifel, S.: Interpolation in Valiant's theory (2007) (to appear)
17. Koiran, P., Perifel, S.: A superpolynomial lower bound on the size of uniform non-constant-depth threshold circuits for the permanent. In: Proc. 24th Annual IEEE Conference on Computational Complexity (2009)
18. Raz, R.: Elusive functions and lower bounds for arithmetic circuits. Theory of Computing 6 (2010)
19. Raz, R., Yehudayoff, A.: Lower bounds and separations for constant depth multilinear circuits. Computational Complexity 18(2) (2009)
20. Schwartz, J.: Fast probabilistic algorithms for polynomial identities. J. Assn. Comp. Mach. 27, 701–717 (1980)
21. Shpilka, A., Wigderson, A.: Depth-3 arithmetic formulae over fields of characteristic zero. Journal of Computational Complexity 10(1), 1–27 (2001)
22. Toda, S.: PP is as hard as the polynomial-time hierarchy. SIAM J. Comput. 20, 865–877 (1991)
23. Torán, J.: Complexity classes defined by counting quantifiers. J. Assn. Comp. Mach. 38(3), 753–774 (1991)
24. Valiant, L.: The complexity of computing the permanent. Theor. Comp. Sci. 8, 189–201 (1979)
25. Vollmer, H.: Introduction to Circuit Complexity. A uniform approach. Springer, Heidelberg (1999)
26. Wagner, K.: The complexity of combinatorial problems with succinct input representation. Acta Informatica 23, 325–356 (1986)
27. Zankó, V.: #P-completeness via many-one reductions. International Journal of Foundations of Computer Science 2, 77–82 (1991)
28. Zippel, R.: Probabilistic algorithms for sparse polynomials. In: Ng, K.W. (ed.) EUROSAM 1979 and ISSAC 1979. LNCS, vol. 72, pp. 216–226. Springer, Heidelberg (1979)

On the Power of Algebraic Branching Programs of Width Two

Eric Allender and Fengming Wang

Department of Computer Science, Rutgers University, Piscataway, NJ 08855, USA
{allender,fengming}@cs.rutgers.edu

Abstract. We show that there are families of polynomials having small depth-two arithmetic circuits that cannot be expressed by algebraic branching programs of width two. This clarifies the complexity of the problem of computing the product of a sequence of two-by-two matrices, which arises in several settings.

1 Introduction

The n^{th} Iterated Matrix Multiplication polynomial of degree d, denoted $\mathrm{IMM}_{d,n}$ is the multilinear polynomial with d^2n variables that is the result of multiplying n d-by-d matrices of indeterminates. This family plays a central role in the study of algebraic complexity. Ben-Or and Cleve showed that $\mathrm{IMM}_{3,n}$ is complete (under projections) for the class of polynomials that can be expressed by arithmetic formulae of polynomial size [5,6]. This class is sometimes denoted VNC^1 [12] (as the analog of the Boolean class NC^1 in the setting of algebraic complexity initiated by Valiant [18]) and is also sometimes denoted VP_e (corresponding to the subclass of Valiant's class VP of polynomials of polynomial degree that have arithmetic circuits of polynomial size, where we restrict the circuits to be *expressions*).

It is natural to wonder if Ben-Or and Cleve's construction is optimal, in terms of dimension. That is: What can one say about $\mathrm{IMM}_{2,n}$?

There are some indications that $\mathrm{IMM}_{2,n}$ should be nearly as powerful as $\mathrm{IMM}_{3,n}$. For instance, Ben-Or and Cleve's completeness argument proceeds by showing that arithmetic formulae can be efficiently evaluated by a restricted type of straight-line program with three registers (and this translates into an implementation with 3-by-3 matrices). In the original conference publication of their results [5], Ben-Or and Cleve credit Coppersmith with the observation that if the underlying ring is commutative and has an element $\frac{1}{2}$ such that $\frac{1}{2} + \frac{1}{2} = 1$, then in fact *two* registers suffice to evaluate any arithmetic formula (albeit via straight-line programs that do not immediately lend themselves to implementation as $\mathrm{IMM}_{2,n}$ computations).

Perhaps the first study of the complexity of evaluating $\mathrm{IMM}_{2,n}$ arose in the work of Lipton and Zalcstein [11], who (in modern terminology) showed that the word problem over the free group with two generators (also known as the two-sided Dyck language) is AC^0-reducible to the problem of determining if a product of n two-by-two integer matrices evaluates to the identity matrix. Since the two-sided Dyck language is hard for NC^1 [15], this gives a lower bound on the complexity of evaluating $\mathrm{IMM}_{2,n}$ instances.

This lower bound is rather close to the best known upper bound. The problem of evaluating integer instances of $\mathrm{IMM}_{3,n}$ is complete for the Boolean complexity class

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 736–747, 2011.

GapNC^1 [7] (consisting of functions that have *arithmetic* circuits of polynomial size and logarithmic depth), and every problem in this latter class has *Boolean* circuits of polynomial size, bounded-fan-in, and depth $O(\log n \log^* n)$ [10]. The closeness of these bounds has led some researchers to wonder whether the classes of functions in NC^1 and GapNC^1 are in fact equal [2], in which case IMM$_{2,n}$ and IMM$_{3,n}$ would be interreducible under AC^0 reductions.

The NC^1-hardness of IMM$_{2,n}$ over the integers holds even for restricted cases of the problem. In [3], it is asserted that counting paths in planar width-two graphs (a restricted case of IMM$_{2,n}$ over the integers) is hard for NC^1 under ACC^0 reductions. (An error in the proof of this claim in [3] has been identified and corrected [13].)

On the other hand, there have also been indications that IMM$_{2,n}$ should be weaker than IMM$_{3,n}$. Ben-Or and Cleve point out that problems over GF(2) having what they called "LBS" straight-line programs (i.e., restricted straight-line programs which they used as a tool in presenting their completeness result) that use only two registers translate into permutation branching programs of width three [6], which Barrington showed require exponential size in order to compute the AND function [4]. However, this does not strictly rule out more general computations over IMM$_{2,n}$.

The AC^0 reductions from problems in NC^1 to IMM$_{2,n}$ are not projections, which are the usual type of reductions that are used in studying algebraic complexity classes. To illustrate the difference, consider the class GapAC^0: functions computed by polynomial-size constant-depth arithmetic circuits over the integers, where the input variables take only Boolean inputs. GapAC$^0 \subseteq$ TC$^0 \subseteq$ NC^1 [1], and hence any bit of any function $f \in$ GapAC^0 can be computed by an AC^0 reduction to the problem of multiplying a sequence of 2-by-2 integer matrices. However, any such function f can also be viewed as a polynomial $f(x_1, \ldots, x_n)$ in its input variables, and the AC^0 reduction does not allows us to obtain f from IMM$_{2,n^k}$ by substituting field elements and the variables $x_1, \ldots, x_n$ for the variables of IMM, even though this is possible for IMM$_{3,n^k}$. It follows from our main result that, even for fairly simple functions $f \in$ GapAC^0, no such reduction is possible – even if we allow projections to arbitrarily large IMM instances, and even if we greatly enlarge the type of substitutions that are considered.

If we expand the notion of projection, to allow variables to be replaced by arbitrary linear expressions, then we obtain an alternative characterization of algebraic branching programs, which were introduced by Nisan in order to study the complexity of determinant and permanent computations in various settings [14].

Definition 1. *An* Algebraic Branching Program *over $\mathbb{F}$ is a layered DAG with a single source s and exactly one sink t. The layers are numbered as $0, 1, 2, \ldots, d$; let V_i denote the set of vertices in the ith layer. The source (the sink, respectively) is the unique vertex in V_0 (V_d, respectively). Edges exist only between vertices in adjacent layers (i.e., each edge (a, b) has $a \in V_i$ and $b \in V_{i+1}$ for some $0 \leq i < d$). Each edge e is associated with a linear function l_e over $\mathbb{F}$ in the variables $\{x_i \mid 1 \leq i \leq n\}$. Every directed path $p = e_1 e_2 \ldots e_k$ represents the product $f_p = \prod_{j=1}^{k} l_{e_j}$. The polynomial f_v represented by v is $\sum_{p \in P_{s,v}} f_p$, where $P_{s,v}$ is the set of paths from s to v. The output of the algebraic branching program is f_t. The* width *of the program is* $\max_i |V_i|$.

It follows from [6] that polynomial-size algebraic branching programs of width three (or of any constant width $w \geq 3$) characterize exactly the polynomials in VNC^1. Algebraic

branching programs of constant width have been studied by several authors; we cite some recent examples [9,8]. We show that width three is optimal; the expressive power of width two algebraic branching programs is severely limited.

Theorem 1. *Let $l(\overline{x})$ be an arbitrary linear function. $\forall k \geq 8$, the polynomial $f(\overline{x}) = \sum_{i=1}^{k} x_{2i-1}x_{2i} + l(\overline{x})$ can not be computed by algebraic branching programs of width two over any field $\mathbb{F}$. This implies that IMM$_{2,n}$ is not complete for VNC^1 under regular projections (defined in Section 2).*

Width-two algebraic branching programs were also explored by Saha, Saptharishi and Saxena [16]. They considered "degree-restricted" algebraic branching programs (meaning that, if the output polynomial has degree n, then no intermediate polynomial in the branching program has degree greater than n). Their Theorem 16 shows that degree-restricted width-two algebraic branching programs compute polynomials only if they belong to an ideal generated by at most five linear forms (and thus they cannot compute the polynomial f in our Theorem 1 [17]). We do not know whether width-two algebraic branching programs can be simulated by width-two degree-restricted branching programs. Thus our Theorem 1 is incomparable with [16, Theorem 16]; their result applies to a larger class of polynomials, but relies on the degree restriction.

The remaining part of the paper is organized as follows: Section 2 provides the formal definitions and terminology that we use. In Section 3, we study homogeneous projections (defined in Section 2) of IMM$_{2,n}$ and prove a structural theorem for this type of computation as well as an impossibility result. Finally, we extend these results to more general settings in Section 4. Many proofs are omitted, due to lack of space.

2 Preliminaries

Let the underlying field be $\mathbb{F}$. Let $q(\overline{x}) \in \mathbb{F}[\overline{x}]$ be a multivariate polynomial over a set of variables $\overline{x}$. A projection p on $q(\overline{x})$ is an operation to generate new polynomials; a projection is described by a set of assignments $\{x_i \leftarrow v_i\}$, where the values v_i come from a particular set (to be specified later), and each variable $x_i \in \overline{x}$ appears at most once on the left-hand-side of a rule in p; furthermore, variables on the left-hand-side never occur on the right-hand-side. We get the new instance $q(\overline{x})|_p$ by replacing all occurrences of x_i in $q(\overline{x})$ with its counterpart v_i and leaving untouched those variables that are not in p. In this way, we say that $q(\overline{x})|_p$ *is obtained from* $q(\overline{x})$ *under the projection* p.

Let $\mathbb{H}$ be the set of homogeneous linear terms $\{c \cdot x_i \mid c \in \mathbb{F}^*, i \in \mathbb{N}\}$ where $\mathbb{F}^*$ is the set of units (i.e., non-zero elements). Let $\mathbb{L}$ be the set of general linear terms $\{\sum_{i=1}^{n} c_i \cdot x_i + w \mid n \in \mathbb{N}, c_i, w \in \mathbb{F}\}$. We define a projection $p = \{x_i \leftarrow v_i\}$ to be a *homogeneous projection* if $\forall i, v_i \in \mathbb{H} \cup \mathbb{F}$. If $\forall i, v_i \in \mathbb{L}$, then p is a *regular projection*. We mention that the most restrictive of these three types of projections, homogeneous projections, are the usual types of projections studied in algebraic complexity [18,6].

Consider n square matrices of dimension two $m_1, m_2, \ldots, m_n$, the entries of which are distinct variables. The $(1,1)$-entry of their product $\prod_{i=1}^{n} m_i$ is a multi-linear polynomial, denoted as IMM$_{2,n}$, which is called the *nth iterated matrix multiplication polynomial of dimension two*. The matrix $m_i|_p$ is obtained from m_i under the projection p, which means that the entries of m_i are substituted by the corresponding values in

p. Given a polynomial $f(\overline{x})$, it is easy to see that $f(\overline{x})$ is obtained from $\mathrm{IMM}_{2,n}$ under some projection p if and only if $f(\overline{x})$ is the $(1,1)$-entry of $\prod_{i=1}^{n} m_i|_p$, and moreover, the variables appearing in $m_i|_p$ belong to the set $\{x_j \mid x_j \text{ occurs in } f(\overline{x})\}$. Note that $f(\overline{x})$ is computable by some algebraic branching program of width two if and only if there exists $n \in \mathbb{N}$ such that $f(\overline{x})$ can be obtained from $\mathrm{IMM}_{2,n}$ under regular projections.

Let M be a set of square matrices of dimension two. We say a polynomial $f(\overline{x})$ is *computable by* M if there is an integer n and a projection p such that $f(\overline{x}) = \mathrm{IMM}_{2,n}|_p$ and furthermore, $\forall i \leq n,\ m_i|_p \in M$.

Let $\mathbb{H}_{2\times 2}$ ($\mathbb{R}_{2\times 2}$) denote the set of square matrices of dimension two with entries from $\mathbb{H} \cup \mathbb{F}$ ($\mathbb{L}$, respectively). Obviously, $\mathbb{H}_{2\times 2} \subseteq \mathbb{R}_{2\times 2}$.

We divide all square matrices of dimension two whose entries belong to $\mathbb{L}$ into three groups, Indg, Idg and Pdg. The matrices in Indg are called *inherently non-degenerate matrices* and their determinants evaluate to a fixed element in $\mathbb{F}^*$ while Idg consists of *inherently degenerate matrices* with zero determinants. $\mathrm{Pdg} = \mathbb{R}_{2\times 2} \setminus (\mathrm{Indg} \cup \mathrm{Idg})$ is the set of *potentially degenerate matrices*. Obviously the determinants of matrices in Pdg are nonzero polynomials of degree at least one.

We need some facts about the simple degree-two polynomials we study:

Fact 2. *Over any field $\mathbb{F}$, $x_1x_2 + x_3x_4$ is an irreducible polynomial.*

Fact 3. *Let $\mathbb{F}$ be any field, $k \geq 2$ and let $l(\overline{x})$ be an arbitrary linear function. Then $\Sigma_{i=1}^{k} x_{2i-1}x_{2i} + l(\overline{x})$ is an irreducible polynomial, and furthermore, its degree-two homogeneous part is irreducible as well.*

We group the variables x_{2i-1} and x_{2i} together, and call each the other's *partner variable*. It is convenient to consider a restricted class of projections:

Definition 2. *A regular projection $\{x_i \leftarrow v_i\}$ is* well-formed *if a variable appears on the left-hand-side iff its partner variable does, and $\forall i \in \mathbb{N}, \{v_{2i-1}, v_{2i}\} \cap \mathbb{F} \neq \emptyset$.*

Fact 4. *Let $k < n$ be two natural numbers. Consider the polynomial $\Sigma_{i=1}^{n} x_{2i-1}x_{2i}$. Then under any well-formed regular projection p of size $2k$, $f(\overline{x})|_p = \Sigma_{i=1}^{n-k} x_{2i-1}x_{2i} + l(\overline{x})$ (up to re-numbering the variables) is also an irreducible polynomial, where $l(\overline{x})$ is a linear function. Furthermore, its degree-two homogeneous part is also irreducible.*

Note that we may assume that the underlying field $\mathbb{F}$ is algebraically closed.

Fact 5. *Let $\mathbb{F}'$ be the algebraic closure of $\mathbb{F}$ and let M be a set of matrices. For any polynomial $f(\overline{x})$, if $f(\overline{x})$ is computable by M over $\mathbb{F}$, then it is computable by M over $\mathbb{F}'$ as well.*

3 $\mathrm{IMM}_{2,n}$ under Homogeneous Projections

Recall that $\mathbb{H}_{2\times 2}$ denotes the set of square matrices of dimension two with entries from $\mathbb{H} \cup \mathbb{F}$. We will show that it causes no loss of computational power, if we restrict the type of matrices that are used in $\mathbb{H}_{2\times 2}$ computations. First, however, it is very useful to observe that $\mathbb{H}_{2\times 2}$ corresponds exactly to a type of straight-line programs.

Let μ be a set of allowable straight-line program instructions (rules), and let R_i^t denote the contents of the register R_i at time t. A straight-line program P over the rule set μ using 2 registers (μ-SLP) is a sequence of pairs of instructions from μ, denoted as $\{(s_1^t, s_2^t) \mid 1 \leq t \leq |P|, (s_1^t, s_2^t) \in \mu\}$, where $|P|$ is the size of the program. P computes a function $p(\overline{x})$ in the natural way: Initially, $R_1^0 = 1$ and $R_2^0 = 0$. At the t-th step, R_i is updated according to the rule s_i^t. The final output $p(\overline{x})$ is stored as $R_1^{|P|}$. In this section, we consider only instructions that come from the set $\mu_{\mathbb{H}_{2\times 2}} = \{(R_1^{t+1} \leftarrow a \cdot R_1^t + b \cdot R_2^t,\ R_2^{t+1} \leftarrow a' \cdot R_1^t + b' \cdot R_2^t) \mid a, b, a', b' \in \mathbb{H} \cup \mathbb{F},\ t \in \mathbb{N}\}$. Under these assumptions, each R_i^t is a polynomial over the variables $\{x_j \mid j \in \mathbb{N}\}$. It is not hard to see that $\mu_{\mathbb{H}_{2\times 2}}$-SLPs and $\mathrm{IMM}_{2,n}$ under homogeneous projections compute the same set of polynomials. (Similar observations were made by Ben-Or and Cleve [6].) Furthermore, for any subset $N \subseteq \mathbb{H}_{2\times 2}$, there is a corresponding rule set $\mu_N \subseteq \mu_{\mathbb{H}_{2\times 2}}$ such that a polynomial $f(\overline{x})$ is computable by N if and only if there is a μ_N-SLP for it. Hence, given an arbitrary μ_N-SLP P, we may abuse the notations and identify the ith pair of instructions with its matrix representations m_P^i, which means that P can also be characterized by a sequence of matrices $\{m_P^i \mid 1 \leq i \leq |P|\}$.

3.1 Classification of $\mathbb{H}_{2\times 2} \cap$ Indg

We present a collection μ_N of rules that suffice to simulate any straight-line program using the rules $\mu_{\mathbb{H}_{2\times 2} \cap \mathrm{Indg}}$. Let $a, b, c, d \in \mathbb{F}^*$.

1. Transposition rule.

$$\begin{matrix} R_1^{t+1} \leftarrow R_2^t \\ R_2^{t+1} \leftarrow R_1^t \end{matrix} \text{ given by matrix } \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

2. Scalar rules.

$$\begin{matrix} R_1^{t+1} \leftarrow a \cdot R_1^t \\ R_2^{t+1} \leftarrow b \cdot R_2^t \end{matrix} \text{ given by matrix } \begin{bmatrix} a & 0 \\ 0 & b \end{bmatrix}$$

3. Offsetting rules of degree one.
   (a)

$$\begin{matrix} R_1^{t+1} \leftarrow a \cdot R_1^t + c \cdot x_i \cdot R_2^t \\ R_2^{t+1} \leftarrow b \cdot R_2^t \end{matrix} \text{ given by matrix } \begin{bmatrix} a & c \cdot x_i \\ 0 & b \end{bmatrix}$$

   (b)

$$\begin{matrix} R_1^{t+1} \leftarrow a \cdot R_1^t \\ R_2^{t+1} \leftarrow c \cdot x_i \cdot R_1^t + b \cdot R_2^t \end{matrix} \text{ given by matrix } \begin{bmatrix} a & 0 \\ c \cdot x_i & b \end{bmatrix}$$

4. Offsetting rules of degree zero.
   (a)

$$\begin{matrix} R_1^{t+1} \leftarrow a \cdot R_1^t + c \cdot R_2^t \\ R_2^{t+1} \leftarrow b \cdot R_2^t \end{matrix} \text{ given by matrix } \begin{bmatrix} a & c \\ 0 & b \end{bmatrix}$$

(b)

$$\begin{matrix} R_1^{t+1} \leftarrow a \cdot R_1^t \\ R_2^{t+1} \leftarrow c \cdot R_1^t + b \cdot R_2^t \end{matrix} \text{ given by matrix } \begin{bmatrix} a & 0 \\ c & b \end{bmatrix}$$

5. Other non-degenerate linear transformations.

$$\begin{matrix} R_1^{t+1} \leftarrow a \cdot R_1^t + c \cdot R_2^t \\ R_2^{t+1} \leftarrow d \cdot R_1^t + b \cdot R_2^t \end{matrix} \text{ given by matrix } \begin{bmatrix} a & c \\ d & b \end{bmatrix}$$

where $ab - cd \neq 0$.

Observation 6. *Any straight-line program using $\mu_{\mathbb{H}_{2\times 2}\cap Indg}$ can be simulated by a straight-line program using μ_N. That is, without loss of generality, one can assume that any straight-line program P has the following properties.*

- *If the transposition matrix is ever adopted by P, it is applied only once as the final pair of instructions.*
- *The following matrices need never appear, because they are transpositions of rules in μ_N (and transpositions introduced in this way can be eliminated).*

$$\begin{bmatrix} 0 & a \\ b & 0 \end{bmatrix}, \begin{bmatrix} 0 & a \\ b & c \end{bmatrix}, \begin{bmatrix} c & a \\ b & 0 \end{bmatrix}, \begin{bmatrix} 0 & a \\ b & c \cdot x_i \end{bmatrix}, \begin{bmatrix} c \cdot x_i & a \\ b & 0 \end{bmatrix}$$

- *This leaves only the rule set μ_N.*

3.2 Structure of $\mu_{\mathbb{H}_{2\times 2}\cap \mathbf{Indg}}$-SLPs and Its Implications

Definition 3. *Let $deg(f)$ denote the degree of the polynomial f. For any straight-line program P, let $deg(P,t) = deg(R_1^t) + deg(R_2^t)$ be* the degree of P at time t. *We call $deg(P,0), deg(P,1), \ldots, deg(P,|P|)$* the degree sequence of P.

An ordered pair of non-negative integers (t_1, t_2), where $t_1 + 1 < t_2$, is called a mesa *in the degree sequence of P if there exists $d > 0$ (the mesa's* height*) such that*

- *For all $t_1 < t' < t_2$, $deg(P,t') = d$;*
- *$deg(P,t_1) < d$;*
- *$deg(P,t_2) < d$.*

The operations in Observation 6 that simplify the straight-line program P do not change the height of any mesa in which the operations are applied.

Theorem 7. *If f is computable by $\mathbb{H}_{2\times 2} \cap Indg$, then there is a $\mu_{\mathbb{H}_{2\times 2}\cap Indg}$-SLP P for f with the property that there are no mesas in the degree sequence of P.*

Proof. By our assumption, there is some $\mu_{\mathbb{H}_{2\times 2}\cap \mathrm{Indg}}$-SLP P' computing f. If P' does not contain any mesas in its degree sequence, then we are done. Otherwise, we will show how to obtain P from P' by a series of transformations. At every step, we turn the current P' into an equivalent $\mu_{\mathbb{H}_{2\times 2}\cap \mathrm{Indg}}$-SLP while reducing the total height of all mesas by at least one. Ultimately we will obtain a $\mu_{\mathbb{H}_{2\times 2}\cap \mathrm{Indg}}$-SLP P with the desired property. Hence, it suffices to verify the correctness of a single step.

Let (t_1, t_2) be the first mesa in the current P' and d be its height. There are three cases to consider.

1. $\deg(R_1^{t_1+1}) > \deg(R_2^{t_1+1})$.
   We claim that the only instruction that can produce this outcome at time $t_1 + 1$ is the degree-one offsetting rule $3(a)$. Rule 2 is impossible since it only scales the registers by a constant factor respectively. Rule $3(b)$ implies that $\deg(R_1^{t_1+1}) = \deg(R_1^{t_1})$; there are two subcases to consider:
   - If $\deg(R_1^{t_1}) \geq \deg(R_2^{t_1})$, then $\deg(R_1^{t_1+1}) \leq \deg(R_2^{t_1+1})$, a contradiction to our assumption that $\deg(R_1^{t_1+1}) > \deg(R_2^{t_1+1})$.
   - If $\deg(R_1^{t_1}) < \deg(R_2^{t_1})$, then $\deg(R_2^{t_1+1}) \leq \deg(R_2^{t_1})$. This contradicts our assumption that $\deg(P, t_1) < \deg(P, t_1 + 1)$.

   For similar reasons, one can show that rules $4(a)$ and $4(b)$ are not applicable either. There are two cases that arise, in dealing with rule 5:
   - If $\deg(R_1^{t_1}) \neq \deg(R_2^{t_1})$, then under rule 5, $\deg(R_1^{t_1+1}) = \deg(R_2^{t_1+1})$, which contradicts our assumption that $\deg(R_1^{t_1+1}) > \deg(R_2^{t_1+1})$;
   - If $\deg(R_1^{t_1}) = \deg(R_2^{t_1})$, then $\deg(P', t_1) \geq \deg(P', t_1+1)$, which contradicts our assumption that (t_1, t_2) is a mesa.

   $\forall t_1 < t' < t_2$, $\deg(P', t') = d$ and $\deg(P', t_2) < d$ implies that rules $3(b)$, $4(b)$ and 5 are impossible at time t' (and at time t_2), since under our assumptions they would increase the degree of R_2 while maintaining the degree of R_1. Hence, for all $t_1 < t' \leq t_2$, the product $\prod_{i=t_1+1}^{t'} m_{P'}^i$ is an upper triangular matrix of the form $\begin{bmatrix} a & g_{t'} + w \\ 0 & b \end{bmatrix}$, where $w \in \mathbb{F}$, $a, b \in \mathbb{F}^*$ and $g_{t'}$ is a linear homogeneous polynomial. In other words, $R_1^{t'} = a \cdot R_1^{t_1} + (g_{t'} + w) \cdot R_2^{t_1}$ and $R_2^{t'} = b \cdot R_2^{t_1}$. Since $\deg(P', t_2) < d = \deg(P', t_1 + 1)$, it follows that $\deg(R_1^{t_1}) < \deg(R_1^{t_1+1})$ and $g_{t_2} = 0$. Thus, we can replace the whole computation between t_1 and t_2 by a simple application of rule 2 or $4(a)$ while avoiding the mesa (t_1, t_2).
2. $\deg(R_1^{t_1+1}) < \deg(R_2^{t_1+1})$.
   This is completely analogous to case 1.
3. $\deg(R_1^{t_1+1}) = \deg(R_2^{t_1+1})$.
   We argue that neither of rules $3(a)$ and $3(b)$ can happen at time $t_1 + 1$. We study the reasons for $3(a)$ and those for $3(b)$ are symmetric.
   - If $\deg(R_1^{t_1}) \leq \deg(R_2^{t_1})$, then $\deg(R_1^{t_1+1}) > \deg(R_2^{t_1+1})$, a contradiction to our assumption $\deg(R_1^{t_1+1}) = \deg(R_2^{t_1+1})$.
   - If $\deg(R_1^{t_1}) > \deg(R_2^{t_1})$, then $\deg(P', t_1 + 1) < \deg(P', t_1)$ since $\deg(R_2^{t_1}) = \deg(R_2^{t_1+1})$. This contradicts our assumption that (t_1, t_2) is a mesa.

   Furthermore, $\forall t_1 < t' < t_2$, $\deg(P', t') = d$ implies that rules $3(a)$ and $3(b)$ are impossible at time t' (and at time t_2). Thus, we obtain that for all $t_1 < t' \leq t_2$, the product $\prod_{i=t_1+1}^{t'} m_{P'}^i$ is a non-degenerate linear transformation, which can be captured by one of the other rules or their transposed counterparts. The analysis of this case can now be completed similarly to Case 1.

In all cases, we are able to reduce the total height of all mesas in P' by at least one, which concludes our proof. □

Corollary 1. *If a polynomial $f(\overline{x})$ is computable by $\mathbb{H}_{2\times 2} \cap$ Indg, then there exists a $\mu\mathbb{H}_{2\times 2}\cap$Indg-SLP for $f(\overline{x})$ with a monotonically nondecreasing degree sequence.*

The analysis in the proof of Theorem 7 also shows:

Fact 8. *For any $\mu_{\mathbb{H}_{2\times 2}\cap Indg}$-SLP P, $\forall 0 < t \leq |P|$, if $deg(P,t) > deg(P,t-1)$, then only one of the following two scenarios can happen.*

- *If either $3(a)$ or $3(b)$ is applied at time t, then $|deg(R_1^t) - deg(R_2^t)| = 1$.*
- *If the other rules are used at time t, then $deg(R_1^t) = deg(R_2^t)$.*

Lemma 1. *If P is a $\mu_{\mathbb{H}_{2\times 2}\cap Indg}$-SLP with a monotonically nondecreasing degree sequence, then for all $0 \leq t \leq |P|$, $|deg(R_1^t) - deg(R_2^t)| \leq 1$. Furthermore, we can assume that $deg(R_1^{|P|}) \geq deg(R_2^{|P|})$.*

Theorem 9. *Let $f(\overline{x})$ be a polynomial of total degree at least two whose highest-degree homogeneous part is irreducible. Then $f(\overline{x})$ is not computable by $\mathbb{H}_{2\times 2} \cap Indg$.*

Remark 1. The proof of Theorem 9 reveals that if $f(\overline{x})$ is computable by $\mathbb{H}_{2\times 2} \cap$ Indg, then the highest-degree homogeneous part of $f(\overline{x})$ can be completely factored into homogeneous linear polynomials.

3.3 Limitation of $\mu_{\mathbb{H}_{2\times 2}}$-SLPs

Note that we may assume that entries in inherently degenerate matrices all belong to $\mathbb{F}$, since they can be factored into a product of matrices, exactly one of which, denoted as m_1, belongs to Idg and furthermore, all m_1's entries are from $\mathbb{F}$.

Given a projection p and a straight-line program $P = \{m_P^i \mid 1 \leq i \leq |P|\}$ for a polynomial $f(\overline{x})$, we obtain the straight-line program $P|_p = \{m_P^i|_p \mid 1 \leq i \leq |P|\}$ for $f(\overline{x})|_p$. In the remaining part, all polynomials considered will be nonzero under any regular projection of size at most four.

Lemma 2. *There does not exist a matrix in P such that all of its entries belong to $\mathbb{H}$. This implies all matrices in P must contain an entry from $\mathbb{F}$.*

Lemma 3. *For any matrix $m \in \mathbb{H}_{2\times 2} \cap Pdg$, there exists a homogeneous projection p of size at most three such that $m|_p$ is degenerate and all of the entries in $m|_p$ belong to $\mathbb{F}$. Moreover, there is a well-formed homogeneous projection q of size at most six extending p.*

Definition 4. *We call such a well-formed homogeneous projection q as in Lemma 3 a* degenerating projection *for the potentially degenerate matrix m.*

Lemma 4. *Let $f(\overline{x})$ be a polynomial and P be one of its $\mu_{\mathbb{H}_{2\times 2}}$-SLPs. Suppose that there exists $0 < t \leq |P|$ such that m_P^t is a potentially degenerate matrix. Let p be one of its degenerating projections. Let $P' = P|_p$ and let $R_i^t(P')$ be the contents of R_i at time t in P'. Then up to the permutation of the indices, only one of the three following cases will happen.*

1. *$R_1^t(P') = R_2^t(P') = 0$ and $f(\overline{x})|_p = 0$. This is the uninteresting case and we ignore it in the remainder of the proof.*

2. $R_1^t(P') \in \mathbb{F}^*$ *and* $R_2^t(P') = w \cdot R_1^t(P')$ *for some* $w \in \mathbb{F}$.
3. $R_1^t(P')$ *is a polynomial of degree at least one,* $R_2^t(P') = w \cdot R_1^t(P')$ *for some* $w \in \mathbb{F}$, *and* $f(\overline{x})|_p$ *is divisible by* $R_1^t(P')$.

Corollary 2. *If* $f(\overline{x})|_p$ *is a nonzero irreducible polynomial and the other hypotheses of Lemma 4 hold, then* $R_1^t(P') = c \cdot f(\overline{x})|_p$ *for some* $c \in \mathbb{F}^*$.

Definition 5. *Let* $f(\overline{x})$, P, m_P^t *and* P' *satisfy the conditions of Lemma 4. If under the degenerating projection* p, *case* 2 *of Lemma 4 happens, then we call* p *a* cutting projection for m_P^t in P. *If instead we have case* 3, *then we call* p *a* finishing projection for m_P^t in P.

Observation 10. *Let* $f(\overline{x})$ *be a polynomial such that under any well-formed homogeneous projection* q *of size at most ten,* $f(\overline{x})|_q$ *is always a nonzero irreducible polynomial. Let* P *be a* $\mu_{\mathbb{H}_{2\times 2}}$*-SLP for* $f(\overline{x})$, *and let* m_P^t, p *and* P' *be the corresponding objects as in Lemma 4. We obtain a* $\mu_{\mathbb{H}_{2\times 2}}$*-SLP for* $f(\overline{x})|_p$ *from* P *as follows:*

- *If the projection* p *is a cutting projection for* m_P^t *in* P, *then we can simply ignore the instructions in* P' *before time* t *(including the* t*-th instruction), and concatenate a single instruction, which is a linear transformation from the initial condition* $(R_1^0, R_2^0) = (1, 0)$ *to the current status* $(R_1^t(P'), R_2^t(P'))$, *with the remaining segment of* P'. *This produces a* $\mu_{\mathbb{H}_{2\times 2}}$*-SLP of size at most* $|P| - t + 1$ *for* $f(\overline{x})|_p$.
- *If* p *is a finishing projection for* m_P^t *in* P, *then by Corollary 2,* $R_1^t(P')$ *is a nonzero multiple of* $f(\overline{x})|_p$ *and moreover,* $R_1^t(P') = a \cdot R_1^{t-1}(P') + b \cdot R_2^{t-1}(P')$ *where* $a, b \in \mathbb{H} \cup \mathbb{F}$. *We claim that* a *and* b *can not both belong to* $\mathbb{H}$. *Otherwise, let* $a = c \cdot x_i$ *and* $b = c' \cdot x_j$, *then* $p' = p \cup \{x_i = 0, x_j = 0\}$ *is a homogeneous projection of size no more than eight. Hence,* $f(\overline{x})|_{p'} \neq 0$ *while* $R_1^t(P')|_{x_i=0, x_j=0} = 0$, *a contradiction. The same observation implies that one of* a *and* b *must be a unit. Therefore, we can throw away the portion of* P' *after time* t *(including the* t*-th instruction) and generate* R_1*'s contents* $R_1^t(P')$ *by an offsetting matrix* m' *at time* t, *as follows: We have that* $R_1^t(P')$ *is some non-zero multiple of* $f(\overline{x})|_{p'}$, *say* $R_1^t(P') = s \cdot f(\overline{x})|_{p'}$. *We also have that* $R_1^t(P') = a \cdot R_1^{t-1}(P') + b \cdot R_2^{t-1}(P')$. *If* a *is a unit, then the desired output* $f(\overline{x})|_{p'}$ *is produced by the assignment* $R_1^t(P') \leftarrow (a/s) \cdot R_1^{t-1}(P') + (b/s) \cdot R_2^{t-1}(P')$, *which can be accomplished by a rule of type 3(a) or 4(a) (since we do not care what value is placed in* R_2*). If* b *is a unit, then the desired assignment instead is produced by a transposition of a rule of type 3(b) or 4(b). Thus, in either case, we obtain a* $\mu_{\mathbb{H}_{2\times 2}}$*-SLP of size at most* $t + 1$ *for* $f(\overline{x})|_p$.

Definition 6. *Let* $f(\overline{x})$ *be a polynomial and* P *be one of its* $\mu_{\mathbb{H}_{2\times 2}}$*-SLPs. We classify the potentially degenerate matrices* m_P^t *in* P *according to the following criterion: If* m_P^t *has at least one finishing projection, then* m_P^t *is* good*; Otherwise,* m_P^t *is* bad.

In the same spirit, we can classify inherently degenerate matrices in $\mu_{\mathbb{H}_{2\times 2}}$-SLPs. In this case, we can think that the degenerating projection is the empty set. The notions of badness and goodness also carry over; note that the notions of badness and goodness only apply to potentially and inherently degenerate matrices.

Observation 11. *Let p be an arbitrary homogeneous projection and let $P' = P|_p$. If m_P^t is bad in P, then $m_{P'}^t$ can not be good in P' (This is because, if m_P^t is bad, then under any extension of p, at time t both registers compute field elements which are constant polynomials with no variables). More precisely, $m_{P'}^t$ either stays as a bad matrix or becomes an inherently non-degenerate matrix. Furthermore, inherently non-degenerate matrices will never be turned into some other type by any projection.*

Theorem 12. *If $k \geq 8$, then $f(\overline{x}) = \sum_{i=1}^{k} x_{2i-1}x_{2i}$ is not computable by $\mathbb{H}_{2\times 2}$. That is, for every n, $f(\overline{x})$ can not be obtained from $IMM_{2,n}$ under homogeneous projections.*

Proof. We prove the theorem by contradiction. Suppose P is a $\mu_{\mathbb{H}_{2\times 2}}$-SLP for $f(\overline{x})$. We define the set G of time steps as:

$$G = \{t \mid m_P^t \text{ is a good matrix}\}.$$

There are two cases to consider.

The first case is that $G = \emptyset$. Define the set B similarly as:

$$B = \{t \mid m_P^t \text{ is a bad matrix}\}.$$

If B is empty as well, then P is indeed a $\mu_{\mathbb{H}_{2\times 2}\cap \text{Indg}}$-SLP. By Fact 3, the highest-degree homogeneous part of $f(\overline{x})$ is irreducible, and by Theorem 9, we have reached a contradiction. Otherwise, let $t_B = \max(B)$. Note that, by Fact 4, the output at time $|P|$ is a non-zero polynomial under any well-formed regular projection of size at most six, which means that $m_P^{|P|}$ cannot be bad, and hence $t_B < |P|$. Let p be one of the cutting projections of $m_P^{t_B}$. Consider $P|_p$ and the polynomial $f(\overline{x})|_p$ it computes. Since the size of p is bounded by six, by Fact 4, $f(\overline{x})|_p$ is again an irreducible polynomial and moreover, its degree-two homogeneous part is irreducible. For all t such that $t_B \leq t \leq |P|$, m_P^t is an inherently non-degenerate matrix. By the first item of Observation 10, we now have a $\mu_{\mathbb{H}_{2\times 2}\cap \text{Indg}}$-SLP for $f(\overline{x})|_p$ which is a contradiction to Theorem 9. Notice that by Fact 4, the above arguments apply to any polynomial of the form $\sum_{i=1}^{5} x_{2i-1}x_{2i} + l(\overline{x})$ where $l(\overline{x})$ is an arbitrary linear function.

We now assume that $G \neq \emptyset$. Let $t_G = \min G$. Suppose first that $m_P^{t_G}$ is an inherently degenerate matrix. Since $m_P^{t_G}$ is good, we have by Lemma 4 that, at time t_G, register R_1 computes a nonzero multiple of f. Hence by the second item of Observation 10 with $p = \emptyset$, we obtain a new $\mu_{\mathbb{H}_{2\times 2}}$-SLP for $f(\overline{x})$, consisting of only the matrices before t_G – none of which are good. This brings us back to the first case and a contradiction.

Otherwise, assume that $m_P^{t_G}$ is a potentially degenerate matrix and let p be one of its finishing projections of size at most six. Consider $P|_p$ and the polynomial $f(\overline{x})|_p$ it computes. By the second item of Observation 10, we obtain a new $\mu_{\mathbb{H}_{2\times 2}}$-SLP P' for $f(\overline{x})|_p$ and furthermore, by Observation 11, P' does not contain any good matrices. Hence, this reduces us to the first case, since $f(\overline{x})|_p$ is of the form $\sum_{i=1}^{5} x_{2i-1}x_{2i} + l(\overline{x})$ where $l(\overline{x})$ is an arbitrary linear function. It is not hard to see that we will arrive at a contradiction for $f(\overline{x})|_p$, which completes our proof. □

Corollary 3. *If $k \geq 8$, then $f(\overline{x}) = \sum_{i=1}^{k} x_{2i-1}x_{2i} + l(\overline{x})$ is not computable by $\mathbb{H}_{2\times 2}$, where $l(\overline{x})$ is an arbitrary linear function.*

4 Extensions to Regular Projections

In this section, we show that in the seemingly more powerful model, it is still hard to compute simple polynomials. The following lemma (whose proof is omitted due to lack of space) shows that, for nondegenerate matrices, the regular case reduces to the homogeneous case.

Lemma 5. *Every matrix m in $\mathbb{R}_{2\times 2} \cap$ Indg can be represented by a product of matrices in $\mathbb{H}_{2\times 2} \cap$ Indg.*

Theorem 13. *Let $f(\overline{x})$ be a polynomial whose highest-degree homogeneous part is irreducible. Then $f(\overline{x})$ is not computable by $\mathbb{R}_{2\times 2} \cap$ Indg.*

In the remaining part of this section, we show how to adapt the machinery in Section 3.3 to prove Theorem 1. As part of this adaptation, we use a measure of "independence" among the linear functions appearing in a matrix (and indeed, this is also used in the proof of Lemma 5). The following paragraph makes this precise:

Let $m \in \mathbb{R}_{2\times 2}$ be of the following form: $\begin{bmatrix} l_{1,1}+w_{1,1} & l_{1,2}+w_{1,2} \\ l_{2,1}+w_{2,1} & l_{2,2}+w_{2,2} \end{bmatrix}$. where $w_{i,j} \in \mathbb{F}$ and the $l_{i,j}$'s are homogeneous linear forms in $\{\sum_{k=1}^{n} c_k x_k \mid n \in \mathbb{N}, c_k \in \mathbb{F}\}$. We will pay attention to the rank of the subspace spanned by $\{l_{i,j} \mid i,j \in \{1,2\}\}$, denoted as $r(m)$, which in some sense characterizes the number of "independent variables" among the $l_{i,j}$'s.

Theorem 14 (Theorem 1 restated). *Let $l(\overline{x})$ be an arbitrary linear function. $\forall k \geq 8$, $f(\overline{x}) = \sum_{i=1}^{k} x_{2i-1}x_{2i} + l(\overline{x})$ is not computable by $\mathbb{R}_{2\times 2}$. That is, for any n, $f(\overline{x})$ can not be obtained from $IMM_{2,n}$ under regular projections.*

Proof (Proof sketch). The proof is by contradiction. Suppose P is a $\mu_{\mathbb{R}_{2\times 2}}$-SLP for $f(\overline{x})$. The following lemma plays the same role as Lemma 3 played in Section 3.3.

Lemma 6. *For any matrix $m \in \mathbb{R}_{2\times 2} \cap$ Pdg, there exists a regular projection p of size at most three such that $m|_p$ is degenerate. Moreover, there is a well-formed projection q of size at most six such that $m|_q$ is degenerate and all of the entries in $m|_p$ belong to $\mathbb{F}$.*

Define cutting and finishing projections in terms of well-formed regular degenerating projections as we did in Definition 5. Combined with variations of Lemma 4 and Observation 10, the remaining proof proceeds along similar lines as in Section 3.3. □

Acknowledgment

Discussions that the first author had with Meena Mahajan, Guillaume Malod, and Sylvain Perifel at the 2010 Dagstuhl Seminar 10481 on Computational Counting were very influential in helping us understand the limitations of width-two computations. We thank Luke Friedman, Chandan Saha, Ramprasad Saptharishi and anonymous referees for many helpful comments. We are grateful to Luke Friedman and Ramprasad Saptharishi for pointing out mistakes in earlier versions. Useful feedback was also provided by Maurice Jansen and Ran Raz. This work was supported by NSF Grants CCF-0830133 and CCF-0832787.

References

1. Agrawal, M., Allender, E., Datta, S.: On TC^0, AC^0, and arithmetic circuits. Journal of Computer and System Sciences 60, 395–421 (2000)
2. Allender, E.: Arithmetic circuits and counting complexity classes. In: Krajíček, J. (ed.) Complexity of Computations and Proofs. Quaderni di Matematica, vol. 13, pp. 33–72. Seconda Università di Napoli (2004)
3. Ambainis, A., Allender, E., Barrington, D.A.M., Datta, S., LêThanh, H.: Bounded depth arithmetic circuits: Counting and closure. In: Wiedermann, J., Van Emde Boas, P., Nielsen, M. (eds.) ICALP 1999. LNCS, vol. 1644, pp. 149–158. Springer, Heidelberg (1999)
4. Barrington, D.A.: Width-3 permutation branching programs. Technical Report Technical Memorandum MIT/LCS/TM-293, MIT (1985)
5. Ben-Or, M., Cleve, R.: Computing algebraic formulas using a constant number of registers. In: Proc. ACM Symp. on Theory of Computing (STOC), pp. 254–257 (1988)
6. Ben-Or, M., Cleve, R.: Computing algebraic formulas using a constant number of registers. SIAM Journal on Computing 21(1), 54–58 (1992)
7. Caussinus, H., McKenzie, P., Thérien, D., Vollmer, H.: Nondeterministic NC^1 computation. Journal of Computer and System Sciences 57, 200–212 (1998)
8. Jansen, M.J.: Lower bounds for syntactically multilinear algebraic branching programs. In: Ochmański, E., Tyszkiewicz, J. (eds.) MFCS 2008. LNCS, vol. 5162, pp. 407–418. Springer, Heidelberg (2008)
9. Jansen, M.J., Raghavendra Rao, B.V.: Simulation of arithmetical circuits by branching programs with preservation of constant width and syntactic multilinearity. In: Frid, A., Morozov, A., Rybalchenko, A., Wagner, K.W. (eds.) CSR 2009. LNCS, vol. 5675, pp. 179–190. Springer, Heidelberg (2009)
10. Jung, H.: Depth efficient transformations of arithmetic into Boolean circuits. In: Budach, L. (ed.) FCT 1985. LNCS, vol. 199, pp. 167–173. Springer, Heidelberg (1985)
11. Lipton, R., Zalcstein, Y.: Word problems solvable in logspace. Journal of the ACM 24, 522–526 (1977)
12. Mahajan, M., Raghavendra Rao, B.V.: Small-space analogues of valiant's classes. In: Kutyłowski, M., Charatonik, W., Gębala, M. (eds.) FCT 2009. LNCS, vol. 5699, pp. 250–261. Springer, Heidelberg (2009)
13. Mahajan, M., Saurabh, N., Sreenivasaiah, K.: Counting paths in planar width 2 branching programs (2011) (manuscript)
14. Nisan, N.: Lower bounds for non-commutative computation (extended abstract). In: Proc. ACM Symp. on Theory of Computing (STOC), pp. 410–418 (1991)
15. Robinson, D.: Parallel algorithms for group word problems. PhD thesis, Univ. of California, San Diego (1993)
16. Saha, C., Saptharishi, R., Saxena, N.: The power of depth 2 circuits over algebras. In: Proc. Conference on Foundations of Software Technology and Theoretical Computer Science (FST&TCS), pp. 371–382 (2009)
17. Saha, C.: Private communication (2011)
18. Valiant, L.: Completeness classes in algebra. In: Proc. ACM Symp. on Theory of Computing (STOC), pp. 249–261 (1979)

Primal-Dual Approximation Algorithms for Node-Weighted Steiner Forest on Planar Graphs

Carsten Moldenhauer

Humboldt-Universität zu Berlin, Institut für Informatik, D-10099 Berlin, Germany
carsten.moldenhauer@gmail.com

Abstract. NODE-WEIGHTED STEINER FOREST is the following problem: Given an undirected graph, a set of pairs of terminal vertices, a weight function on the vertices, find a minimum weight set of vertices that includes and connects each pair of terminals. We consider the restriction to planar graphs where the problem remains NP-complete. Demaine *et al.* [DHK09] showed that the generic primal-dual algorithm of Goemans and Williamson [GW97] is a 6-approximation on planar graphs. We present (1) a different analysis to prove an approximation factor of 3, (2) show that this bound is tight for the generic algorithm, and (3) show how the algorithm can be improved to yield a $9/4$-approximation algorithm.

We give a simple proof for the first result using contraction techniques and present an example for the lower bound. Then, we establish a connection to the feedback problems studied by Goemans and Williamson [GW98]. We show how our constructions can be combined with their proof techniques yielding the third result and an alternative, more involved, way of deriving the first result. The third result induces an upper bound on the integrality gap of $9/4$. Our analysis implies that improving this bound for NODE-WEIGHTED STEINER FOREST via the primal-dual algorithm is essentially as difficult as improving the integrality gap for the feedback problems in [GW98].

1 Introduction

We consider the following problem, called NODE-WEIGHTED STEINER FOREST (NWSF): Given an undirected graph, a set of pairs of terminal vertices, a weight function on the vertices, find a minimum weight set of vertices whose induced graph includes and connects each pair of terminals. This problem is a generalization of STEINER FOREST, where the weight function is defined on the edges and the task is to find a minimum weight set of edges. STEINER FOREST reduces to our problem by placing a new vertex on every edge and setting its weight to the edge weight. It follows that NWSF is NP-complete and remains so when restricted to planar graphs [GJ77].

Steiner problems are among the oldest NP-complete problems. In fact, the (edge-weighted) STEINER TREE problem, where all terminal pairs have one common terminal, is one of the 21 NP-complete problems from Karp's list [Kar72]. Besides numerous research in the past, Steiner problems have particularly received much attention in the last three years: STEINER TREE is known to be MAX-SNP hard on general graphs and Chlebík and Chlebíková [CC08] presented the current best lower bound of $96/95$

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 748–759, 2011.

in 2008. Further, Byrka *et al.* [BGRS10] recently achieved a 1.39-approximation algorithm. On planar graphs, Borradaile *et al.* [BKM09] gave a polynomial time approximation scheme for STEINER TREE that has been extended to polynomial time approximation schemes for STEINER FOREST [BHM10] and the prize-collecting variants [BCE+10]. In contrast, for node-weighted Steiner problems on planar graphs, the only known result is a 6-approximation algorithm [DHK09].

Node-weighted Steiner problems have various applications reaching from maintenance of electric power networks [GMNS99] to computational sustainability [DG10]. Particularly, they have been studied as subproblems in more constrained settings [MR07]. Due to an easy approximation preserving reduction from SET COVER, NWSF cannot be approximated up to $(1 - o(1)) \ln k$ in general graphs where k is the number of terminal pairs, unless NP admits slightly superpolynomial time algorithms [Fei98]. Hence, the node-weighted problem is much harder to approximate than the edge-weighted one. Klein and Ravi [KR95] presented an algorithm for NWSF which is implicitly a primal-dual algorithm and has an approximation guarantee of $2 \ln k$. Thus, their algorithm is optimal up to a constant factor on general graphs. However, on planar graphs substantial improvement is still possible.

Since most practical applications occur in planar or nearly planar environments, we focus on planar graphs. The only algorithm that has been previously applied to NWSF in this setting is the primal-dual algorithm of Goemans and Williamson [GW97]. Demaine *et al.* [DHK09] showed that this algorithm is a 6-approximation. Due to an edge counting argument their analysis also extends to graphs with a fixed forbidden minor. It is not known if polynomial time approximation schemes exist for planar graphs.

Our contributions

1. We improve the result of Demaine *et al.* [DHK09] by showing that the generic primal-dual algorithm has an approximation factor of 3 for NODE-WEIGHTED STEINER FOREST on planar graphs. We outline our proof in Section 3 using contraction techniques. Further, our analysis implies a reduction by a factor of $2/3$ in the constants in [DHK09, p. 337] for graphs with a fixed forbidden minor.

2. We show that the factor of 3 is best possible for the generic algorithm by presenting a worst-case example in Section 4.

3. The primal-dual algorithm is defined by an oracle. By improving this oracle we obtain an algorithm with approximation factor of $9/4$. This factor is best possible for this algorithm. Our analysis combines our constructions with the proof techniques of Goemans and Williamson for feedback problems on planar graphs [GW98]. Additionally, we derive an alternative, but more involved, proof of the factor of 3 for the generic algorithm. Further, our work implies strong similarities between the analysis of the primal-dual algorithm for the feedback problems studied in [GW98] and the analysis of the algorithm for NWSF. It is very likely that an improvement in the approximation factor in one of these problems via the primal-dual method will also carry over to the other problems. Moreover, we reduce FEEDBACK VERTEX SET to NWSF on undirected planar graphs.

We present the standard integer linear programming formulation for NWSF in Section 1.1. The integrality gap for this formulation is lower bounded by 2 and our

results imply an upper bound of $9/4$. Essentially, our analysis shows that improving this bound via the primal-dual method is as difficult as improving the bound of $9/4$ on the integrality gap for the feedback problems from [GW98] which is an open problem since 1996.

1.1 Preliminaries

Let $G = (V, E)$ be an undirected simple plane graph, *i.e.*, a planar graph with given embedding. Let $S \subseteq V$ be a set of vertices. We denote the induced subgraph of S in G by $G[S]$. The set of neighbors of S is denoted by $\Gamma(S) = \{v : u \in S, v \notin S, uv \in E\}$. The set of connected components of G is denoted by $\mathcal{C}(G)$. A ρ-approximation algorithm is an algorithm that runs in polynomial time and produces a feasible solution of weight at most ρ times the weight of an optimal solution.

Let $w : V \to \mathbb{R}^+$ be a weight function on the vertices. Let $\{(s_1, t_1), \ldots, (s_k, t_k)\}$ be a set of pairs of vertices, called *terminal pairs*. The problem NODE-WEIGHTED STEINER FOREST is to find a minimum weight set $X \subseteq V$ of vertices such that for all $1 \leq i \leq k$, s_i and t_i are in the same component of $G[X]$. Let $\mathcal{T}$ be the family of vertex sets that separate a terminal pair

$$\mathcal{T} = \{S \subseteq V : |\{s_i, t_i\} \cap S| = 1 \text{ for some } 1 \leq i \leq k\}.$$

Then, NODE-WEIGHTED STEINER FOREST is the problem

(NWSF)

$$\begin{aligned} &\text{minimize} && \sum_{v \in V} w(v)\, x_v && && (1)\\ &\text{subject to} && \sum_{v \in \Gamma(S)} x_v \geq 1 && \forall S \in \mathcal{T} && (2)\\ & && x_v \in \{0, 1\} && \forall v \in V. && (3) \end{aligned}$$

The constraints (2) ensure that for every set that separates terminals the solution has to select one of its neighbors. This guarantees connectedness of the terminal pairs.

We can assume w.l.o.g. that all terminals have zero weight. Otherwise, attaching a new dummy zero-weight vertex to each terminal and replacing the terminal vertex set by all dummy vertices gives an equivalent instance.

2 The Primal-Dual Algorithm

In this section, we will briefly describe the primal-dual method developed by Goemans and Williamson. For more information, we refer to their excellent survey [GW97].

Consider the LP relaxation of (NWSF) that replaces the integrality constraints (3) by $x_v \geq 0$. The dual of the relaxation is

(DNWSF)

$$\begin{aligned} &\text{maximize} && \sum_{S \in \mathcal{T}} y_S && && (4)\\ &\text{subject to} && \sum_{S \in \mathcal{T}:\, v \in \Gamma(S)} y_S \leq w(v) && \forall v \in V && (5)\\ & && y_S \geq 0 && \forall S \in \mathcal{T}. && (6) \end{aligned}$$

Intuitively, the dual packing constraints (5) ensure that the weight distributed among the sets adjacent to v is not more than the weight of v.

The algorithm simultaneously constructs a feasible solution X corresponding to (NWSF) and a feasible solution $\boldsymbol{y}$ to the dual of the LP relaxation (DNWSF), beginning with X equal to the set of all terminals and a dual solution $\boldsymbol{y} \equiv 0$. It is defined by an oracle which, given an infeasible solution X, selects a set of violated constraints in the primal program (NWSF). We denote this oracle by VIOL(X) and it returns the sets from $\mathcal{T}$ associated with the selected violated constraints (2) of X. While X is not a feasible solution, the algorithm simultaneously increases the dual variables on the sets returned by VIOL(X). When one of the dual packing constraints (5) becomes tight for some vertex $v \in V$, v is added to X. Observe that after v is added to X, no variable y_S occurring in v's packing constraint (5) will be increased again since the respective constraint (2) for S is satisfied. After a feasible solution is obtained, the algorithm performs a clean-up step. It considers all vertices in X in the reverse order in which they were added and removes any vertex that is not necessary for feasibility of X.

Primal-Dual Algorithm

(i) $X \leftarrow \{s_i : 1 \leq i \leq k\} \cup \{t_i : 1 \leq i \leq k\}$
(ii) $\boldsymbol{y} \leftarrow 0$.
(iii) While X is not feasible
  (a) Increase y_S uniformly for all $S \in \text{VIOL}(X)$ until $\exists S \in \text{VIOL}(X), v \in \Gamma(S) : \sum_{S' \in \mathcal{T}: v \in \Gamma(S')} y_{S'} = w(v)$.
  (b) $X \leftarrow X \cup \{v\}$.
(iv) For each $v \in X$ in the reverse order in which they were added in Step (iii):
  (a) If $X \setminus \{v\}$ is feasible then set $X \leftarrow X \setminus \{v\}$.
(v) Return X (and $\boldsymbol{y}$).

Note that the algorithm runs in polynomial time if the oracle VIOL can be computed in polynomial time and returns only polynomially many sets. Even though there might be exponentially many variables y_S, only polynomially many will be changed in the course of the algorithm. In fact, the algorithm can be implemented to encode the values of $\boldsymbol{y}$ implicitly. We refer the reader to [GW97, p. 164, Figure 4.5] for details. In the following sections we will present two different oracles that can easily be computed in polynomial time. In fact, for a given infeasible set X they each return a subset of the components $\mathcal{C}(X)$. Recall, that a solution X is feasible if and only if all terminals are selected and all its components do not separate any terminal pair. Thus, for our oracles, the while loop in the algorithm is executed until the oracle does not return any sets anymore.

The above algorithm differs slightly from the generic primal-dual algorithm because in (i) X is initialized with the set of all terminals instead of the empty set. Since we can assume that all terminals have zero weight (see above), this is no restriction. In [GW97] it is proved that the performance guarantee can be obtained using the theorem below. A *minimal augmentation* F of X is a feasible solution F containing X such that for any $v \in F \setminus X$, $F \setminus \{v\}$ is not feasible.

Theorem 1 ([GW97]). *Let* OPT *denote the weight of an optimal solution. Let* γ *satisfy that for any infeasible set* $X \subseteq V$ *containing all the terminals, and any minimal augmentation* F *of* X

$$\sum_{S \in \text{VIOL}(X)} |F \cap \Gamma(S)| \leq \gamma \, |\text{VIOL}(X)|.$$

Then, the primal-dual algorithm delivers a solution of weight at most $\gamma \sum_{S \in \mathcal{T}} y_S \leq \gamma$ OPT.

Therefore, given an oracle VIOL, showing that Theorem 1 holds for an appropriate choice of γ proves the algorithm to be a γ-approximation. For the here presented oracles we will show that the minimal values of γ are 3 and $9/4$.

3 A 3-Approximation on Planar Graphs

We now consider planar graphs and the oracle AVC that returns all components of the current solution that separate a terminal pair. Recall that a set S separates a terminal pair (s_i, t_i) if $|S \cap \{s_i, t_i\}| = 1$. We call a component that separates a terminal pair *violated*. Let X be the current solution in any iteration of the algorithm. The set of all violated components is $\text{AVC}(X) = \mathcal{C}(X) \cap \mathcal{T}$.

Note that $\text{AVC}(X)$ returns the inclusion-wise minimal sets from $\mathcal{T}$ for which the constraint (2) is violated: Let S be a set with violated constraint (2) for X. Let (s_i, t_i) be any terminal pair that is separated by S and w.l.o.g. $s_i \in S$. Due to the initialization in Step (i), every terminal is included in a component of X. Denote by C the component of X containing s_i. Clearly, $C \subseteq S$ because otherwise S has a neighbor in $C \setminus S \subseteq X$ and its respective constraint would not have been violated. Now, $C \in \text{AVC}(X)$ since C is a component of X and separates (s_i, t_i).

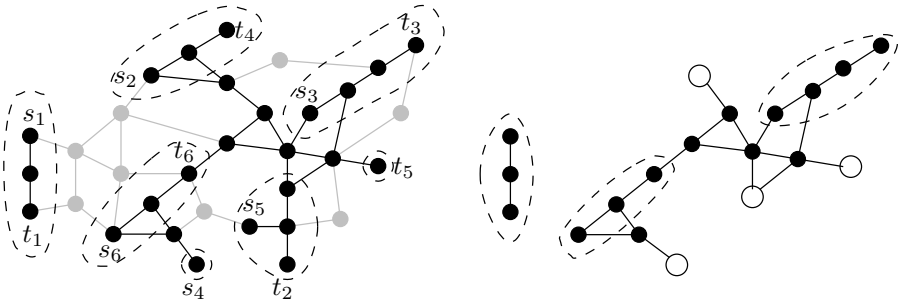
Clearly, AVC can be computed in polynomial time by checking each component of X if it separates two terminals. We will show that the corresponding value of γ in Theorem 1 is 3 and therefore the primal-dual algorithm is a 3-approximation.

Theorem 2. *Let* G *be planar,* X *be an infeasible solution containing all the terminals and* F *be any minimal feasible augmentation of* X*. Then,*

$$\sum_{S \in \mathit{AVC}(X)} |F \cap \Gamma(S)| \leq 3 \, |\mathit{AVC}(X)|. \tag{7}$$

Proof. The sum in (7) counts the number of adjacencies between F and all violated components of X, counting multiply if one violated component is adjacent to several vertices in F, but counting only once if multiple vertices in a common violated component are adjacent to one vertex in F.

Recall that contraction of an edge in a planar graph preserves planarity. Let $\overline{G}$ be the graph obtained from $G[F]$ by contracting each violated component of X to a single vertex and discarding multiple copies of edges (Figure 1). In the following, we will refer to vertices corresponding to violated components of X as *white* vertices. All other



(a) G in gray; $G[F]$ in black; $\mathcal{C}(X)$ marked with dashed lines.

(b) $\overline{G}$ with non-violated components of X marked with dashed lines.

Fig. 1. Construction of $\overline{G}$ from $G[F]$

vertices are called *black*. Thus, we want to bound the number of edges between white and black vertices in $\overline{G}$ by three times the number of white vertices.

Now, consider any component $C \in \mathcal{C}(X)$ that is not violated and has therefore not been contracted. Since all white vertices in $\overline{G}$ correspond to other components of X, C does not have a white neighbor in $\overline{G}$ and therefore does not affect the number of edges between white and black vertices. We construct the graph $\hat{G}$ by copying $\overline{G}$ and contracting or removing all these components in the following way (Figure 2(a)). Either C is isolated or has at least one neighbor in $\overline{G}$. In the first case, remove C from $\hat{G}$ because there are no connections from C to any white vertex. In the second case, let v be any neighbor of C in $\overline{G}$. Then, contract $C \cup \{v\}$ to a black vertex in $\hat{G}$ that will again be identified with v and discard multiple copies of edges. Observe that this contraction preserves planarity of $\hat{G}$.

If $\hat{G}$ is not connected, we apply the following arguments to each component. Since we want to bound the number of edges between white and black vertices the claim then follows for the entire graph. Hence, we may assume that $\hat{G}$ is connected.

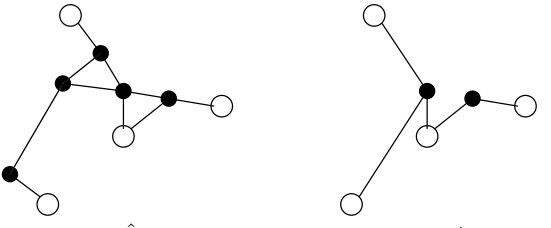
The graph $\hat{G}$ has the following properties:

(a) $\hat{G}$ is planar, simple and connected.
(b) Removing a black vertex from $\hat{G}$ splits the graph into multiple components (by minimality of F).

In the following, we transform $\hat{G}$ to a bipartite graph G' that maintains the above properties, has the same number of white vertices and the same number of edges between white and black vertices as $\hat{G}$ (Figure 2(b)). Until there are no more edges between two black vertices, iteratively perform one of the following two operations:

- Contract an edge between two black vertices that have no common white neighbor.
- Remove an edge between two black vertices with a common white neighbor.

For the sequel, the order of these operations can be arbitrary. However, different orderings of these operations can result in different graphs.



(a) $\hat{G}$. (b) G'.

Fig. 2. Construction of $\hat{G}$ and G' from $\overline{G}$ in Figure 1

Edges between white and black vertices are not affected because black vertices with a common white neighbor are not contracted. White vertices remain unchanged. G' is bipartite since all edges between two black vertices have been contracted or removed. Since white vertices correspond to components of X there are no edges between them in $\overline{G}$ and consequently $\hat{G}$ and G'. G' obeys property (a): Contraction along an edge maintains planarity and connectedness. It is simple because black vertices with common white neighbor are not contracted. Removing an edge maintains planarity and connectedness since an edge is only removed when the two black vertices have a common white neighbor. G' obeys property (b): Fix a black vertex v of G'. Due to the above contractions, v is the replacement for a set of black vertices in $\hat{G}$. Due to the minimality of F, deleting this set in $\hat{G}$ splits $\hat{G}$ into components each containing at least one white vertex. Since the white vertices are equivalent in $\hat{G}$ and G', the deletion of v also splits G' into components that each contains at least one white vertex.

Let W and B be the sets of white and black vertices in G', respectively. Recall, that we would like to bound the number of edges between W and B in terms of $|W|$. Fix any $r \in W$. Construct a breadth-first search tree T in G' rooted at r. Due to the minimality property (b), all the leafs of T are white. Since each white vertex has at most one black parent in the tree and G' is bipartite, the number of black vertices is at most the number of white vertices, *i.e.*, $|B| \leq |W|$.

Note that Demaine *et al.* [DHK09, Lemma 3, p. 337] showed a bound of $|B| \leq 2|W|$. If G is H-minor free then G' is H-minor free and the number of edges in G' is $O((|B|+|W|)m\sqrt{\log m})$ if H has m vertices. Thus, our bound implies an improvement by a factor of $2/3$ in the approximation factors for graphs with a forbidden minor.

In the following, we show the existence of a white vertex that is only connected to one black vertex in G'. We call such a vertex an *earring*. Consider the breadth-first search tree T and a leaf w with its parent v. Recall that v is black and all its children are white. Due to property (b), v has a child in T that is only connected to v in G'; otherwise removing v would leave G' connected.

Now, we show that the number of faces in G' is at most $|W|$. The argument proceeds by induction on the number of black vertices. Clearly, there is at least one black vertex. Consider a bipartite graph H on b black vertices, any number of white vertices, and satisfying (a) and (b). Let $b = 1$. Then, H is a star having one face and at least two white vertices, yielding the induction base. Suppose H has $b+1$ black vertices. Due to

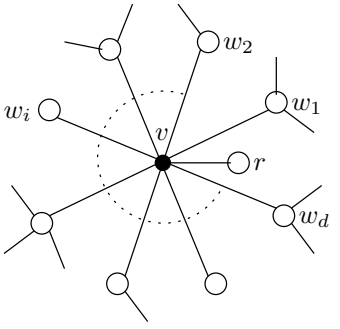


Fig. 3. Earring r in H with adjacent vertex v and its other neighbors $w_1, \ldots, w_d$

the above discussion, there exists an earring r in H (Figure 3). Let v denote its black neighbor. Further, let $(d+1)$ denote the degree of v in H and let $r, w_1, \ldots, w_d$ denote the neighbors of v. Since r is an earring, v is adjacent to at most d faces of H.

Let H' denote the graph obtained from H by contracting $v, r, w_1, \ldots, w_d$ to a new white vertex identified with r, and deleting multiple copies of edges. Clearly, H' has b black vertices, is planar, simple and connected, thus obeys (a). Consider any black vertex $v' \neq v$ in H. Since H obeys (b) we know that removing v' from H splits H into multiple components, with $v, r, w_1, \ldots, w_d$ falling into the same component. Hence, there must be a second component of $H \setminus \{v'\}$ that is also contained in H'. Therefore, v' disconnects r from this component in H' and therefore H' also obeys (b). By induction hypothesis we have that in H' the number of faces is at most the number of white vertices. H has at most d more faces than H' and exactly d more white vertices. Thus, the number of faces in H is at most the number of white vertices in H.

We now finish the proof. Let E' and F' denote the edges and faces of G', respectively. Since G' is a connected planar graph, Euler's formula yields $|E'| = |B| + |W| + |F'| - 2$. Due to the above we have $|B| \leq |W|$ and $|F'| \leq |W|$ which yields $|E'| = |B| + |W| + |F'| - 2 \leq 3|W| - 2 \leq 3|W|$. □

4 Lower Bound

We now show that the approximation factor of 3 in Theorem 2 is best possible, even when restricted to NODE-WEIGHTED STEINER TREE. Let $\epsilon > 0$ be arbitrarily small. Set $m = \lceil \frac{4}{\epsilon}(3-\epsilon) \rceil$ and $\delta = \frac{1}{3}\left(3 - \frac{1}{2}\epsilon\right)$. Construct the graph G_m as illustrated in Figure 4. Let $p, q, t_1, \ldots, t_m$ be the terminals and require that they are all connected. Set the weights $w(a_i) = w(b_i) = w(c_i) = \delta$ $(1 \leq i \leq m)$, $w(d_i) = 0$ $(1 \leq i \leq m)$, and $w(e_i) = 1$ $(2 \leq i \leq m)$.

Selecting all d_i and e_i as well as a_1, b_1 and c_1 gives a feasible solution. Therefore, the weight of an optimal solution is at most $3\delta + (m-1) \leq m+2$. The algorithm selects all a_i, b_i, c_i, d_i, incurring a weight of $3\delta m$ (see the full version of this paper for further discussion). Due to the definition of δ and m we have $(3\delta - 3 + \epsilon)m = m\epsilon/2 \geq 2(3-\epsilon)$. This gives $3m\delta \geq (3-\epsilon)(m+2) \geq (3-\epsilon)(m+3\delta-1)$ which means that the

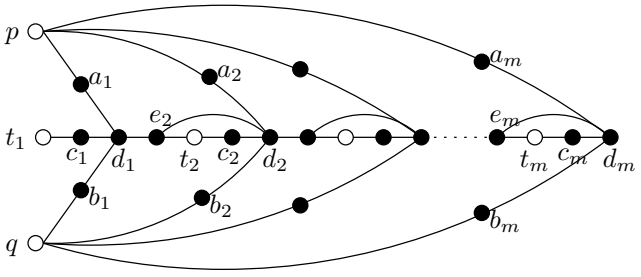


Fig. 4. Graph G_m for the lower bound construction

approximation factor of the algorithm is at least $(3 - \epsilon)$. Since ϵ was arbitrary, the algorithm cannot achieve a better approximation factor than 3 on planar graphs.

5 A 9/4-Approximation Algorithm on Planar Graphs

It can be shown that on undirected planar graphs FEEDBACK VERTEX SET reduces to NODE-WEIGHTED STEINER TREE by placing a terminal vertex in each face of the graph and connecting it to the vertices on the perimeter of the face. This suggests similarities to the feedback problems on planar graphs studied by Goemans and Williamson [GW98] which include FEEDBACK VERTEX SET. We show that after constructing a certain family of sets it is also possible to leverage the proof techniques from [GW98]. This yields an alternative derivation of Theorem 2 which we outline first. Further, reusing the arguments from this derivation we can prove a bound of $9/4$ for an improved oracle.

Alternative proof of Theorem 2. As before, let X be the infeasible solution containing all the terminals and F be the minimal feasible augmentation of X. Again, construct the graph $\hat{G}$ and let W and B denote the sets of white and black vertices in $\hat{G}$, respectively. Recall that we would like to bound the number of edges between W and B in terms of $|W|$.

A family of sets $\mathcal{S}$ is said to be *laminar* if for any two sets $S_1, S_2 \in \mathcal{S}$ we have $S_1 \subseteq S_2$, $S_2 \subseteq S_1$ or $S_1 \cap S_2 = \emptyset$. For each black vertex $v \in B$ we will select a set S_v of vertices of $\hat{G}$, called the *witness set* for v, with the following properties:

(c) For all $v \in B$, S_v contains at least one white vertex.
(d) For all $v \in B$, v is the only neighbor of S_v in $\hat{G}$.
(e) The family $\{S_v : v \in B\}$ is laminar.

Let $w \in W$ and T be a breadth-first search tree in $\hat{G}$ rooted at w. Denote the subtree of T below a vertex v and including v by $T(v)$. We will use the same notation for the vertex sets of these trees. Recall that, due to property (b), all leafs in T are white. Now, fix a vertex $v \in B$. Clearly, v is neither a leaf nor the root of T and therefore has a parent and at least one child in T. Due to property (b), removing v from $\hat{G}$ splits $\hat{G}$

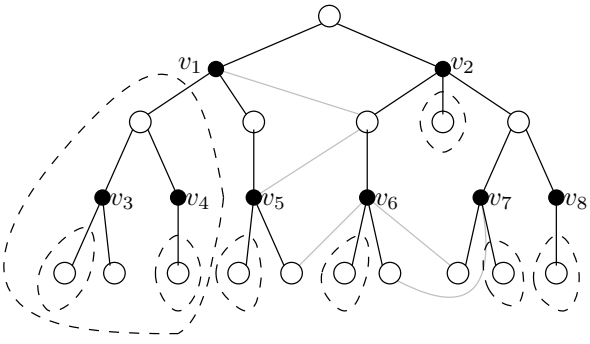


Fig. 5. Construction of S_v's from $\hat{G}$. T is marked with black edges. S_v is marked with dashed lines for all $v \in B$.

into multiple components. Let C_1 denote the component containing the parent of v. Let $C_2 \neq C_1$ be any other component of $\hat{G} \setminus \{v\}$. Set $S_v = C_2$ (see Figure 5).

Since $T \setminus T(v)$ is connected and the parent of v is in C_1 we have $T \setminus T(v) \subseteq C_1$. Hence, C_2 is the union of subtrees of some children of v in T. Since all the leafs of T are white, (c) follows. Further, v is the only neighbor of S_v in $\hat{G}$ because S_v is a component of $\hat{G} \setminus \{v\}$, which gives (d). To show (e) consider any other vertex $v' \in B$, $v' \neq v$. If neither v is a predecessor of v' in T nor v' a predecessor of v, then $S_v \subseteq T(v)$ and $S_{v'} \subseteq T(v')$ are disjoint. W.l.o.g. let v be the predecessor of v'. If $v' \in S_v$, then $S_{v'} \subseteq S_v$ since $S_{v'}$ is the union of subtrees of some children of v' which are all in S_v. If $v' \notin S_v$, then $S_v \cap T(v') = \emptyset$ since $T(v')$ is connected and therefore entirely contained in a component of $\hat{G} \setminus \{v\}$. Since $S_{v'} \subseteq T(v')$, S_v and $S_{v'}$ are disjoint. Hence, (e) holds.

The proof now continues with the arguments from Goemans and Williamson [GW98, p. 45-47]. Intuitively, the idea is to count the number of edges of some special bipartite subgraphs of $\hat{G}$ and thereby double counting all black vertices. Replace "witness cycles" by witness sets: the property that is exploited by the proof is property (d). We have shown in (e) that our family of witness sets is laminar. Property (c) is equivalent to the "Minimal Cycle Property 2" [GW98, top of p. 46]. The "Minimal Cycle Property 1" does not make sense in our context since the violated components of X are the inclusion-wise minimal sets with violated constraints (2) and are therefore contracted to (white) vertices in $\hat{G}$. □

Improved Oracle. The arguments from Goemans and Williamson heavily depend on bounding the number of edges in a bipartite graph by the number of its vertices. This bound can be improved if it is known that the bipartite graph does not have an induced cycle of length four. This is the basic idea behind the following oracle.

Consider a plane graph on white and black vertices. Two white vertices w_1 and w_2 with common neighbors $b_1, \ldots, b_d$ partition the plane into several regions. One of these regions contains the exterior face and we refer to all the other regions as *pockets*. We say that two white vertices w_1 and w_2 *surround* a third white vertex w_3 if w_3 is contained in one of the pockets of w_1 and w_2.

The oracle C_4-free-violated-components first contracts every violated component of X in G to a white vertex (Step (ii)). Second, if the obtained graph does not contain two white vertices w_1, w_2 that surround a third white vertex w_3, the oracle returns all violated components. Otherwise, the oracle subsequently selects the pocket of w_3 with respect to w_1 and w_2 (Step (iii)). Let $\mathcal{V}$ be the set of selected white vertices whose corresponding components are returned in Step (iv). The oracle ensures that there do not exist two vertices in $\mathcal{V}$ that surround a third vertex in $\mathcal{V}$, and $\mathcal{V}$ consists of all white vertices contained in one pocket of two white vertices.

C_4-free-violated-components(G, X)

(i) $H \leftarrow G$.
(ii) Contract each violated component $C \in \text{AVC}(X)$ to a white vertex in H. (Call all other vertices black.)
(iii) While H contains two white vertices w_1 and w_2 that surround another white vertex w_3
Set H to be the interior of the pocket of w_1 and w_2 containing w_3.
(iv) Return the violated components of X corresponding to the white vertices of H.

Theorem 3. *Let G be planar, X be an infeasible solution containing all the terminals, F be any minimal augmentation of X, and $\mathcal{V}$ be the sets returned by C_4-free-violated-components(G, X). Then,*

$$\sum_{S \in \mathcal{V}} |F \cap \Gamma(S)| \leq \frac{9}{4} |\mathcal{V}|.$$

Proof. Analog to the proof of Theorem 2, construct $\overline{G}$ by contracting every violated component of X in $G[F]$ to a white vertex. Further, obtain $\hat{G}$ from $\overline{G}$ by removing all isolated non-violated components of X and contracting all other non-violated components of X into a black neighbor. Recall that all non-violated components of X do not have a white neighbor in $\overline{G}$ since the white vertices also correspond to components of X. Therefore, contracting non-violated components does not create new cycles containing two white vertices. Hence, two white vertices w_1 and w_2 surround a third white vertex w_3 in $\hat{G}$ if and only if they surround w_3 in $\overline{G}$. Thus, the white vertices corresponding to $\mathcal{V}$ in $\hat{G}$ do not contain two white vertices surrounding a third vertex from $\mathcal{V}$.

We now use the above construction of a laminar family of witness sets that satisfy (c), (d) and (e). The claim then follows by applying the arguments of Goemans and Williamson [GW98, p. 49-52] by, again, interchanging "witness cycles" with witness sets. □

The analysis given in Theorem 3 is best possible (see the full version of this paper for a worst case example).

References

[BCE+10] Bateni, M., Chekuri, C., Ene, A., Hajiaghayi, M., Korula, N., Marx, D.: Prize-collecting Steiner Problems on Planar Graphs. In: 22nd Annual ACM-SIAM Symposium on Discrete Algorithms (2010)

[BGRS10] Byrka, J., Grandoni, F., Rothvoß, T., Sanità, L.: An improved LP-based Approximation for Steiner Tree. In: 42nd ACM Symposium on Theory of Computing, pp. 583–592 (2010)

[BHM10] Bateni, M., Hajiaghayi, M., Marx, D.: Approximation Schemes for Steiner Forest on Planar Graphs and Graphs of Bounded Treewidth. In: 42nd ACM Symposium on Theory of Computing, pp. 211–220 (2010)

[BKM09] Borradaile, G., Klein, P., Mathieu, C.: An O($n \log n$) Approximation Scheme for Steiner Tree in Planar Graphs. ACM Transactions on Algorithms 5(3), 1–31 (2009)

[CC08] Chlebík, M., Chlebíková, J.: The Steiner tree problem on graphs: Inapproximability results. Theoretical Computer Science 406, 207–214 (2008)

[DG10] Dilkina, B., Gomes, C.P.: Solving Connected Subgraph Problems in Wildlife Conservation. In: Lodi, A., Milano, M., Toth, P. (eds.) CPAIOR 2010. LNCS, vol. 6140, pp. 102–116. Springer, Heidelberg (2010)

[DHK09] Demaine, E.D., Hajiaghayi, M., Klein, P.N.: Node-Weighted Steiner Tree and Group Steiner Tree in Planar Graphs. In: Albers, S., Marchetti-Spaccamela, A., Matias, Y., Nikoletseas, S., Thomas, W. (eds.) ICALP 2009. LNCS, vol. 5555, pp. 328–340. Springer, Heidelberg (2009)

[Fei98] Feige, U.: A threshold of $\ln n$ for approximating set cover. Journal of the ACM 45, 634–652 (1998)

[GJ77] Garey, M.R., Johnson, D.S.: The Rectilinear Steiner Tree Problem is NP-Complete. SIAM Journal on Applied Mathematics 32(4), 826–834 (1977)

[GMNS99] Guha, S., Moss, A., Naor, J.S., Schieber, B.: Efficient recovery from power outage (extended abstract). In: 31st ACM Symposium on Theory of Computing, pp. 574–582 (1999)

[GW97] Goemans, M.X., Williamson, D.P.: The Primal-Dual Method for Approximation Algorithms and its Application to Network Design Problems, ch. 4, pp. 144–191 (1997), In Hochbaum [Hoc97]

[GW98] Goemans, M.X., Williamson, D.P.: Primal-Dual Approximation Algorithms for Feedback Problems in Planar Graphs. Combinatorica 18, 37–59 (1998)

[Hoc97] Hochbaum, D.S. (ed.): Approximation Algorithms for NP-hard Problems. PWS Publishing Co., Boston (1997)

[Kar72] Karp, R.M.: Reducibility Among Combinatorial Problems. In: Complexity of Computer Computations, pp. 85–103 (1972)

[KR95] Klein, P., Ravi, R.: A Nearly Best-Possible Approximation Algorithm for Node-Weighted Steiner Trees. Journal of Algorithms 19(1), 104–115 (1995)

[MR07] Moss, A., Rabani, Y.: Approximation Algorithms for Constrained Node Weighted Steiner Tree Problems. SIAM Journal on Computing 37(2), 460–481 (2007)

Steiner Transitive-Closure Spanners of Low-Dimensional Posets*,**

Piotr Berman[1], Arnab Bhattacharyya[2], Elena Grigorescu[3], Sofya Raskhodnikova[1], David P. Woodruff[4], and Grigory Yaroslavtsev[1]

[1] Pennsylvania State University, USA
{berman,sofya,grigory}@cse.psu.edu
[2] Massachusetts Institute of Technology, USA
abhatt@mit.edu
[3] Georgia Institute of Technology, USA
elena@cc.gatech.edu
[4] IBM Almaden Research Center, USA
dpwoodru@us.ibm.com

Abstract. Given a directed graph $G = (V, E)$ and an integer $k \geq 1$, a *Steiner k-transitive-closure-spanner* (Steiner k-TC-spanner) of G is a directed graph $H = (V_H, E_H)$ such that (1) $V \subseteq V_H$ and (2) for all vertices $v, u \in V$, the distance from v to u in H is at most k if u is reachable from v in G, and ∞ otherwise. Motivated by applications to property reconstruction and access control hierarchies, we concentrate on Steiner TC-spanners of directed acyclic graphs or, equivalently, partially ordered sets. We study the relationship between the dimension of a poset and the size, denoted S_k, of its sparsest Steiner k-TC-spanner.

We present a nearly tight lower bound on S_2 for d-dimensional directed hypergrids. Our bound is derived from an explicit dual solution to a linear programming relaxation of the 2-TC-spanner problem. We also give an efficient construction of Steiner 2-TC-spanners, of size matching the lower bound, for all low-dimensional posets. Finally, we present a nearly tight lower bound on S_k for d-dimensional posets.

1 Introduction

Graph spanners were introduced in the context of distributed computing by Awerbuch [3] and Peleg and Schäffer [12], and since then have found numerous applications. Our focus is on transitive-closure spanners, introduced explicitly in [5], but studied prior to that in many different contexts (see references in [5]).

Given a directed graph $G = (V, E)$ and an integer $k \geq 1$, a **k-transitive-closure-spanner** (k-TC-spanner) of G is a directed graph $H = (V, E_H)$ such

* E.G. is supported by NSF award CCR-0829672 and NSF award 1019343 to the Computing Research Association for the Computing Innovation Fellowship Program. S.R. and G.Y. are supported by NSF / CCF CAREER award 0845701. G.Y. is also supported by University Graduate Fellowship and College of Engineering Fellowship.

** Full version is available at http://arxiv.org/abs/1011.6100.

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 760–772, 2011.

that: (1) E_H is a subset of the edges in the transitive closure of G; (2) for all vertices $u, v \in V$, if $d_G(u,v) < \infty$ then $d_H(u,v) \leq k$ and if $d_G(u,v) = \infty$ then $d_H(u,v) = \infty$, where $d_G(u,v)$ denotes the distance from u to v in G. That is, a k-TC-spanner is a graph with a small diameter that preserves the connectivity of the original graph. The edges of the transitive closure of G, added to G to obtain a TC-spanner, are called *shortcuts* and the parameter k is called the *stretch*.

TC-spanners have numerous applications, and there has been a lot of work on finding sparse TC-spanners for specific graph families. See [13] for a survey. In some applications of TC-spanners, in particular, to access control hierarchies [2,8], the shortcuts can use *Steiner* vertices, that is, vertices not in the original graph G. The resulting spanner is called a *Steiner TC-spanner*.

Definition 1.1 (Steiner TC-spanner). *Given a directed graph $G = (V, E)$ and an integer $k \geq 1$, a* ***Steiner k-transitive-closure-spanner (Steiner k-TC-spanner)*** *of G is a directed graph $H = (V_H, E_H)$ such that: (1) $V \subseteq V_H$; (2) for all vertices $u, v \in V$, if $d_G(u,v) < \infty$ then $d_H(u,v) \leq k$ and if $d_G(u,v) = \infty$ then $d_H(u,v) = \infty$. Vertices in $V_H \backslash V$ are called* Steiner vertices.

For some graphs, Steiner TC-spanners can be significantly sparser than ordinary TC-spanners. For example, consider a complete bipartite graph $K_{\frac{n}{2},\frac{n}{2}}$ with $n/2$ vertices in each part and all edges directed from the first part to the second. Every ordinary 2-TC-spanner of this graph has $\Omega(n^2)$ edges. However, $K_{\frac{n}{2},\frac{n}{2}}$ has a Steiner 2-TC-spanner with n edges: it is enough to add one Steiner vertex v, edges to v from all nodes in the left part, and edges from v to all nodes in the right part. Thus, for $K_{\frac{n}{2},\frac{n}{2}}$ there is a factor of $\Theta(n)$ gap between the size of the sparsest Steiner 2-TC-spanner and the size of an ordinary 2-TC-spanner.

We focus on Steiner TC-spanners of directed *acyclic* graphs (DAGs) or, equivalently, partially ordered sets (posets). They represent the most interesting case in applications of TC-spanners. In addition, there is a reduction from constructing TC-spanners of graphs with cycles to constructing TC-spanners of DAGs, with a small loss in stretch ([13], Lemma 3.2), which also applies to Steiner TC-spanners.

The goal of this work is to understand the minimum number of edges needed to form a Steiner k-TC-spanner of a given graph G as a function of n, the number of nodes in G. More specifically, motivated by applications to access control hierarchies [2,8] and property reconstruction [4,11], described in Section 1.2, we study the relationship between the dimension of a poset and the size of its sparsest Steiner TC-spanner. The *dimension* of a poset G is the smallest d such that G can be embedded into a d-dimensional directed hypergrid via an order-preserving embedding. (See Definition 2.1). Atallah *et al.* [2], followed by De Santis *et al.* [8], use Steiner TC-spanners in key management schemes for access control hierarchies. They argue that many access control hierarchies are low-dimensional posets that come equipped with an embedding demonstrating low dimensionality. For this reason, we focus on the setting where the dimension d is small relative to the number of nodes n.

We also study the size of sparsest (Steiner) 2-TC-spanners of specific posets of dimension d, namely, d-dimensional directed hypergrids. Our lower bound on this

quantity improves the lower bound of [4] and nearly matches their upper bound. It implies that our construction of Steiner 2-TC-spanners of d-dimensional posets is optimal up to a constant factor for any constant number of dimensions. It also has direct implications for property reconstruction.

1.1 Our Results

Steiner 2-TC-spanners of Directed d-dimensional Grids. The *directed hypergrid*, denoted $\mathcal{H}_{m,d}$, has vertex set[1] $[m]^d$ and edge set $\{(x, y) : \exists$ unique $i \in [d]$ such that $y_i - x_i = 1$ and if $j \neq i, y_j = x_j\}$. We observe (in Corollary 2.4) that for the grid $\mathcal{H}_{m,d}$, Steiner vertices do not help to create sparser k-TC-spanners. In [4], it was shown that for $m \geq 3$, sparsest (ordinary) 2-TC-spanners of $\mathcal{H}_{m,d}$ have size at most $m^d \log^d m$ and at least $\Omega\big(\frac{m^d \log^d m}{(2d \log\log m)^{d-1}}\big)$. They also give tight upper and lower bounds for the case of constant m and large d. Our first result is an improvement on the lower bound for the hypergrid for the case when m is significantly larger than d, i.e., the setting in the above applications.

Theorem 1.1. *All (Steiner) 2-TC-spanners of $\mathcal{H}_{m,d}$ have $\Omega(\frac{m^d (\ln m - 1)^d}{(4\pi)^d})$ edges.*

The proof of Theorem 1.1 constructs a dual solution to a linear programming relaxation of the 2-TC-spanner problem. We consider a linear program (LP) for the sparsest 2-TC-spanner of $\mathcal{H}_{m,d}$. Our program is a special case of a more general LP for the sparsest directed k-spanner of an arbitrary graph G, used in [5] to obtain an approximation algorithm for that problem. We show that for our special case the integrality gap of this LP is small and, in particular, does not depend on n. Specifically, we find a solution to the dual LP by selecting initial values that have a combinatorial interpretation: they are expressed in terms of the *volume* of d-dimensional *boxes* contained in $\mathcal{H}_{m,d}$. For example, the dual variable corresponding to the constraint that enforces the existence of a length-2 path from u to v in the 2-TC-spanner is initially assigned a value inversely proportional to the number of nodes on the paths from u to v. The final sum of the constraints is bounded by an integral which, in turn, is bounded by an expression depending only on the dimension d.

We note that the best lower bound known previously [4] was proved by a long and sophisticated combinatorial argument that carefully balanced the number of edges that stay within different parts of the hypergrid and the number of edges that cross from one part to another. The recursion in the combinatorial argument is an inherent limitation of [4], resulting in suboptimal bounds even for constant d. In contrast, our linear programming argument can be thought of as assigning types to edges based on the volume of the boxes they define, and automatically balancing the number of edges of different types by selecting the correct coefficients for the constraints corresponding to those edges. It achieves an optimal bound for any constant number of dimensions.

[1] For a positive integer m, we denote $\{1, \ldots, m\}$ by $[m]$.

Table 1. Steiner k-TC-spanner sizes for d-dimensional posets on n vertices for $d \geq 2$

Stretch k	Prior bounds on $S_k(G)$
$2d-1$	$O(n^2)$ [2]
$2d-2+t\ \forall t \geq 2$	$O(n(\log^{d-1} n)\lambda_t(n))$ [2]
$2d+O(\log^* n)$	$O(n \log^{d-1} n)$ [2]
3	$O(n \log^{d-1} n \log\log n)$ for fixed d [8]

Stretch k	Our bounds on $S_k(G)$	
2	$O(n \log^d n)$	$\Omega\left(n\left(\frac{\log n}{cd}\right)^d\right)$ for a fixed $c > 0$
≥ 3		$\Omega(n \log^{\lceil (d-1)/k \rceil} n)$ for fixed d

Steiner TC-spanners of General *d*-dimensional Posets. We continue the study of the number of edges in a sparsest Steiner k-TC-spanner of a poset as a function of its dimension, following [2,8]. We note that the only poset of dimension 1 is the directed line $\mathcal{H}_{n,1}$. TC-spanners of directed lines were discovered under many different guises. (See references in [5].) It was implicitly shown in [6,7] that, for constant k, the size of the sparsest k-TC-spanner of $\mathcal{H}_{n,1}$ is $\Theta(n \cdot \lambda_k(n))$, where $\lambda_k(n)$ is the k^{th}-row inverse Ackermann function.

Table 1 compares old and new results for $d \geq 2$. $S_k(G)$ denotes the number of edges in the sparsest Steiner k-TC-spanner of G. The upper bounds hold for all posets of dimension d. The lower bounds mean that there is an infinite family of d-dimensional posets with sparsest Steiner k-TC-spanners of the specified size.

Atallah *et al.* constructed Steiner k-TC-spanners with k proportional to d. De Santis *et al.* improved their construction for constant d. They achieved $O(3^{d-t} nt \log^{d-1} n \log\log n)$ edges for odd stretch $k = 2t+1$, where $t \in [d]$. In particular, setting $t = 1$ gives $k = 3$ and $O(n \log^{d-1} n \log\log n)$ edges.

We present the first construction of Steiner 2-TC-spanners for d-dimensional posets. In our construction, the spanners have $O(n \log^d n)$ edges, and the length-2 paths can be found in $O(d)$ time. This result is stated in Theorem 2.2 (in Section 2). Our construction, like all previous constructions, takes as part of the input an explicit embedding of the poset into a d-dimensional grid. (Finding such an embedding is NP-hard [15]. Also, as mentioned previously, in the application to access control hierarchies, such an embedding is usually given.) The Steiner vertices used in our construction are necessary to obtain sparse TC-spanners. An (easy) example that demonstrates this is deferred to the full version.

Theorem 1.1 implies that there is an absolute constant $c > 0$ for which our upper bound for $k = 2$ is tight within an $O((cd)^d)$ factor, showing that no drastic improvement in the upper bound is possible. To obtain a bound in terms of the number n of vertices and dimension d, substitute m^d with n and $\ln m$ with $(\ln n)/d$ in the theorem statement. This gives the following corollary.

Corollary 1.2 *There is an absolute constant $c > 0$ for which for all $d \geq 2$ and n larger than some constant to the power d, there exists a d-dimensional poset G on n vertices such that every Steiner 2-TC-spanner of G has $\Omega\left(n\left(\frac{\log n}{cd}\right)^d\right)$ edges.*

In addition, we prove a lower bound for all constant $k > 2$ and constant dimension d, which qualitatively matches known upper bounds. It shows that, in

particular, every Steiner 3-TC-spanner has size $\Omega(n \log n)$, and even with significantly larger constant stretch, every Steiner TC-spanner has size $n \log^{\Omega(d)} n$.

Theorem 1.3. *For all constant $d \geq 2$ and sufficiently large n, there exists a d-dimensional poset G on n vertices such that for all $k \geq 3$, every Steiner k-TC-spanner of G has $\Omega(n \log^{\lceil (d-1)/k \rceil} n)$ edges.*

This theorem (see Section 4) captures the dependence on d and greatly improves upon the previous $\Omega(n \log \log n)$ bound, which follows trivially from known lower bounds for 3-TC-spanners of a directed line.

The lower bound on the size of a Steiner k-TC-spanner for $k \geq 3$ is proved by the probabilistic method. We note that using the hypergrid as an example of a poset with large Steiner k-TC-spanners for $k > 2$ would yield a much weaker lower bound because $\mathcal{H}_{m,d}$ has a 3-TC-spanner of size $O((m \log \log m)^d)$ and, more generally, a k-TC-spanner of size $O((m \cdot \lambda_k(m))^d)$, where $\lambda_k(m)$ is the k^{th}-row inverse Ackermann function [4]. Instead, we construct an n-element poset embedded in $\mathcal{H}_{n,d}$ as follows: all poset elements differ on coordinates in dimension 1, and for each element, the remaining $d-1$ coordinates are chosen uniformly at random from $[n]$. We consider a set of partitions of the underlying hypergrid into d-dimensional boxes, and carefully count the expected number of edges in a Steiner k-TC-spanner that cross box boundaries for each partition. We show that each edge is counted only a small number of times, proving that the expected number of edges in a Steiner k-TC-spanner is large. We conclude that some poset attains the expected number of edges.

Organization. We explain applications of Steiner TC-spanners in Section 1.2. Section 2 gives basic definitions and observations. In particular, our construction of sparse Steiner 2-TC-spanners for d-dimensional posets (the proof of Theorem 2.2) is presented there. Our lower bounds constitute the main technical contribution of this paper. The lower bound for the hypergrid for $k = 2$ (Theorem 1.1) is proved in Section 3. The lower bound for $k > 2$ (Theorem 1.3) is presented in Section 4.

1.2 Applications

Numerous applications of TC-spanners are surveyed in [13]. We focus on two of them: property reconstruction, described in [4,11], and key management for access control hierarchies, described in [2,5,8].

Property Reconstruction. A *local filter* [14] (see also a slightly modified definition in [4,11]) reconstructs an arbitrary function f to ensure that the reconstructed function g has the desired property, changing f only when necessary. A local filter is given a function f and a query x and, after looking up the value of f on a small number of points, it has to output $g(x)$ for some function g, which has the desired property and does not depend on x. If f has the property, g must be equal to f.

Our results on TC-spanners are relevant to reconstruction of two properties of functions: monotonicity, studied in [1,4,14] and having a low Lipschitz constant, studied in [11]. In [4], the authors proved that the existence of a local filter for monotonicity of functions with low lookup complexity implies the existence of a sparse 2-TC-spanner of $\mathcal{H}_{m,d}$. In [11], an analogous connection was drawn between local reconstruction of functions with low Lipschitz constant and 2-TC-spanners. Our improvement in the lower bound on the size of 2-TC-spanners of $\mathcal{H}_{m,d}$ directly translates into an improvement by the same factor in the lower bounds on lookup complexity of local nonadaptive filters for these two properties, showing they are nearly optimal for any constant d.

Key Management for Access Control Hierarchies. Atallah *et al.* [2] used sparse Steiner TC-spanners to construct efficient key management schemes for access control hierarchies. An *access hierarchy* is a partially ordered set G of access classes. Each user is entitled to access a certain class and all classes reachable from the corresponding node in G. In the approach from [2,8] to enforcing the access hierarchy, a user from a class u can compute cryptographic keys necessary to access a class v in time proportional to $d_G(u, v)$. To speed this up, Atallah *et al.* suggest adding edges and nodes to G to increase connectivity. To preserve the access hierarchy represented by G, the new graph H must be a Steiner TC-spanner of G. With this modification, the number of edges in H corresponds to the space complexity of the scheme, while the running time has two components: the time to find a path of length at most k from u to v in H and the time to compute the cryptographic keys. The second component is proportional to the stretch k of H. In our construction of Steiner 2-TC-spanners, the time to find length-k paths is $O(d)$. For small d, it is likely to be dominated by the second component which involves a (time-consuming) evaluation of a cryptographic hash function.

2 Definitions and Observations

For integers $j \geq i$, an interval $[i, j]$ refers to the set $\{i, i+1, \ldots, j\}$. Logarithms are always base 2, except for ln which is the natural logarithm.

Each DAG $G = (V, E)$ is equivalent to a poset with elements V and partial order $\preceq$, where $x \preceq y$ if y is reachable from x in G. Elements x and y are *comparable* if $x \preceq y$ or $y \preceq x$, and *incomparable* otherwise. We write $x \prec y$ if $x \preceq y$ and $x \neq y$. The *hypergrid* $\mathcal{H}_{m,d}$ *with dimension* d *and side length* m was defined in the beginning of Section 1.1. Equivalently, it is the poset on elements $[m]^d$ with the *dominance order*, defined as follows: $x \preceq y$ for two elements $x, y \in [m]^d$ iff $x_i \leq y_i$ for all $i \in [d]$.

A mapping f from a poset G to a poset G' is called an *embedding* if it respects the partial order, that is, $f(x) \preceq_{G'} f(y)$ iff $x \preceq_G y$ for all $x, y \in G$.

Definition 2.1 ([10]). *Let G be a poset with n elements. The* dimension *of G is the smallest integer d such that G can be embedded into the hypergrid $\mathcal{H}_{n,d}$.*

As shown in [9], for any $m > 1$, the hypergrid $\mathcal{H}_{m,d}$ has dimension exactly d.

Fact 2.1. *Each d-dimensional poset G with n elements can be embedded into a hypergrid $\mathcal{H}_{n,d}$, so that for all $i \in [d]$, the ith coordinates of images of all elements are distinct. Moreover, such an embedding can be obtained from an arbitrary embedding of G into $\mathcal{H}_{n,d}$ in time $O(dn \log n)$.*

Sparse Steiner 2-TC-spanners for d-dimensional Posets. We give a simple construction of sparse Steiner 2-TC-spanners for d-dimensional posets. For constant d, it matches the lower bound from Section 3 up to a constant factor. Note that the construction itself works for arbitrary, not necessarily constant, d.

Theorem 2.2. *Each d-dimensional poset G on n elements has a Steiner 2-TC-spanner H of size $O(n \log^d n)$. Given an embedding of G into the hypergrid $\mathcal{H}_{n,d}$, H can be constructed in time $O(dn \log^d n)$. Moreover, for all $x, y \in G$, where $x \prec y$, one can find a path in H from x to y of length at most 2 in time $O(d)$.*

Proof. Consider an n-element poset G embedded into the hypergrid $\mathcal{H}_{n,d}$. Transform it, so that for all $i \in [d]$, the ith coordinates of images of all elements are distinct. (See Fact 2.1) In this proof, assume that the hypergrid coordinates start with 0, i.e., its vertex set is $[0, n-1]^d$. Let $\ell = \lceil \log n \rceil$ and $b(t)$ be the ℓ-bit binary representation of t, possibly with leading zeros. Let $p_i(t)$ denote the i-bit prefix of $b(t)$ followed by a single 1 and then $\ell - i - 1$ zeros. Let $lcp(t_1, t_2) = p_i(t_1)$, where i is the length of the longest common prefix of $b(t_1)$ and $b(t_2)$.

To construct a Steiner 2-TC-spanner (V_H, E_H) of G, we insert at most ℓ^d edges into E_H per each poset element. Consider a poset element with coordinates $x = (x_1, \ldots, x_d)$ in the embedding. For each d-tuple $(i_1, \ldots, i_d) \in [0, \ell - 1]^d$, let p be a hypergrid vertex whose coordinates have binary representations $(p_{i_1}(x_1), \ldots, p_{i_d}(x_d))$. If $x \prec p$, we add an edge (x, p) to E_H; otherwise, if $p \prec x$ we add an edge (p, x) to E_H. Note that only edges between comparable points are added to E_H.

Observe that for $d > (2 \log n)/(\log \log n)$, the theorem is trivial since then $n \log^d n > n^3$, and the transitive-closure of G has $O(n^2)$ edges and can be computed in $O(n^3)$ time. For smaller d, $\lceil \log n \rceil^d = O(\log^d n)$ and, consequently, E_H contains $O(n \log^d n)$ edges and can be constructed in $O(dn \log^d n)$ time, as described, if bit operations on coordinates can be performed in $O(1)$ time.

For all pairs of poset elements $x = (x_1, \ldots, x_d)$ and $y = (y_1, \ldots, y_d)$, such that $x \prec y$, there is an intermediate point z with coordinates whose binary representations are $(lcp(x_1, y_1), \ldots, lcp(x_d, y_d))$. By construction, both edges (x, z) and (z, y) are in E_H. Point z can be found in $O(d)$ time, since $lcp(x_i, y_i)$ can be computed in $O(1)$ time, assuming $O(1)$ time bit operations on coordinates. □

Equivalence of Steiner and non-Steiner TC-spanners for Hypergrids. Our lower bound on the size of 2-TC-spanners for d-dimensional posets of size n is obtained by proving a lower bound on the size of the Steiner 2-TC-spanner of $\mathcal{H}_{m,d}$ where $m = n^{1/d}$. The following lemma, used in Section 4, implies Corollary 2.4 that shows that sparsest Steiner and non-Steiner 2-TC-spanners of $\mathcal{H}_{m,d}$ have the same size. The proof of the lemma is deferred to the full version.

Lemma 2.3 *Let G be a poset on elements $V \subseteq [m]^d$ with the dominance order and $H = (V_H, E_H)$ be a Steiner k-TC-spanner of G with minimal V_H. Then H can be embedded into $\mathcal{H}_{m,d}$.*

Corollary 2.4 *If $\mathcal{H}_{m,d}$ has a Steiner k-TC-spanner H, it also has a k-TC-spanner with the same number of nodes and at most the same number of edges.*

3 Lower Bound for 2-TC-spanners of the Hypergrid

In this section, we prove Theorem 1.1 that gives a nearly tight lower bound on the size of (Steiner) 2-TC-spanners of the hypergrids $\mathcal{H}_{m,d}$. By Corollary 2.4, we only have to consider non-Steiner TC-spanners.

Proof (of Theorem 1.1). We start by introducing an LP for the sparsest 2-TC-spanner of an arbitrary graph. Our lower bound on the size of a 2-TC-spanner of $\mathcal{H}_{m,d}$ is obtained by finding a feasible solution to the dual program, which, by definition, gives a lower bound on the objective function of the primal.

An Integer LP for Sparsest 2-TC-spanner. For each graph $G = (V, E)$, we can find the size of a sparsest 2-TC-spanner by solving the following {0,1}-LP, a special case of an LP from [5] for directed k-spanners. For all vertices $u, v \in V$ satisfying $u \preceq v$, we introduce variables $x_{uv} \in \{0, 1\}$. For $u \neq v$, they correspond to potential edges in a 2-TC-spanner H of G. For all vertices $u, v, w \in V$ satisfying $u \preceq w \preceq v$, we introduce auxiliary variables $x'_{uwv} \in \{0, 1\}$, corresponding to potential paths of length at most 2 in H. The {0,1}-LP is as follows:

$$\begin{aligned} \text{minimize} \quad & \sum_{u,v\colon u \preceq v} x_{uv} \\ \text{subject to} \quad & x_{uw} - x'_{uwv} \geq 0, x_{wv} - x'_{uwv} \geq 0 && \forall u, v, w\colon u \preceq w \preceq v; \\ & \sum_{w\colon u \preceq w \preceq v} x'_{uwv} \geq 1 && \forall u, v\colon u \preceq v. \end{aligned}$$

Given a solution to the LP, we can construct a 2-TC-spanner $H = (V, E_H)$ of G of size not exceeding the value of the objective function by including (u, v) in E_H iff the corresponding variable $x_{uv} = 1$ and $u \neq v$. In the other direction, given a 2-TC-spanner $H = (V, E_H)$ of G, we can find a feasible solution of the LP with the value of the objective function not exceeding $|E_H| + |V|$. Let $E'_H = E_H \cup L$, where L is the set of loops (v, v) for all $v \in V$. Then we set $x_{uv} = 1$ iff $(u, v) \in E'_H$ and $x'_{uwv} = 1$ iff both $(u, w) \in E'_H$ and $(w, v) \in E'_H$. Therefore, the size of a sparsest 2-TC-spanner of G and the optimal value of the objective function of the LP differ by at most $|V|$. They are asymptotically equivalent because $|V| = O(|E_H|)$ for every weakly connected graph G.

A Fractional Relaxation of the Dual LP. Every feasible solution of the following fractional relaxation of the dual LP gives a lower bound on the optimal value of the objective function of the primal:

$$\begin{aligned}
&\text{maximize} && \sum_{u,v\colon u\preceq v} y_{uv} && \\
&\text{subject to} && \sum_{w\colon v\preceq w} y'_{uvw} + \sum_{w\colon w\preceq u} y''_{wuv} \le 1 && \forall u,v\colon u\preceq v; && (1)\\
& && y_{uv} - y'_{uwv} - y''_{uwv} \le 0 && \forall u,v,w\colon u\preceq w\preceq v; && (2)\\
& && y_{uv}\ge 0, y'_{uwv}\ge 0, y''_{uwv}\ge 0 && \forall u,v,w\colon u\preceq w\preceq v. &&
\end{aligned}$$

Finding a Feasible Solution for the Dual. When the graph G is a hypergrid $\mathcal{H}_{n,d}$, we can find a feasible solution of the dual, which gives a lower bound on the objective function of the primal. To do that, we perform the following three steps. First, we choose initial values $\hat{y}_{uv}$ for the variables y_{uv} of the dual program and, in Lemma 3.1, give a lower bound on the resulting value of the objective function of the primal program. Second, we choose initial values $\hat{y}'_{uvw}$ and $\hat{y}''_{uvw}$ for variables y'_{uvw} and y''_{uvw} so that (2) holds. Finally, in Lemma 3.2, we give an upper bound on the left-hand side of (1) for all $u\preceq v$. Our bound is a constant larger than 1 and independent of n. We obtain a feasible solution to the dual by dividing the initial values of the variables (and, consequently, the value of the objective function) by this constant.

Step 1. For a vector $x=(x_1,\ldots,x_d)\in[0,m-1]^d$, let the *volume* $V(x)$ denote $\prod_{i\in[d]}(x_i+1)$. This corresponds to the number of hypergrid points inside a d-dimensional box with corners u and v, where $v-u=x$. We start building a solution to the dual by setting $\hat{y}_{uv}=\frac{1}{V(v-u)}$ for all $u\preceq v$. This gives the value of the objective function of the dual program, according to the following lemma.

Lemma 3.1 $\displaystyle\sum_{u,v\colon u\preceq v}\hat{y}_{uv} > m^d(\ln m-1)^d.$

Proof. Substituting $1/(V(v-u))$ for $\hat{y}_{uv}$, we get:

$$\begin{aligned}
\sum_{u,v\colon u\preceq v}\hat{y}_{uv} &= \sum_{u,v\colon u\preceq v}\frac{1}{V(v-u)} = \sum_{l\in[m]^d}\prod_{i\in[d]}\frac{m-l_i+1}{l_i} = \left(\sum_{l\in[m]}\frac{m-l+1}{l}\right)^d\\
&> ((m+1)\ln(m+1)-m)^d > m^d(\ln m-1)^d. \qquad \square
\end{aligned}$$

Step 2. The values of $\hat{y}'_{uvw}$ and $\hat{y}''_{uvw}$ are set as follows to satisfy (2) tightly (without any slack):

$$\hat{y}'_{uvw}=\hat{y}_{uw}\frac{V(v-u)}{V(v-u)+V(w-v)},\quad \hat{y}''_{uvw}=\hat{y}_{uw}-\hat{y}'_{uvw}=\hat{y}_{uw}\frac{V(w-v)}{V(v-u)+V(w-v)}.$$

Step 3. The initial values $\hat{y}'_{uvw}$ and $\hat{y}''_{uvw}$ do not necessarily satisfy (1). The following lemma, whose proof is deferred to the full version, gives an upper bound on the left-hand side of all constraints in (1).

Lemma 3.2 *For all* $u \preceq v$, $\sum_{w\colon v \preceq w} \hat{y}'_{uvw} + \sum_{w\colon w \preceq u} \hat{y}''_{wuv} \leq (4\pi)^d$.

Finally, we obtain a feasible solution by dividing initial values $\hat{y}_{uv}$, $\hat{y}'_{uvw}$ and $\hat{y}''_{uvw}$ by the upper bound $(4\pi)^d$ from Lemma 3.2. Then Lemma 3.1 gives the desired bound on the value of the objective function:

$$\sum_{u,v\colon u \preceq v} \frac{\hat{y}_{uv}}{(4\pi)^d} > m^d \left(\frac{\ln m - 1}{4\pi}\right)^d.$$

This concludes the proof of Theorem 1.1. □

4 Lower Bound for k-TC-Spanners for $k > 2$

In this section, we prove Theorem 1.3 that gives a lower bound on the size of Steiner k-TC-spanners of d-dimensional posets for $k > 2$ and $d \geq 2$.

Proof (of Theorem 1.3). Unlike in the previous section, the poset which attains the lower bound is constructed probabilistically, not explicitly.

We consider n-element posets G embedded in the hypergrid $\mathcal{H}_{n,d}$, where the partial order is given by the dominance order $x \preceq y$ on $\mathcal{H}_{n,d}$. The elements of G are points $p_1, p_2, \ldots, p_n \in [n]^d$, where the first coordinate of each p_a is a. (By Fact 2.1, each d-dimensional poset with n elements can be embedded into $\mathcal{H}_{n,d}$, so that the first coordinates of all points are distinct.) Let $\mathcal{G}_d$ be a distribution on such posets G, where the last $d-1$ coordinates of each point p_a are chosen uniformly and independently from $[n]$.

Recall that $S_k(G)$ denotes the size of the sparsest Steiner k-TC-spanner of poset G. The following lemma gives a lower bound on the expected size of a Steiner k-TC-spanner of a poset drawn from $\mathcal{G}_d$.

Lemma 4.1 $\mathop{\mathbb{E}}_{G \leftarrow \mathcal{G}_d}[S_k(G)] = \Omega(n \log^{\lceil \frac{d-1}{k} \rceil} n)$ *for all* $k \geq 3$ *and constant* $d \geq 2$.

In this extended abstract, we only prove the special case of Lemma 4.1 for 2-dimensional posets (Lemma 4.2). The general case is deferred to the full version. Since Lemma 4.1 implies the existence of a poset G, for which every Steiner k-TC-spanner has $\Omega(n \log^{\lceil (d-1)/k \rceil} n)$ edges, Theorem 1.3 follows. □

The Case of $d = 2$. Next we prove a special case of Lemma 4.1 for 2-dimensional posets, which illustrates many ideas used in the proof of Lemma 4.1.

Lemma 4.2 $\mathop{\mathbb{E}}_{G \leftarrow \mathcal{G}_2}[S_k(G)] = \Omega(n \log n)$ *for all* $k \geq 3$ *and* $d = 2$.

Proof. We can assume that $\ell = \log n$ is an integer. To analyze the expected number of edges in a Steiner TC-spanner H of G, we consider ℓ partitions of $[n]^2$ into horizontal strips. We call strips *boxes* for compatibility with the case of general d.

Definition 4.1 (Box partition). *For each $i \in [\ell]$, define sets of equal size that partition $[n]$ into 2^i intervals: the jth such set, for $j \in [2^i]$, is $I^i_j = [(j-1)2^{\ell-i} + 1, j2^{\ell-i}]$. Given $i \in [\ell]$, and $j \in [2^i]$, the* box $\mathbb{B}(i,j)$ *is $[n] \times I^i_j$ and the* box partition $\mathbb{BP}(i)$ *is a partition of $[n]^2$ that contains boxes $\mathbb{B}(i,j)$ for all $j \in [2^i]$.*

For each odd j, we group boxes $\mathbb{B}(i,j)$ and $\mathbb{B}(i,j+1)$ into a *box-pair*. We call j the *index* of the box-pair and refer to $\mathbb{B}(i,j)$ and $\mathbb{B}(i,j+1)$ as the *bottom* and the *top* box in the box-pair. Recall that a poset G consists of elements $p_1, p_2, \ldots, p_n \in [n]^2$, where the first coordinate of each p_a is a. We analyze the expected number of edges in a Steiner TC-spanner H of G that cross from bottom to top boxes in all box-pairs. To do that, we identify pairs of poset elements (p_a, p_b), called *jumps*, that force such edges to appear. By Lemma 2.3, we can assume that all Steiner vertices of H are embedded into $\mathcal{H}_{n,2}$. Therefore, if p_a is in the bottom box and p_b is in the top box of the same box-pair then H must contain an edge from the bottom to the top box. To ensure that we count such an edge just once, we consider only p_a and p_b for which no other point p_c with $c \in (a,b)$ is contained in this box pair. Next we define *jumps* formally. This concept is also illustrated in Figure 1.

Definition 4.2 (Jumps). *Given a poset G, embedded into $\mathcal{H}_{n,2}$, and an index $i \in [\ell]$, a* jump generated by the box partition $\mathbb{BP}(i)$ *is a pair (p_a, p_b) of elements of G, such that for some odd $j \in [2^i]$, the following holds: $p_a \in \mathbb{B}(i,j)$, $p_b \in \mathbb{B}(i,j+1)$, but $p_c \notin \mathbb{B}(i,j) \cup \mathbb{B}(i,j+1)$ for all $c \in (a,b)$. The set of jumps generated by all partitions $\mathbb{BP}(i)$ for $i \in [\ell]$ is denoted by $\mathcal{J}$.*

Next we establish that the number of jumps in a poset G is a lower bound on the number of edges in a Steiner TC-spanner of G (Claim 4.3) and bound the expected number of jumps from below (Claim 4.4).

Claim 4.3. *Let G be a poset, embedded into $\mathcal{H}_{n,2}$, and $H = (V_H, E_H)$ be a Steiner k-TC-spanner of G. Then $|E_H| \geq |\mathcal{J}|$.*

To prove the claim, we establish an injective mapping from $\mathcal{J}$ to E_H. The proof is deferred to the full version.

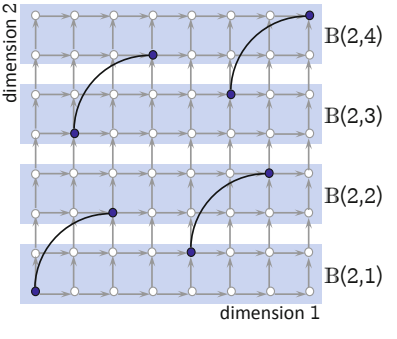


Fig. 1. Box partition $\mathbb{BP}(2)$ and jumps it generates

Claim 4.4. *When a poset G is drawn from the distribution $\mathcal{G}_2$, the expected size of $\mathcal{J}$ is at least $n(\ell-1)/4$.*

Proof. We first find the expected number of jumps generated by the partition $\mathbb{BP}(i)$ for a specific i. Let $\lambda_i(p_a)$ be the index j of the box-pair $\mathbb{B}(i,j) \cup \mathbb{B}(i,j+1)$ that contains p_a. Let $\rho_i(p_a)$ be 0 if p_a is in the bottom box of that box pair, and 1 otherwise. One can think of $\lambda_i(p_a)$ as the location of p_a, and of $\rho_i(p_a)$ as its relative position within a box-pair. Importantly, when G is drawn from $\mathcal{G}_2$, that is, the second coordinates of points p_a for all $a \in [n]$ are chosen uniformly and independently from $[n]$, then random variables $\rho_i(p_a)$ are independent and uniform over $\{0,1\}$ for all $a \in [n]$.

We group together points p_a that have equal values of $\lambda_i(p_a)$, and sort points within groups in increasing order of their first coordinate a. Since there are 2^{i-1} box-pairs, the number of groups is at most 2^{i-1}. Observe that random variables $\rho_i(p_a)$ within each group are uniform and independent because random variables $\lambda_i(p_a)$ and $\rho_i(p_a)$ are independent for all $a \in [n]$. Now, if we list $\rho_i(p_a)$ in the sorted order for all points in a particular group, we get a sequence of 0s and 1s. Two consecutive entries correspond to a jump iff they are 01. The last position in a group cannot correspond to the beginning of a jump. The number of positions that can correspond to the beginning of a jump in all groups is n minus the number of groups, which gives at least $n - 2^{i-1}$. For each such position, the probability that it starts a jump (i.e., the probability of 01) is $1/4$. Thus, the expected number of jumps generated by the partition $\mathbb{BP}(i)$ is at least $(n - 2^{i-1})/4$.

Summing over all $i \in [\ell]$, we get the expected number of jumps in all partitions: $(n\ell - \sum_{i=1}^{\ell} 2^{i-1})/4 > n(\ell-1)/4 = \Omega(n \log n)$. □

Claims 4.3 and 4.4 imply that, for a poset G drawn from $\mathcal{G}_2$, the expected number of edges in a Steiner TC-spanner of G is $\Omega(n \log n)$, concluding the proof of Lemma 4.2. □

References

1. Ailon, N., Chazelle, B., Comandur, S., Liu, D.: Property-preserving data reconstruction. Algorithmica 51(2), 160–182 (2008)
2. Atallah, M.J., Blanton, M., Fazio, N., Frikken, K.B.: Dynamic and efficient key management for access hierarchies. ACM Trans. Inf. Syst. Secur. 12(3) (2009)
3. Awerbuch, B.: Communication-time trade-offs in network synchronization. In: PODC, pp. 272–276 (1985)
4. Bhattacharyya, A., Grigorescu, E., Jha, M., Jung, K., Raskhodnikova, S., Woodruff, D.P.: Lower bounds for local monotonicity reconstruction from transitive-closure spanners. In: Serna, M., Shaltiel, R., Jansen, K., Rolim, J. (eds.) APPROX 2010, LNCS, vol. 6302, pp. 448–461. Springer, Heidelberg (2010)
5. Bhattacharyya, A., Grigorescu, E., Jung, K., Raskhodnikova, S., Woodruff, D.P.: Transitive-closure spanners. In: Mathieu, C. (ed.) SODA, pp. 932–941. SIAM, Philadelphia (2009)
6. Chandra, A.K., Fortune, S., Lipton, R.J.: Lower bounds for constant depth circuits for prefix problems. In: Díaz, J. (ed.) ICALP 1983. LNCS, vol. 154, pp. 109–117. Springer, Heidelberg (1983)

7. Chandra, A.K., Fortune, S., Lipton, R.J.: Unbounded fan-in circuits and associative functions. J. Comput. Syst. Sci. 30(2), 222–234 (1985)
8. De Santis, A., Ferrara, A.L., Masucci, B.: New constructions for provably-secure time-bound hierarchical key assignment schemes. Theor. Comput. Sci. 407(1-3), 213–230 (2008)
9. Dushnik, B., Miller, E.: Concerning similarity transformations of linearly ordered sets. Bulletin Amer. Math. Soc. 46, 322–326 (1940)
10. Dushnik, B., Miller, E.W.: Partially ordered sets. Amer. J. Math. 63, 600–610 (1941)
11. Jha, M., Raskhodnikova, S.: Testing and reconstruction of Lipschitz functions with applications to data privacy. Electronic Colloquium on Computational Complexity (ECCC) TR11-057 (2011)
12. Peleg, D., Schäffer, A.A.: Graph spanners. J. Graph Theory 13(1), 99–116 (1989)
13. Raskhodnikova, S.: Transitive-closure spanners: A survey. In: Goldreich, O. (ed.) Property Testing. LNCS, vol. 6390, pp. 167–196. Springer, Heidelberg (2010)
14. Saks, M.E., Seshadhri, C.: Local monotonicity reconstruction. SIAM J. Comput. 39(7), 2897–2926 (2010)
15. Yannakakis, M.: The complexity of the partial order dimension problem. SIAM Journal on Matrix Analysis and Applications 3(3), 351–358 (1982)

Solving the Chromatic Cone Clustering Problem via Minimum Spanning Sphere*

Hu Ding and Jinhui Xu

Department of Computer Science and Engineering
State University of New York at Buffalo
{huding,jinhui}@buffalo.edu

Abstract. In this paper, we study the following *Chromatic Cone Clustering (CCC)* problem: Given n point-sets with each containing k points in the first quadrant of the d-dimensional space R^d, find k cones apexed at the origin such that each cone contains at least one distinct point (i.e., different from other cones) from every point-set and the total size of the k cones is minimized, where the size of a cone is the angle from any boundary ray to its center line. CCC is motivated by an important biological problem and finds applications in several other areas. Our approaches for solving the CCC problem relies on solutions to the *Minimum Spanning Sphere (MinSS)* problem for point-sets. For the MinSS problem, we present two $(1+\epsilon)$-approximation algorithms based on core-sets and ϵ-net respectively. With these algorithms, we then show that the CCC problem admits $(1+\epsilon)$-approximation solutions for constant k. Our results are the first solutions to these problems.

Keywords: High Dimension, Chromatic, Clustering, Core-Set.

1 Introduction

In this paper, we consider the following *Chromatic Cone Clustering (CCC)* problem: Given n point-sets with each containing k points in the first quadrant of R^d space, find k cones apexed at the origin such that each cone contains at least one distinct point (i.e., different from other cones) from every point-set and the total size of the k cones is minimized, where the size of a cone is the angle from any boundary ray (emitting from the apex) to its center line. For a particular point p, it is possible that more than one cone contains it. But p is viewed as "belonging" to only one of them. Similarly, a cone could contain multiple points from a point-set, but only one of them is viewed as "contained" by this cone. Essentially, the k cones induce a k-coloring for the n point-sets with the constraint that no pair of points in a point-set shares the same color. The CCC problem is thus seeking to solve the k-coloring and k-clustering problems simultaneously. In this paper, we consider the cases of $k = 2$ or a small constant number.

* This research was partially supported by NSF through CAREER Award CCF-0546509 and grant IIS-0713489.

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 773–784, 2011.

The CCC problem is motivated by an important application in biology for determining topologies of chromosomes. In such applications, the interested chromosome is first labeled with a number of BAC probes along the DNA chain. Each probe forms a point in the 3D nuclear images. To determine the existence of any common spatial distribution pattern of probes among a population of cells, the set of probes from each chromosome is first converted into a point in the feature space, where each dimension is the distance between a particular pair of probes. The feature points from the same chromosome are then clustered into a cone apexed at the origin. Using feature points and cone clustering determines how well a spatial distribution pattern of the probes is preserved among all cells. Since each chromosome has two copies (or homologs) inside the nucleus, each normal cell contributes two points in the feature space. For cancer cells, the number of chromosome homologs could be more than 2. Due to physical limitations, it is in general difficult to identify the same homolog from all cells in a population. Thus the problem of finding common spatial distribution pattern needs to first classify (i.e., coloring) all feature points into (two or more) groups and then cluster each of them using a minimum bounding cone apex at the origin, which can be formulated as a CCC problem. The CCC problem also arises in other pattern recognition, data mining, or machine learning applications where each data item could have more than one independent copy.

CCC is a new and challenging problem. To our best knowledge, no previous algorithm exists. The challenge mainly comes from the requirement on simultaneous coloring and clustering. Coloring could significantly change the proximity of point-sets (since it is now determined by the distance of points with the same color) and force the algorithm to take into consideration of all possible colorings. To overcome this difficulty, Our main idea is to first reduce the problem to the Minimum Spanning Sphere (MinSS) problem and then use the solution of the MinSS to derive a solution to the CCC problem.

The MinSS is an interesting problem in its own right. It is a natural generalization of the Minimum Enclosing Sphere (MinES) which has received considerable attention in recent years [2,3,4,6,8,10]. For the MinSS problem, previous research has focused on lower dimensional space (e.g., 2 and 3-D space) [1,5], and not much work has done in higher dimensional space. In this paper, we first show that the MinSS problem is NP-Complete and has no FPTAS unless $P = NP$. Then we give two $(1+\epsilon)$-approximation algorithms for MinSS based on core sets and ϵ-net techniques respectively. For the CCC problem, we consider the case that k is either 2 or some other small constant. Using the algorithms for MinSS and a number of other interesting geometric techniques, we show that there exist PTAS for both cases. Our algorithms are the first approximation solutions to both MinSS and CCC problems.

2 Minimum Spanning Sphere

In this section, we study the MinSS problem. We first give some hardness results and then present two PTAS algorithms for MinSS.

Definition 1 (MinSS). *Let $\mathcal{G} = \{G_1, G_2, \ldots, G_n\}$ be n point-sets (or groups) in R^d space with each $G_i = \{p_1^i, p_2^i, \ldots, p_k^i\}$, $1 \leq i \leq n$, consisting of k points, $k > 1$. The minimum spanning sphere B_{ms} of $\mathcal{G}$ is the sphere with the minimum radius and containing at least one point from each point-set.*

Existing techniques for the MinSS problem have mainly focused on lower dimensional space (i.e., 2 or 3-D space). Huttenlocher *et al.* [7] introduced a Voronoi diagram based method for the MinSS problem in 2 and 3-D space. Their method first constructs the upper envelope of the Voronoi surfaces of each point, and then shows that if a point is on the boundary of the MinSS, the center of the MinSS must lie on the upper envelope (of the Voronoi surfaces). Based on this observation, their algorithm searches the center of the MinSS on the upper envelope by first identifying a set of candidate points and then choosing the best candidate as the center of MinSS. It is easy to see that the above observation still holds for higher dimensional space. However, due to the high complexity (exponential with respect to the dimensionality) associated with the upper envelope in higher dimensional space, such an approach would not lead to efficient solution. Thus our main focus will be on deriving approximation solution to the MinSS problem in higher dimensional space.

Definition 2 (Selection and Optimal Selection). *Given n point-sets $\mathcal{G} = \{G_1, G_2, \ldots, G_n\}$ in R^d with each G_i of size k, a selection of $\mathcal{G}$ is a set $S = \{p_1, p_2, \ldots, p_n\}$ of n points with each $p_i \in G_i$ for $1 \leq i \leq n$. A partial selection of $\mathcal{G}$ is a subset of some selection. A selection S is an optimal selection if the radius of the MinES of S is the same as that of the MinSS of $\mathcal{G}$.*

From the above definition, it is easy to see that optimal selection may not be unique for a given instance of the MinSS problem.

Lemma 1. *Finding an Optimal Selection for the MinSS problem is NP-Complete.*

Definition 3. *The distance from a point p to a point set S in R^d is $dis(p, S) = min\{|p - s||s \in S\}$.*

The following lemma is used in this paper and proved in [4].

Lemma 2. *Given a point set $S \in R^d$,and $Ball(c, r)$ is the minimum enclosing sphere. Then any closed half space that contains the center c must also contain one point from S that is at distance r from c.*

Theorem 1. *There is no FPTAS for MinSS when $k \geq 2$, unless $P = NP$.*

Proof. Suppose that the MinSS problem has an FPTAS algorithm. Below we show that the 3-SAT problem can be solved in polynomial time by making use of a construction in [9].

Let $U = \{\pm e_i | i = 1, 2, \cdots, d\}$ be a set of $2d$ points in R^d space, where e_i (or $-e_i$) is the point with its only non-zero coordinate 1 (or -1) in the i-th dimension. For an instance of 3-SAT, $E_1 \wedge E_2 \cdots \wedge E_m$ with $E_i = x_i \vee y_i \vee z_i$ and $x_i, y_i, z_i \in \{\mu_1, \overline{\mu_1}, \cdots, \mu_{d-1}, \overline{\mu_{d-1}}\}$, construct point sets $P = \{p^j | j =$

$1, 2, \cdots, m\}$ and $\overline{P} = \{-p^j | p^j \in P, j = 1, 2, \cdots, m\}$ in R^d space with $p^j_d = 3\alpha$ (i.e., the d-th coordinate of p^j) and $p^j_i = \alpha$, $-\alpha$ or 0 depending on whether E_j contains μ_i, $\overline{\mu_i}$, or neither of them, where α is a positive number satisfying $12\alpha^2 - 4d^{-1}\alpha + d^{-1} < 1 - d^{-1} < 12\alpha^2 + d^{-1}$. Points in $P \cup \overline{P} \cup U$ can be partitioned into $d + m$ point-sets, $A_1, \cdots, A_{d+m}$, with each containing 2 points symmetric about the origin.

Let $\{A_1, \cdots, A_d\}$ be the d point-sets from U, and $\{A_{d+1}, \cdots, A_{d+m}\}$ be the m point-sets from $P \cup \overline{P}$. For any selection from $\{A_1, \cdots, A_d\}$, we know that the minimum enclosing sphere has a radius of $\sqrt{1 - d^{-1}}$, and a center at $(\pm\frac{1}{d}, \cdots, \pm\frac{1}{d})$, where the sign of each coordinate is determined by the selection of e_i or $-e_i$. Thus for any point in $P \cup \overline{P}$, its distance to the center is

$$\sqrt{\frac{d-4}{d^2} + (\frac{1}{d} \pm \alpha)^2 + (\frac{1}{d} \pm \alpha)^2 + (\frac{1}{d} \pm \alpha)^2 + (\frac{1}{d} \pm 3\alpha)^2}.$$

Depending on the sign of α, there are only at most 8 possible values for the distance from any point in $P \cup \overline{P}$ to the center. Furthermore, these 8 possible values are independent from the selection of $\{A_1, \cdots, A_d\}$ and the boolean formula of $E_1 \wedge E_2 \cdots \wedge E_m$.

For any selection S_d of $\{A_1, \cdots, A_d\}$, let $B(c, \sqrt{1 - d^{-1}})$ be its minimum enclosing sphere (MinES). If the MinES of S_d is not the MinES of any optimal selection of $\{A_1, \cdots, A_{d+m}\}$, there must exist a point-set in $\{A_{d+1}, \cdots, A_{d+m}\}$ with a larger than $\sqrt{1 - d^{-1}}$ distance to c. Since there are only 8 possible distances, we let $\{v_1, \cdots, v_8\}$ be the 8 distances and $\beta = min\{v_i - \sqrt{1 - d^{-1}} | v_i > \sqrt{1 - d^{-1}}, i = 1, \cdots, 8\}$.

Let $S = \{a_i | a_i \in A_i, i = 1, \cdots, d + m\}$ be an optimal selection of $\{A_1, \cdots, A_{d+m}\}$. If $E_1 \wedge E_2 \cdots \wedge E_m$ is satisfiable, then the MinES of $S_d = \{a_1, \cdots, a_d\}$ (i.e., the d points from U) is the MinES of S. Otherwise (i.e., $E_1 \wedge E_2 \cdots \wedge E_m$ is unsatisfiable), the MinES of S must include S_d and one point, say q, whose distance to the center c of the MinES of S_d is at least $\beta + \sqrt{1 - d^{-1}}$. By Lemma 2, we can show that the radius of the MinES of S is at least $\overline{R} = \sqrt{1 - d^{-1}} + \frac{\beta^2}{2(\sqrt{1-d^{-1}}+\beta)}$. To see this, let $r = \sqrt{1 - d^{-1}}$ be the radius of the MinES $B(c, \sqrt{1 - d^{-1}})$ of S_d, and $B(c', r')$ be the MinES of $S_d \cup \{q\}$. Let δ be the distance $dis(c, c')$ between c and c'. By Lemma 2 and triangle inequaltity, we know that $r' \geq \sqrt{r^2 + \delta^2}$ and $r' \geq r + \beta - \delta$. Thus $r' \geq max\{\sqrt{r^2 + \delta^2}, r + \beta - \delta\}$, and achieves its minimum value $r + \frac{\beta^2}{2(r+\beta)}$ when $\sqrt{r^2 + \delta^2} = r + \beta - \delta$ and $\delta = \frac{\beta^2 + 2\beta r}{2(r+\beta)}$. This means that $\overline{R} = \sqrt{1 - d^{-1}} + \frac{\beta^2}{2(\sqrt{1-d^{-1}}+\beta)}$. Thus, the minimum radius difference between any satisfiable and any unsatisfiable 3-SAT instances is $\overline{R} - \sqrt{1 - d^{-1}}$ and the relative minimum radius difference is $\frac{\overline{R} - \sqrt{1-d^{-1}}}{\sqrt{1-d^{-1}}}$.

From the definition of α, we know that β can be represented by an arbitrary fraction of $\frac{1}{d}$. Thus we can lower bound $\frac{\overline{R} - \sqrt{1-d^{-1}}}{\sqrt{1-d^{-1}}}$ as $\frac{1}{d^z}$, where z is some natural number.

Let ϵ be $\frac{1}{d^z}$. If the radius of a $(1+\epsilon)$-approximation of the MinSS is larger than or equal to $\overline{R}$, then $E_1 \wedge E_2 \cdots \wedge E_m$ is unsatisfiable. Otherwise (i.e., the radius

is smaller than $\overline{R}$) , $E_1 \wedge E_2 \cdots \wedge E_m$ is satisfiable. This means that if there is an FPTAS for the MinSS problem, there would exist an algorithm for the 3-SAT problem with a polynomial running time in n and d (as $1/\epsilon$ is a polynomial of d). This can only happen if $P = NP$. □

In the following part, we present two PTAS algorithms with running time $O((nkd + 2^{\frac{2}{\epsilon}} d^3\ ) k^{\frac{2}{\epsilon}+1})$ and $O(nk^2 d(1 + \frac{1}{\sqrt{d}} \log \frac{1}{\epsilon} (\frac{\sqrt{2e\pi}}{\epsilon})^d))$ respectively.

Our first algorithm uses core-sets techniques [3,4]. In [3], Badoiu and Clarkson designed an elegant core-set-based algorithm for the minimum enclosing sphere (MinES) problem for n points in R^d space. Their algorithm repeatedly finds the farthest point from the current center and moves the center toward it. The set of farthest points forms the core-set and its size is independent of n and d (i.e., $\frac{2}{\epsilon} + 1$).

Since the MinSS problem shares a similar objective to the MinES problem, i.e., both minimize the enclosing sphere of a set of points, we also use an iterative approach for the MinSS problem. However, due to the different nature of the MinSS problem, we cannot simply choose the farthest point in each step. This is because we do not know in advance which subset of points will be spanned by the MinSS. To overcome this difficulty, our main idea is to maintain a set of possible partial solutions (i.e., selections) and for each partial solution, we adopt a strategy similar to the MinES algorithm in [3]. In [3] the radius in each iteration satisfies the following key inequalities.

$$R_{i+1} \geq \overline{R} - K_i \tag{1}$$

$$R_{i+1} \geq \sqrt{R_i^2 + K_i^2}, \tag{2}$$

where R_i is the radius of the sphere in the i-th iteration, K_i is the distance between the two centers in the i-th and $(i+1)$-th iterations, and $\overline{R} = (1+\epsilon)R_{opt}$. The two inequalities can be proved by using triangle inequality and Lemma 2. For the MinSS problem, we show that among the set of maintained partial solutions, there exists a sequence of partial solutions which converges in a way similar to the MinES algorithm.

Before we present our algorithm for the MinSS problem, we first observe that in each step the algorithm in [3] chooses the farthest point to the current center to expand its core-set. This is to ensure that in each step the radius will have a significant improvement. However, if we take a close look at the proof in [3], this is not always necessary. Actually as long as the next point could ensure that Inequalities (1) and (2) are preserved, the number of needed iterations will still be $\frac{2}{\epsilon} + 1$ (the same proof in [3] still works). The following lemma gives the requirement on the next point.

Lemma 3. *In the MinES algorithm in [3], if the next point has a distance to the current center no smaller than $\overline{R} = (1+\epsilon)R_{opt}$, Inequalities (1) and (2) hold.*

Proof. Let p be the next point and c_i be the current center. By triangle inequality, we have $R_{i+1} + K_i \geq dis\{c_i, p\} \geq \overline{R}$. Thus Inequality (1) holds. Inequality (2) follows directly from Lemma 2. □

MinSS Algorithm 1
Input: A set of point-sets $\mathcal{G} = \{G_1, G_2, \ldots, G_n\}$ with each containing k points in R^d space.
Output: A $(1+\epsilon)$-approximation of the MinSS.

1. Initialize a tree T with each node v of T associating with a partial selection S_v of $\mathcal{G}$ and the MinES $B_v(c, r)$ of S_v. Initially, T is a single-node (i.e., the root) tree associated with an empty selection.
2. Create k children for the root, choose an arbitrary point-set, say G_1, and add the i-th point p_i^1 of G_1 into the partial selection of the i-th child of the root.
3. For each node v of the tree, recursively expand it in the following way.
   (a) If the MinES $B_v(c, r)$ of S_v covers at least one point from each group, v is a leaf node of T.
   (b) If the height of v is $\lceil \frac{2}{\epsilon} \rceil + 1$, v is a leaf node of T.
   (c) Otherwise, find the point-set G_t which has the largest distance (by definition 3) to the center c of the MinES $B_v(c, r)$. Let l be the distance. Then create k children for v, with each child inheriting the partial selection S_v as its current partial selection, insert the i-th point p_i^t of G_t into the partial selection of the i-th child of v, and compute the MinES for each child.
4. Search all nodes in tree T, find the node with the smallest l and its associated $B(c, r)$, and output $B(c, l)$.

Theorem 2. *MinSS Algorithm 1 returns a* $(1+\epsilon)$*-approximation of the MinSS, and runs in* $O((nkd + 2^{\frac{2}{\epsilon}} d^3) k^{\frac{2}{\epsilon}+1})$ *time.*

Due to space limit, we omit the algorithm for another $(1+\epsilon)$-approximation for the MinSS problem based on ϵ-net.

3 The Chromatic Cone Clustering Problem

Definition 4 (CCC). *Let* $\mathcal{G} = \{G_1, G_2, \cdots, G_n\}$ *be the* n *point-sets with each* $G_i = \{p_1^i, \cdots, p_k^i\}$ *containing* k *points in the first quadrant of the* R^d *space. The objective of the CCC problem is to find* k *cones,* $C_1, C_2, \cdots, C_k$*, so that all cones are apexed at the origin, each* C_i *contains a distinct point from every point-set* G_i*, and the total angle* $\sum_{i=1}^{k} \alpha_i$ *is minimized, where* α_i *is the angle between any bounding ray and the center line of* C_i*.*

Without loss of generality, we assume that each point p_j^i is located on the boundary of the R^d unit sphere centered at the origin. Note that this can be easily satisfied by projecting each point onto the unit sphere, without affecting the solution. When $k = 2$, we denote the problem as 2-CCC.

Definition 5 (Chromatic Partition). *For a given CCC problem instance* $\mathcal{G}$*, a chromatic partition of* $\mathcal{G}$ *is a partition of the* nk *points into* k *sets,* $U_1, \cdots, U_k$*, such that each* U_i *contains a distinct point from each* G_j *for* $j = 1, 2, \cdots, n$*. A chromatic partition is optimal if the total angle of the MinEC of each* U_i *is minimized among all possible chromatic partitions.*

Definition 6 (Chromatic Condition). *For a given CCC problem instance $\mathcal{G}$, a set of k selections or partial selections, $S_1, \cdots, S_k$, satisfies the chromatic condition if there exists a chromatic partition, $U_1, \cdots, U_k$, such that $S_i \subseteq U_i$ for $i \in \{1, \cdots, k\}$.*

Definition 7. *Given a point p in R^d and an angle α, the cone centered at the ray emitting from the origin and passing through p and with angle α is denoted as $C(p, \alpha)$.*

The following two lemmas reveal some relation between the problems of computing minimum enclosing spheres and minimum enclosing cones.

Lemma 4. *Let S be a set of points in the first quadrant of R^d and on the unit sphere centered at the origin. The minimum enclosing sphere(MinES) of S has radius of r if and only if the angle of the minimum enclosing cone(MinEC) of S is* $\arcsin(r)$.

Lemma 5. *Let S be a set of points in the first quadrant of R^d and on the unit sphere centered at the origin. If $B(c, r)$ is a $(1+\epsilon)$-approximation of MinES of S, then $C(c, \arcsin r)$ is a $(1 + \frac{\theta}{\frac{\pi}{2}-\theta})$-approximation of MinEC of S, where $\theta = \arcsin(\frac{\sqrt{2\epsilon+\epsilon^2}}{1+\epsilon})$. Furthermore, if the angle of the MinEC of S is upper bounded by a small constant β (i.e., significantly smaller than $\frac{\pi}{2}$), then $C(c, \arcsin r)$ is a $(1 + \frac{\tan\beta}{\beta}\frac{1}{\sqrt{1-(2\epsilon+\epsilon^2)\tan^2\beta}}\epsilon)$- approximation of the MinEC of S.*

3.1 Constant Ratio Approximation Algorithm for $k = 2$

In this section, we present a constant ratio approximation algorithm for yielding upper and lower bounds for the optimal solution and then use them to design a $(1+\epsilon)$-approximation algorithm for the CCC problem when $k = 2$.

2-CCC Algorithm 1
Input: A set of point-sets $\mathcal{G} = \{G_1, G_2, \ldots, G_n\}$ with each containing 2 points in the first quadrant of the R^d space and on the unit sphere centered at the origin, and a constant $\epsilon \in (0, 1]$.
Output: A constant approximation solution to the CCC problem.

1. Let $j = 1$, and repeat Step (2) 2 times.
2. Compute a $(1+\epsilon)$-approximation of the MinSS for $\mathcal{G}$, and let S_j be the resulting sphere with center c_j and radius r_j. Delete the points covered by S_j from each point-set. If more than one point from a point-set is covered by S_j, arbitrarily delete one of them. Let $j = j + 1$.
3. Output the 2 cones, $C(c_i, \arcsin r_i)$, $i = 1, 2$, corresponding to the 2 spheres.

Lemma 6. *2-CCC Algorithm 1 gives a $(\frac{\pi}{2}(2 + \frac{\epsilon}{2})(1+\epsilon))$-approximation of the 2-CCC problem.*

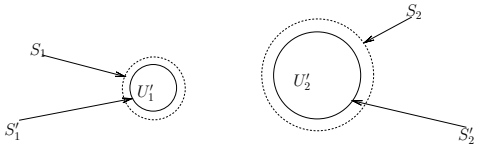


Fig. 1. An example for illustrating Lemma 6: Case 1

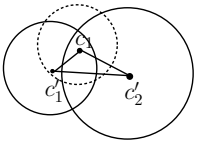


Fig. 2. An example for illustrating Lemma 6: Case 2

Proof. First it is obvious that $C(c_1, \arcsin r_1)$ and $C(c_2, \arcsin r_2)$ form a feasible (may not be optimal) solution to the 2-CCC problem. The corresponding chromatic partition for the $2n$ points of $\mathcal{G}$ are U_1 and U_2. We assume U_1' and U_2' are the optimal chromatic partition according to Definition 5. The MinES for U_1' and U_2' are S_1' and S_2', the centers are c_1' and c_2', and the radii are r_1' and r_2' respectively. Let R_L be the radius of the MinSS of $\mathcal{G}$, and R_U be the radius of the MinES of all points in $\mathcal{G}$. There are two cases to consider. (See Figures 1 and 2).

1. $U_1 = U_1'$ or $U_1 = U_2'$. Without loss of generality, assume that $U_1 = U_1'$. This means that $U_2 = U_2'$. Since S_1 is a $(1+\epsilon)$-approximation of the MinSS of $\mathcal{G}$, we know that $r_1 \leq (1+\epsilon)R_L \leq (1+\epsilon)r_1'$, where the last inequality is due to the fact that S_1' is a spanning sphere of $\mathcal{G}$. Also since S_2 is a $(1+\epsilon)$-approximation of the MinES of U_2 and S_2' is the MinES of U_2', we have $r_2 \leq (1+\epsilon)r_2'$. Thus, $r_1 + r_2 \leq (1+\epsilon)(r_1' + r_2')$. It is easy to get $\arcsin r_1 + \arcsin r_2 \leq \frac{\pi}{2}(r_1 + r_2) \leq \frac{\pi}{2}(1+\epsilon)(r_1' + r_2') \leq \frac{\pi}{2}(1+\epsilon)(\arcsin r_1' + \arcsin r_2')$. Hence $C(c_1, \arcsin r_1)$ and $C(c_2, \arcsin r_2)$ form a $\frac{\pi}{2}(1+\epsilon)$-approximation.
2. U_1 overlaps with both U_1' and U_2'. In this case, S_1 must intersect with both S_1' and S_2'. Let $h = dis(c_1', c_2')$. By triangle inequality, we have $h \leq ||c_1 - c_1'|| + ||c_1 - c_2'|| \leq r_1 + r_1' + r_1 + r_2' = 2r_1 + r_1' + r_2'$. Also, $r_2 \leq (1+\epsilon)R_U \leq (1+\epsilon)\frac{r_1'+r_2'+h}{2} \leq (1+\epsilon)(r_1 + r_1' + r_2')$. Thus, $\frac{r_1+r_2}{r_1'+r_2'} \leq \frac{r_1+(1+\epsilon)(r_1+r_1'+r_2')}{r_1'+r_2'} = 1+\epsilon+\frac{(2+\epsilon)r_1}{r_1'+r_2'}$. Since S_1' and S_2' are both spanning spheres for $\mathcal{G}$, we know that $r_1 \leq (1+\epsilon)r_1'$ and $r_1 \leq (1+\epsilon)r_2'$. Hence, $\frac{r_1+r_2}{r_1'+r_2'} \leq (1+\epsilon)(2+\frac{\epsilon}{2})$. From this, we immediately have $\arcsin r_1 + \arcsin r_2 \leq \frac{\pi}{2}(r_1+r_2) \leq \frac{\pi}{2}(1+\epsilon)(2+\frac{\epsilon}{2})(r_1'+r_2') \leq \frac{\pi}{2}(1+\epsilon)(2+\frac{\epsilon}{2})(\arcsin r_1' + \arcsin r_2')$. This means that $C(c_1, \arcsin r_1)$ and $C(c_2, \arcsin r_2)$ are $\frac{\pi}{2}(1+\epsilon)(2+\frac{\epsilon}{2})$-approximation of the 2-CCC problem.

□

3.2 $(1+\epsilon)$ Algorithms for the Case of $k = 2$

In the 2-CCC problem, each point-set G_i in $\mathcal{G}$ contains two points $\{p_1^i, p_2^i\}$. Let U_1 and U_2 be an optimal chromatic partition of $\mathcal{G}$, C_1 and C_2 be the MinEC, and $S_1(c_1, r_1)$ and $S_2(c_2, r_2)$ be the MinES of U_1 and U_2 respectively.

Definition 8 ((t, δ, b)-Sequence). *Let t, b and δ be three positive numbers with $t \leq 2b$ and $0 \leq \delta \leq 1$. The (t, δ, b)-sequence consists of $\lceil \frac{1}{\delta} \rceil$ pairs of numbers*

(β_1^i, β_2^i), $i = 1, \cdots, \lceil \frac{1}{\delta} \rceil$, *and is defined as follows. If* $t \leq b$, *then* $\beta_1^i = i\frac{t}{\lceil \frac{1}{\delta} \rceil}$ *and* $\beta_2^i = t - (i-1)\frac{t}{\lceil \frac{1}{\delta} \rceil}$; *otherwise (i.e.,* $t > b$*),* $\beta_1^i = t - b + i\frac{t-2(t-b)}{\lceil \frac{1}{\delta} \rceil}$ *and* $\beta_2^i = b - (i-1)\frac{t-2(t-b)}{\lceil \frac{1}{\delta} \rceil}$ *for* $i = 1, \cdots, \lceil \frac{1}{\delta} \rceil$.

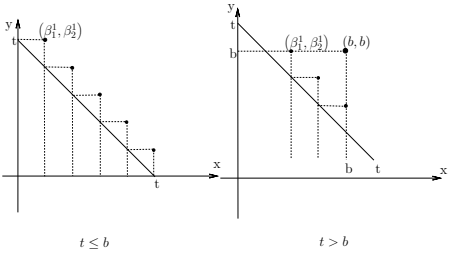


Fig. 3. An example for illustrating Definition 8

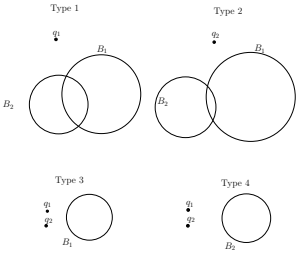


Fig. 4. Four types of configurations for a pair of points and a pair of spheres not forming a 2-Match

Lemma 7. *For any pair of positive numbers* x *and* y *satisfying the conditions of* $x \leq b$, $y \leq b$, *and* $x + y \leq t$, *there exists a pair of* (β_1^i, β_2^i) *in the* (t, δ, b)*-sequence such that* $x \leq \beta_1^i$ *and* $y \leq \beta_2^i$.

Proof. It follows directly from the definition of (t, δ, b)-sequence and Figure 3. □

Definition 9 (2-Match). *Given two points* q_1 *and* q_2, *and two spheres* B_1 *and* B_2, *the points and spheres form a 2-Match if* q_1 *is covered by* B_1 *and* q_2 *is covered by* B_2, *or vice verse.*

If q_1, q_2 and B_1, B_2 do not form a 2-match, it is easy to see that their configuration falls in one or more of the following 4 types (see Figure 4).

- Type1: q_1 is outside of both B_1 and B_2.
- Type2: q_2 is outside of both B_1 and B_2.
- Type3: Both q_1 and q_2 are outside of B_1.
- Type4: Both q_1 and q_2 are outside of B_2.

Note that in the above classification, we do not care about the position of the other point in Types 1 and 2. It may or may not be inside any of the two spheres. For Types 3 and 4, we do not care about the other sphere. It may or may not contain any of the two points.

We assume that the optimal solution for the two angles are α_1 and α_2. The following algorithm uses the concept of 2-Match to determine whether a guess (β_1, β_2) on the values of α_1 and α_2 is correct (i.e. to determine whether $\alpha_1 \leq \beta_1 \leq \frac{\pi}{2}$ and $\alpha_2 \leq \beta_2 \leq \frac{\pi}{2}$). Our main idea of the algorithm is that if the guess (β_1, β_2) is correct, then the optimal chromatic partition U_1 and U_2 can be

covered by two spheres of radii $(1+\epsilon)\sin\beta_1$ and $(1+\epsilon)\sin\beta_2$ respectively. Thus, we can convert the problem of determining the correctness of a guess on α_1 and α_2 to a problem of determining whether there exists a chromatic partition whose two sets can be covered by spheres of the above radii. To check the existence, we build a tree T of height $2\lceil\frac{2}{\epsilon}\rceil$ and use two sets, S_1 and S_2, to guess how U_1 and U_2 partition each point-set G_j. In each level of the tree, there will be at least one guess that is correct about U_1 and U_2.

Guess-Verifier Algorithm
Input: A set of point-sets $\mathcal{G} = \{G_1, G_2, \ldots, G_n\}$ with each containing 2 points in the first quadrant of the R^d space, an $\epsilon \in (0, 1]$, and a guess (β_1, β_2) on α_1 and α_2 with $\beta_1 < \frac{\pi}{2}$ and $\beta_2 < \frac{\pi}{2}$.
Output: An answer on whether $\alpha_1 \leq \beta_1$ and $\alpha_2 \leq \beta_2$.

1. Initialize 2 empty sets S_1 and S_2. Put p_1^1 into S_1 and p_2^1 into S_2. Let $B_1(c_1, r_1)$ and $B_2(c_2, r_2)$ be the MinES of S_1 and S_2 respectively.
2. Build a tree T with root u, associate u with S_1, S_2, $B_1(c_1, r_1)$ and $B_2(c_2, r_2)$, and recursively construct the tree in the following way until no node is growable.
3. Let v be the current node of T.
   (a) If for each G_i, $i = 1, 2, \cdots, n$, p_1^i, p_2^i and $B(c_1, (1+\epsilon)\sin\beta_1)$, $B(c_2, (1+\epsilon)\sin\beta_2)$ are a 2-Match, return 'yes', along with $B(c_1, (1+\epsilon)\sin\beta_1)$ and $B(c_2, (1+\epsilon)\sin\beta_2)$.
   (b) If the height of v is $2\lceil\frac{2}{\epsilon}\rceil$, v is a leaf node and go to another growable node.
   (c) If $r_1 > (1+\epsilon)\sin\beta_1$ or $r_2 > (1+\epsilon)\sin\beta_2$, v is a leaf node and go to another growable node.
   (d) Choose a point-set G_j which does not form a 2-Match with $B(c_1, (1+\epsilon)\sin\beta_1)$ and $B(c_2, (1+\epsilon)\sin\beta_2)$, determine the type of their configuration. If they belong to multiple types, arbitrarily select one as their type and perform the following operation accordingly.

   Type 1: p_1^j is outside of the two spheres. Create a left and a right child for v, and let them inherit S_1, S_2, $B_1(c_1, r_1)$ and $B_2(c_2, r_2)$ from v. For the left child, add p_1^j into S_1 and re-compute $B_1(c_1, r_1)$; for the right child, add p_1^j into S_2 and re-compute $B_2(c_2, r_2)$. For each child, if its $|S_1| > \lceil\frac{2}{\epsilon}\rceil$, $|S_2| > \lceil\frac{2}{\epsilon}\rceil$, or S_1 and S_2 violate the chromatic condition, it will not be created.

   Type 2: Perform a similar operation as in Type 1.

   Type 3: Both p_1^j and p_2^j are outside of B_1. Create a left and a right child for v, and let them inherit S_1, S_2, $B_1(c_1, r_1)$ and $B_2(c_2, r_2)$ from v. For the left child, add p_1^j into S_1, and re-compute $B_1(c_1, r_1)$; for the right child, add p_2^j into S_1, and re-compute $B_1(c_1, r_1)$. For each child, if its $|S_1| > \lceil\frac{2}{\epsilon}\rceil$ or S_1 and S_2 violate the chromatic condition, it will not be created.

   Type 4 Perform a similar operation as in Type 3.
4. Return 'no'.

Lemma 8. *The above algorithm correctly returns a "yes" answer in* $O((nd + 2^{\lceil \frac{2}{\epsilon} \rceil} d^3)\ 2^{2\lceil \frac{2}{\epsilon} \rceil})$ *time, if* $\alpha_1 \leq \beta_1 \leq \frac{\pi}{2}$ *and* $\alpha_2 \leq \beta_2 \leq \frac{\pi}{2}$.

For the limited of space, we omit the proof for lemma 8, the rough idea is: We can consider the Guess-Verifier Algorithm as finding $2\lceil \frac{2}{\epsilon} \rceil$ points one by one. In each step, there are two requirements, one for preserving Inequalities (1) and (2), the other for satisfying the chromatic condition. If $\alpha_1 \leq \beta_1 \leq \frac{\pi}{2}$ and $\alpha_2 \leq \beta_2 \leq \frac{\pi}{2}$, the algorithm will find such $2\lceil \frac{2}{\epsilon} \rceil$ points.

With the above algorithm and lemma, we can now present our PTAS algorithm for the 2-CCC problem.

2-CCC Algorithm
Input: A set of point-sets $\mathcal{G} = \{G_1, G_2, \ldots, G_n\}$ with each containing 2 points in the first quadrant of the R^d space and on the unit sphere centered at the origin, three constant $\epsilon_1, \epsilon_2, \epsilon_3 \in (0, 1]$.
Output: An approximation solution to the 2-CCC problem.

1. Use the algorithm in Lemma 6 to obtain upper and lower bounds, L_1 and L_2, for the optimal solution with $\frac{L_1}{L_2}$ being a constant.
2. Let $R_{low} = L_2, R_{up} = L_1$. Repeat Steps 3 and 4 until $R_{up} - R_{low} \leq \epsilon_3 L_2$.
3. Let $R = \frac{R_{low} + R_{up}}{2}$. Construct a $(R, \epsilon_2, \arcsin\sqrt{1 - \frac{1}{d}})$-Sequence.
4. For each pair of guessing angles in the $(R, \epsilon_2, \arcsin\sqrt{1 - \frac{1}{d}})$-Sequence and ϵ_1, run the Guess-Verifier Algorithm. If there is one guess leading to a "yes" answer, let $R_{up} = R$ and keep the returning two spheres from the Guess-verifier Algorithm. Otherwise (i.e., no "yes" answer), let $R_{low} = R$. Go to Step 3.
5. Output the two cones corresponding to the two spheres from Guess-Verifier Algorithm returned along with the last "yes" answer.

By Lemmas 5 to 8, we immediately have the following theorem.

Theorem 3. *The 2-CCC Algorithm generates a* $(1 + \frac{\theta}{\frac{\pi}{2} - \theta})(1 + \epsilon_2)(1 + \epsilon_3)$*-approximation solution in* $O(\frac{1}{\epsilon_2} \log \frac{1}{\epsilon_3}(nd + 2^{\lceil \frac{2}{\epsilon_1} \rceil} d^3) 2^{2\lceil \frac{2}{\epsilon_1} \rceil})$ *time, where* θ *is determined by the equation* $\sin\theta = \frac{\sqrt{2\epsilon_1 + \epsilon_1^2}}{1 + \epsilon_1}$. *Furthermore, if the angle of each optimal cone is upper bounded by a small constant* β*, the approximation ratio improves to* $(1 + \frac{\tan\beta}{\beta} \frac{1}{\sqrt{1 - (2\epsilon_1 + \epsilon_1^2)\tan^2\beta}} \epsilon_1)\ (1 + \epsilon_2)(1 + \epsilon_3)$ *within the same time bound, and angle of each returning cone is no more than* $(1 + \frac{\tan\beta}{\beta} \frac{1}{\sqrt{1 - (2\epsilon_1 + \epsilon_1^2)\tan^2\beta}} \epsilon_1)\beta$.

3.3 A $(1 + \epsilon)$-Approximation Algorithm for $k > 2$

For the case of $k > 2$, we first extend the constant ratio approximate algorithm for $k = 2$ to get a upper and lower bounds. Then, we generalize the concepts of (t, δ, b)-Sequence to (t, δ, b)-Net and 2-Match to k-Match, and show that there exists a similar Guess-Verifier algorithm for this case. With this algorithm, we

perform a binary search between the upper and lower bounds to obtain a $(1+\epsilon)$-approximation solution to the CCC problem. Below is the main theorem of our algorithm.

Theorem 4. *When $k > 2$, it is possible to obtain a $(1 + \frac{\theta}{\frac{\pi}{2}-\theta})(1+\epsilon_2)(1+\epsilon_3)$-approximation for the CCC problem in $O(\binom{\lceil \frac{k-1}{\epsilon_2} \rceil}{k-1}(\log \frac{1}{\epsilon_3} + k \log k)(nk^2 d + nk^{2.5} + 2^{\lceil \frac{2}{\epsilon_1} \rceil} d^3)(k^2-(k+1))^{k(\lceil \frac{2}{\epsilon_1} \rceil - 1)})$ time, where θ satisfies the equation $\sin\theta = \frac{\sqrt{2\epsilon_1+\epsilon_1^2}}{1+\epsilon_1}$. Furthermore, if the angle of each optimal cone is upper bounded by a small constant β, the approximation ratio improves to*

$$(1 + \frac{\tan\beta}{\beta} \frac{1}{\sqrt{1-(2\epsilon_1+\epsilon_1^2)\tan^2\beta}} \epsilon_1)(1+\epsilon_2)(1+\epsilon_3).$$

Acknowledgments. The authors would like to thank Professor Pankaj K. Agarwal for very helpful discussion and suggestions on this problem.

References

1. Abellanas, M., Hurtado, F., Icking, C., Klein, R., Langetepe, E., Ma, L., Palop, B., Sacristan, V.: The Farthest Color Voronoi Diagram and Related Problems. In: Abstracts 17th European Workshop Comput. Geom. (2001)
2. Agarwal, P.K., Har-Peled, S., Varadarajan, K.R.: Geometric Approximation via Coresets. Combinatorial and Computational Geometry 52, 1–30 (2005)
3. Badoiu, M., Clarkson, K.: Smaller core-sets for balls. In: Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 801–802 (2003)
4. Badoiu, M., Har-Peled, S., Indyk, P.: Approximate clustering via core-sets. In: Proceedings of the 34th Symposium on Theory of Computing, pp. 250–257 (2002)
5. Cheong, O., Everett, H., Glisse, M., Gudmundsson, J., Hornus, S., Lazard, S., Lee, M., Na, H.: Farthest-Polygon Voronoi Diagrams. In: Proceedings of the 15th Annual European Conference on Algorithms, pp. 407-418 (2007)
6. Clarkson, K.: Coresets, sparse greedy approximation, and the Frank-Wolfe algorithm. In: Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 922–931 (2008)
7. Huttenlocher, D.P., Kedem, K., Sharir, M.: The Upper Envelope of Voronoi Surfaces and its Application. In: Proceedings of the Seventh Annual Symposium on Computational Geometry, pp. 194–203 (1991)
8. Kumar, P., Mitchell, J., Yildirim, A.: Computing Core-Sets and Approximate Smallest Enclosing Hyperspheres in High Dimensions (2002) (manuscript)
9. Megiddo, N.: On the Complexity of Some Geometric Problems in Unbounded Dimension. J. Symb. Comput. 10, 327–334 (1990)
10. Panigrahy, R.: Minimum enclosing polytope in high dimensions, CoRR cs.CG/0407020 (2004)
11. Micali, S., Vazirani, V.V.: An $O(\sqrt{|V|}|E|)$ algorithm for finding maximum matching in general graphs. In: Proceedings of 21st IEEE Symp. Foundations of Computer Science, pp. 17–27 (1980), doi:10.1109/SFCS

Clustering with Local Restrictions

Daniel Lokshtanov[1] and Dániel Marx[2,⋆]

[1] University of California, San Diego, USA
dlokshtanov@cs.ucsd.edu
[2] Humboldt-Universität zu Berlin, Berlin, Germany
dmarx@cs.bme.hu

Abstract. We study a family of graph clustering problems where each cluster has to satisfy a certain local requirement. Formally, let μ be a function on the subsets of vertices of a graph G. In the (μ, p, q)-Partition problem, the task is to find a partition of the vertices into clusters where each cluster C satisfies the requirements that (1) at most q edges leave C and (2) $\mu(C) \leq p$. Our first result shows that if μ is an *arbitrary* polynomial-time computable monotone function, then (μ, p, q)-Partition can be solved in time $n^{O(q)}$, i.e., it is polynomial-time solvable *for every fixed* q. We study in detail three concrete functions μ (number of nonedges in the cluster, maximum number of non-neighbours a vertex has in the cluster, the number of vertices in the cluster), which correspond to natural clustering problems. For these functions, we show that (μ, p, q)-Partition can be solved in time $2^{O(p)} \cdot n^{O(1)}$ and in randomized time $2^{O(q)} \cdot n^{O(1)}$, i.e., the problem is fixed-parameter tractable parameterized by p or by q.

1 Introduction

Partitioning objects into clusters or similarity classes is an important task in various applications such as data mining, facility location, interpreting experimental data, VLSI design, and many more. The partition has to satisfy certain constraints: typically, we want to ensure that objects in a cluster are "close" or "similar" to each other and/or objects in different clusters are "far" or "dissimilar." Additionally, we may want to partition the data into a certain prescribed number k of clusters, or we may have upper/lower bounds on the size of the clusters. Different objectives and different distance/similarity measures give rise to specific combinatorial problems.

Correlation clustering [14,1,3,15] deals with a specific form of similarity measure: for each pair of objects, we know that either they are similar or dissimilar. This means that the similarity information can be expressed as an undirected graph, where the vertices represent the objects and similar objects are adjacent. In the ideal situation every connected component of the graph is a clique, in which case the components form a clustering that completely agrees with the

⋆ Research supported by the Alexander von Humboldt Foundation and OTKA grant 67651.

L. Aceto, M. Henzinger, and J. Sgall (Eds.): ICALP 2011, Part I, LNCS 6755, pp. 785–797, 2011.

similarity information. However, due to inconsistencies in the data or experimental errors, such a perfect partitioning might not always be possible. The goal in correlation clustering is to partition the vertices into an arbitrary number of clusters in a way that agrees with the similarity information as much as possible: we want to minimize the number of pairs for which the clustering disagrees with the input data (i.e., similar pairs that are put into different clusters, or dissimilar pairs that are clustered together).

In many cases, such as in variants of the correlation clustering problem defined in the previous paragraph, the objective is to minimize the total error of the solution. Thus the goal is to find a solution that is good in a global sense, but this does not rule out the possibility that the solution contains clusters that are very bad. In this paper, the opposite approach is taken: we want to find a partition where each cluster is "good" in a certain local sense. This means that the partition has to satisfy a set of local constraints on each cluster, but we do not try to optimize the total fitness of clusters.

The setting in this paper is the following. We want to partition the graph into an arbitrary number of clusters such that (1) at most q edges leave each cluster, and (2) each cluster induces a graph that is "cluster-like." Defining what we mean by the abstract notion of cluster-like gives rise to a family of concrete problems. Formally, let μ be a function that assigns a nonnegative integer to each subset of vertices in the graph and let us require $\mu(X) \leq p$ for every cluster X of the partition. There are many reasonable choices for the measure μ that correspond to natural problems. In particular, in this paper we will obtain concrete results for the following three measures:

1. $\mathsf{nonedge}(X)$: number of nonedges induced by X,
2. $\mathsf{nondeg}(X)$: maximum degree of the *complement* of the graph induced by X.
3. $\mathsf{size}(X) = |X|$: number of vertices of X.

The first two functions express that each cluster should induce a graph that is close to being a clique. The third function only requires that each cluster is small. For a given function μ and integers p and q, we denote by (μ, p, q)-Partition the problem of partitioning the vertices into clusters such that at most q edges leave each cluster and $\mu(X) \leq p$ for every cluster.

Our first result is very simple yet powerful. Let μ be a function satisfying the mild technical conditions that it is polynomial-time computable and monotone (i.e., if $X \subseteq Y$, then $\mu(X) \leq \mu(Y)$). Observe that for example all three functions defined above satisfy these conditions. Our first result shows that for *every function* μ satisfying these conditions and *every fixed integer* q, the problem (μ, p, q)-Partition can be solved in polynomial time (the value p is considered to be part of the input). For example, it can be decided in polynomial time if there is a clustering where at most 13 edges leave each cluster and each cluster induces at most 27 nonedges (or even the more general question, where the maximum number p of nonedges is given in the input). This might be surprising: we believe that most people would guess that this problem is NP-hard. The algorithm is based on a simple application of uncrossing of posimodular functions and on the fact that for fixed q we can enumerate every (connected) cluster with

at most q outgoing edges. The crucial observation is that if every vertex can be *covered* by a good cluster, then the vertices can be *partitioned* into good clusters. Thus the problem boils down to checking if a given v is contained in a suitable cluster.

While the algorithm is simple in hindsight, considerable efforts have been spent on solving some very particular special cases. For example, Heggernes et al. [9] gave a polynomial-time algorithm for $(\mathsf{nonedge}, 1, 3)$-Partition and Langston and Plaut [10] argued that the very deep results of Robertson and Seymour on graph minors and immersions imply that (size, p, q)-Partition is polynomial-time solvable for every fixed p and q. These results follow as straightforward corollaries from our first result.

Although this simple algorithm is polynomial for every fixed q, the running time is about $n^{O(q)}$, thus it is not efficient even for small values of q. To improve the running time, we look at the problem from the viewpoint of parameterized complexity. We show that for several natural measures μ, including the three defined above, the clustering problem can be solved in randomized time $2^{O(q)} \cdot n^{O(1)}$, that is, the problem is fixed-parameter tractable (FPT) parameterized by the bound q on the number of edges leaving a cluster. Moreover, the bound p can be assumed to be part of the input. Thus this algorithm can be efficient for small values of q (say, $O(\log n)$) even if p is large. The algorithm has constant probability of error, but it can be derandomized at the cost of worse dependence on q in the running time (details will appear in the full version). The problem (size, p, q)-Partition appears in the open problem list of the 1999 monograph of Downey and Fellows [8] under the name "Minimum Degree Partition," where it is suggested that the problem is probably W[1]-hard parameterized by q. Our result answers this question by showing that the problem is FPT, contrary to the expectation of Downey and Fellows.

A crucial ingredient of our parameterized algorithm is the notion of *important separators,* which has been used (implicitly or explicitly) to obtain fixed-parameter tractability results for various cut or separator related problems. In particular, we use the "randomized selection of important sets" argument that was introduced very recently in [13] to prove the fixed-parameter tractability of (edge and vertex) multicut. With these tools at hand, we can reduce (μ, p, q)-Partition to a special case that we call the "Satellite Problem." We show that if the Satellite Problem is fixed-parameter tractable parameterized by q for a particular function μ, then (μ, p, q)-Partition is also fixed-parameter tractable parameterized by q. It seems that for many reasonable functions μ, the Satellite Problem can be solved by dynamic programming techniques. In particular, this is true for the three functions defined above, and this results in randomized algorithms with running time $2^{O(q)} \cdot n^{O(1)}$. Note that the reduction to the Satellite Problem works for every monotone μ, and we need arguments specific to a particular μ only in the algorithms for Satellite Problem.

2 Clustering and Uncrossing

Given an undirected graph G, we denote by $\Delta(X)$ the set of edges between X and $V(G) \setminus X$, and define $d(X) = |\Delta(X)|$. We will use two well-known and easily checkable properties of the function d: for $X, Y \subseteq V(G)$, d satisfies the *submodular* and *posimodular* inequalities

$$d(X) + d(Y) \geq d(X \cap Y) + d(Y \cup X) \text{ and } d(X) + d(Y) \geq d(X \setminus Y) + d(Y \setminus X).$$

Let $\mu : 2^{V(G)} \to \mathbb{Z}^+$ be a function assigning nonegative integers to sets of vertices of G. Let p and q be two integers. We say that a set $C \subseteq V(G)$ is a (μ, p, q)-*cluster* if $\mu(C) \leq p$ and $d(C) \leq q$. A (μ, p, q)-*partition* of G is a partition of $V(G)$ into (μ, p, q)-clusters. The main problem considered in this paper is finding such a partition. A necessary condition for the existence of (μ, p, q)-partition is that for every vertex $v \in V(G)$ there is a (μ, p, q)-cluster that contains v. Therefore, we are also interested in the problem of finding a cluster containing a vertex v.

(μ, p, q)-PARTITION	(μ, p, q)-CLUSTER
Input: A graph G, integers p, q.	Input: Graph G, integers p, q, vertex v.
Find: A (μ, p, q)-partition of G.	Find: A (μ, p, q)-cluster C containing v.

The main observation of this section is that if μ is *monotone* (i.e., $\mu(X) \leq \mu(Y)$ for every $X \subseteq Y$), then this is actually a sufficient condition. Therefore, in these cases, it is sufficient to solve (μ, p, q)-CLUSTER.

Lemma 1. *Let G be a graph, let $p, q \geq 0$ be two integers, and let $\mu : 2^{V(G)} \to \mathbb{Z}^+$ be a monotone function. If every $v \in V(G)$ is contained in some (μ, p, q)-cluster, then G has a (μ, p, q)-partition, and given a set of (μ, p, q)-clusters $C_1, \ldots, C_n$ whose union is $V(G)$, a (μ, p, q)-partition can be found in polynomial time.*

Proof. Let us consider a collection $C_1, \ldots, C_n$ of (μ, p, q)-clusters whose union is $V(G)$. If the sets are pairwise disjoint, then they form a partition of $V(G)$ and we are done. If $C_i \subseteq C_j$, then the union remains $V(G)$ even after throwing away C_i. Thus we can assume that no set is contained in another. Suppose that C_i and C_j intersect. Now either $d(C_i) \geq d(C_i \setminus C_j)$ or $d(C_j) \geq d(C_j \setminus C_i)$ must be true: it is not possible that both $d(C_i) < d(C_i \setminus C_j)$ and $d(C_j) < d(C_j \setminus C_i)$ hold, as this would violate the posimodularity of d. Suppose that $d(C_j) \geq d(C_j \setminus C_i)$. Now the set $C_j \setminus C_i$ is also a (μ, p, q)-cluster: we have $d(C_j \setminus C_i) \leq d(C_j) \leq q$ by assumption and $\mu(C_j \setminus C_i) \leq \mu(C_j) \leq p$ from the monotonicity of μ. Thus we can replace C_j by $C_j \setminus C_i$ in the collection: the union of the clusters is still $V(G)$. Similarly, if $d(C_j) \geq d(C_j \setminus C_i)$, then we can replace C_j by $C_j \setminus C_i$.

Repeating these steps (throwing away subsets and resolving intersections), we eventually arrive at a pairwise disjoint collection of (μ, p, q)-clusters. Each step decreases the number of cluster pairs C_i, C_j that have non-empty intersection. Therefore, this process terminates after a polynomial number of steps. □

In light of Lemma 1, it is sufficient to find a (μ, p, q)-cluster C_v for each vertex $v \in V(G)$. If there is a vertex v for which there is no such cluster C_v, then obviously there is no (μ, p, q)-partition; if we have such a C_v for every vertex v, then Lemma 1 gives us a (μ, p, q)-partition in polynomial time. For fixed q, (μ, p, q)-Cluster can be solved by brute force if μ is polynomial-time computable:enumerate every set F of at most q edges and check if the component of $G \setminus F$ containing v is a (μ, p, q)-cluster. If C_v is a (μ, p, q)-cluster containing v, then we find it when $F = \Delta(C_v)$ is considered by the enumeration procedure.

Theorem 2. *Let μ be a polynomial-time computable monotone function. Then for every fixed q, there is an $n^{O(q)}$ time algorithm for (μ, p, q)-*Partition.

As we have seen, an algorithm for (μ, p, q)-Cluster gives us an algorithm for (μ, p, q)-Partition. In the rest of the paper, we devise more efficient algorithms for (μ, p, q)-Cluster than the $n^{O(q)}$ time brute force method described above.

3 Parameterization by q

The main result of this section is that (μ, p, q)-Partition is (randomized) FPT parameterized by q for the three functions nonedge, nondeg, and size.

Theorem 3. *There is an algorithm for* (size, p, q)-Partition, $(\mathsf{nonedge}, p, q)$-Partition *and* (nondeg, p, q)-Partition *using* $2^{O(q)}|V(G)|^{O(1)}$ *randomized time. If the input instance is a* yes*-instance the algorithm incorrectly returns* no *with probability less than* $\frac{1}{2}$*. On* no*-instances the algorithm always answers* no*.*

By Lemma 1, all we need to show is that (μ, p, q)-cluster is fixed-parameter tractable parameterized by q. We introduce a somewhat technical variant of this question, the Satellite Problem, and show that for *every monotone function* μ, if Satellite Problem is FPT, then (μ, p, q)-cluster is FPT as well. Thus we need arguments specific to a particular μ only for the Satellite Problem.

Satellite Problem	
Input:	A graph G, integers p, q, a vertex $v \in V(G)$, a partition V_0, V_1, $\ldots$, V_n of $V(G)$ such that $v \in V_0$ and there is no edge between V_i and V_j for any $1 \leq i < j \leq n$.
Find:	A (μ, p, q)-cluster C with $V_0 \subseteq C$ such that for every $1 \leq i \leq n$, either $C \cap V_i = \emptyset$ or $V_i \subseteq C$.

That is, for every V_i, we have to decide whether to include or exclude it from the solution C. If we exclude V_i from C, then $d(C)$ increases by the number of edges between V_0 and V_i. If we include V_i into C, then $\mu(C)$ increases accordingly. Thus we need to solve the knapsack-like problem of including sufficiently many V_i such that $d(C) \leq q$, but not including too many to ensure $\mu(C) \leq p$. As we shall see in Section 3.3, in many cases this problem can be solved by dynamic programming (and some additional arguments). The important fact that we use is that there are no edges between V_i and V_j, thus for many reasonable functions

μ, the way $\mu(C)$ increases by including V_i is fairly independent from whether V_j is included in C or not.

The reduction to Satellite Problem uses the concept of important separators (Section 3.1). The reduction itself is given in Section 3.2. In Section 3.3, we show how the Satellite Problem can be solved for the three functions nonedge, nondeg, size.

3.1 Important Separators and Important Sets

The notion of *important separators* was introduced in [12] to prove the fixed-parameter tractability of multiway cut problems. This notion turned out to be useful in other applications as well [5,6,17]. The basic idea is that in many problems where terminals need to be separated in some way, it is sufficient to consider separators that are "as far as possible" from one of the terminals. Let s, t be two vertices of a graph G. An $s-t$ *separator* is a set $S \subseteq E(G)$ of edges separating s and t, i.e., there is no $s-t$ path in $G \setminus S$. An $s-t$separator is *inclusionwise minimal* if there is an $s-t$ path in $G \setminus S'$ for every $S' \subset S$.

Definition 4. *Let $s, t \in V(G)$ be vertices, $S \subseteq E(G)$ be an $s-t$ separator, and let K be the component of $G \setminus S$ containing s. We say that S is an* important $s-t$ separator *if it is inclusionwise minimal and there is no $s-t$ separator S' with $|S'| \leq |S|$ such that $K \subset K'$ for the component K' of $G \setminus S'$ containing s.*

We now define *important sets*, which are natural companions to important separators.

Definition 5. *We say that a set $X \subseteq V(G)$, $v \notin X$ is* important *if (1) $d(X) \leq q$, (2) $G[X]$ is connected and (3) there is no $Y \supset X$, $v \notin Y$ such that $d(Y) \leq d(X)$ and $G[Y]$ is connected.*

It is easy to see that X is an important set if and only if $\Delta(X)$ is an important $u-v$ separator for every $u \in X$. As there are differences between edge and vertex separators, and some of the results appear only implicitly in previous papers, the full version of this article [11] contains proofs of Theorem 6 and Lemma 7. Since X is an important set if and only if $\Delta(X)$ is an important $u-v$ separator, we can use Theorem 6 and Lemma 7 to enumerate important sets.

Theorem 6 ($\star$). [1] *Let $s, t \in V(G)$ be two vertices in graph G. For every $k \geq 0$, there are at most 4^k important $s-t$ separators of size at most k. Furthermore, these important separators can be enumerated in time $4^k \cdot n^{O(1)}$.*

Lemma 7 ($\star$). *Let $s, t \in V(G)$. If $\mathcal{S}$ is the set of all important $s-t$ separators, then $\sum_{S \in \mathcal{S}} 4^{-|S|} \leq 1$. Thus $\mathcal{S}$ contains at most 4^k separators of size at most k.*

3.2 Reduction to the Satellite Problem

In this section we reduce (μ, p, q)-Cluster to the Satellite Problem.

[1] Proofs of results labelled with $\star$ have been omitted due to space restrictions.

Lemma 8. *If* Satellite Problem *can be solved in time* $f(q) \cdot n^{O(1)}$ *for some monotone* μ*, then there is a randomized* $2^{O(q)} \cdot f(q) \cdot n^{O(1)}$ *algorithm with constant error probability that finds a* (μ, p, q)*-cluster containing* v *(if one exists).*

The following lemma establishes the connection between important sets and finding (μ, p, q)-clusters: we can assume that the components of $G \setminus C$ for the solution C are important sets. In Lemma 10, we show that by randomly choosing important sets, with some probability we can obtain an instance of the Satellite Problem where $V_1, \ldots, V_n$ contain all the components of $G \setminus C$. This gives us the reduction stated in Lemma 8 above.

Lemma 9. *Let* C *be an inclusionwise minimal* (μ, p, q)*-cluster containing* v*. Then every component of* $G \setminus C$ *is an important set.*

Proof. Let X be a component of $G \setminus C$. It is clear that X satisfies the first two properties of Definition 5 (note that $\Delta(X) \subseteq \Delta(C)$). Thus let us suppose that there is a $Y \supset X$, $v \notin Y$ such that $d(Y) \leq d(X)$ and $G[Y]$ is connected. Let $C' := C \setminus Y$. Note that C' is a proper subset of C: every neighbor of X is in C, thus a connected superset of X has to contain at least one vertex of C. It is easy to see that C' is a (μ, p, q)-cluster: we have $\Delta(C') \subseteq (\Delta(C) \setminus \Delta(X)) \cup \Delta(Y)$ and therefore $d(C') \leq d(C) - d(X) + d(Y) \leq d(C) \leq q$ and $\mu(C') \leq \mu(C) \leq p$ (by the monotonicity of μ). This contradicts the minimality of C. □

Lemma 10. *Given a graph* G*, vertex* $v \in V(G)$*, integers* p*,* q*, and a monotone function* $\mu : 2^{V(G)} \to \mathbb{Z}^+$*, we can construct in time* $2^{O(q)} \cdot n^{O(1)}$ *an instance* I *of the* Satellite Problem *such that*

- *If some* (μ, p, q)*-cluster contains* v*, then* I *is a yes-instance with probability* $2^{-O(q)}$*,*
- *If there is no* (μ, p, q)*-cluster containing* v*, then* I *is a no-instance.*

Proof. For every $u \in V(G)$, $u \neq v$, let us use the algorithm of Lemma 7 to enumerate every important $u - v$ separator of size at most q. For every such separator S, let us put the component K of $G \setminus S$ containing u into the collection $\mathcal{S}$. Note that a component K can be obtained for more than one vertex u, but we put only one copy into $\mathcal{S}$.

Let $\mathcal{S}'$ be a subset of $\mathcal{S}$, where each member K of $\mathcal{S}$ is chosen with probability $2^{-d(K)}$ independently at random. Let Z be the union of the sets in $\mathcal{S}'$, let $V_1, \ldots, V_n$ be the connected components of $G[Z]$, and let $V_0 = V(G) \setminus Z$. It is clear that $V_0, V_1, \ldots, V_n$ give an instance I of Satellite Problem, and a solution for I gives a (μ, p, q)-cluster containing v. Thus we only need to show that if there is a (μ, p, q)-cluster C containing v, then I is a yes-instance with probability $2^{-O(q)}$.

Let C be an inclusionwise minimal (μ, p, q)-cluster containing v. Let B be the vertices on the boundary of C, i.e., the vertices of C incident to $\Delta(C)$. Let $K_1, \ldots, K_t$ be the components of $G \setminus C$. Note that every edge of $\Delta(C)$ enters some K_i, thus $\sum_{i=1}^{t} d(K_i) = d(C) \leq q$. By Lemma 9, every K_i is an important set, and hence it is in $\mathcal{S}$. Consider the following two events:

(1) Every component K_i of $G \setminus C$ is in $\mathcal{S}'$ (and hence $K_i \subseteq Z$).
(2) $Z \cap B = \emptyset$.

The probability that (1) holds is $\prod_{i=1}^{t} 4^{-d(K_i)} = 4^{-\sum_{i=1}^{t} d(K_i)} \geq 4^{-q}$. Event (2) holds if for every $b \in B$, no set $K \in \mathcal{S}$ with $b \in K$ is selected into $\mathcal{S}'$. It follows directly from the definition of important separators that for every $K \in \mathcal{S}$ with $b \in K$, $\Delta(K)$ is an important $b - v$ separator. Thus by Lemma 7, $\sum_{K \in \mathcal{S}, b \in K} 4^{-|d(K)|} \leq 1$. The probability that $Z \cap B = \emptyset$ can be bounded by

$$\prod_{K \in \mathcal{S}, K \cap B \neq \emptyset} (1 - 4^{-d(K)}) \geq \prod_{b \in B} \prod_{K \in \mathcal{S}, b \in K} (1 - 4^{-d(K)}) \geq \prod_{b \in B} \prod_{K \in \mathcal{S}, b \in K} \exp\left(\frac{-4^{-d(K)}}{(1 - 4^{-d(K)})}\right)$$

$$\geq \prod_{b \in B} \prod_{K \in \mathcal{S}, b \in K} \exp(-\frac{4}{3} \cdot 4^{-d(K)}) = \prod_{b \in B} \exp\left(-\frac{4}{3} \cdot \sum_{K \in \mathcal{S}, b \in K} 4^{-d(K)}\right) \geq (e^{-\frac{4}{3}})^{|B|} \geq e^{-4q/3}.$$

In the first inequality, we use that every term is less than 1 and every term on the right hand side appears at least once on the left hand side; in the second inequality, we use that $1 + x \geq \exp(x/(1 + x))$ for every $x > -1$. Events (1) and (2) are independent: (1) is a statement about the selection of subsets of $\mathcal{S}$ that are disjoint from B, while (2) involves only sets intersecting B. Thus by probability $2^{-O(q)}$, both (1) and (2) hold.

Suppose that both (1) and (2) hold, we show that instance I of the Satellite Problem is a yes-instance. In this case, every component K_i of $G \setminus C$ is a component V_j of $G[Z]$: $K_i \subseteq Z$ by (1) and every neighbor of K_i is outside Z. Thus C is a solution of I, as it can be obtained as the union of V_0 and some components of $G[Z]$. □

3.3 Solving the Satellite Problem

In this section, we give efficient algorithms for solving the Satellite Problem when the function μ is size, nonedge and nondeg. We describe the three algorithms by increasing difficulty. In the case when μ is size, solving the Satellite Problem turns out to be equivalent to the classical Knapsack problem with polynomial bounds on the values and weights of the items.

Recall that the input to the Satellite Problem is a graph G, integers p, q, a vertex $v \in V(G)$, a partition $V_0, V_1, \ldots, V_n$ of $V(G)$ such that $v \in V_0$ and there is no edge between V_i and V_j for any $1 \leq i < j \leq n$. The task is to find a vertex set C, such that $C = V_0 \cup \bigcup_{i \in S} V_i$ for a subset S of $\{1, \ldots, n\}$ and C satisfies $d(C) \leq q$ and $\mu(C) \leq p$. For a subset S of $\{1, \ldots, n\}$ we define $C(S) = V_0 \cup \bigcup_{i \in S} V_i$.

Lemma 11. *The* Satellite Problem *for measure* size *can be solved in time* $O(q|V(G)| \log |V(G)|)$.

Proof. Notice that $d(C) = d(V_0) - \sum_{i \in S} d(V_i)$. Hence, we can reformulate the Satellite Problem with $\mu =$ size as finding a subset S of $\{1, \ldots, n\}$ such that

$\sum_{i\in S} d(V_i) \geq d(V_0) - q$ and $\sum_{i\in S} |V_i| \leq p - |V_0|$. Thus, we can associate with every i an item with value $d(V_i)$ and weight $|V_i|$. The objective is to find a set of items with total value at least $d(V_0) - q$ and total weight at most $p - |V_0|$. This problem is known as KNAPSACK and can be solved in $O(nv \log w)$ time by a classical dynamic programming [4,7] algorithm, where n is the number of items, v is the value we seek to attain and w is the weight limit. Since the value is bounded from above by q and the weight by $|V(G)|$, the statement of the lemma follows. □

The case that $\mu =$ nonedge is slightly more complicated, however we can still solve it using a dynamic programming algorithm. For the version of SATELLITE PROBLEM when $\mu =$ nondeg we do not have a polynomial time algorithm. Instead, we give a $2^q|V(G)|^{O(1)}$ time randomized algorithm.

Lemma 12 (⋆). *The* SATELLITE PROBLEM *for* nonedge *can be solved in time* $O(pn|E(G)||V(G)|)$. *There is a randomized algorithm which given an instance of* nondeg-SATELLITE PROBLEM *runs in* $2^q|V(G)|^{O(1)}$ *time, correctly answers* no *on all* no*-instances and answers* yes *on* yes*-instances with probability at least* e^{-2q}.

Repeating the algorithm for nondeg-SATELLITE PROBLEM $O(e^{2q})$ times will decrease the probability of false negatives from $1 - e^{-2q}$ to $\frac{1}{2}$. Lemmata 10, 11, and 12 give Theorem 3.

4 Parameterization by p

Theorem 13. *There is a* $8e^{p+o(p)}|V(G)|^{O(1)}$ *time algorithm for the problem* (size, p, q)-PARTITION *and a* $8e^{3p+o(p)}|V(G)|^{O(1)}$ *time algorithm for the problems* $(\mathsf{nonedge}, p, q)$-PARTITION *and* (nondeg, p, q)-PARTITION.

Because of Lemma 1, it is sufficient to solve the corresponding (μ, p, q)-CLUSTER problem within the same time bound. The setting is as follows. We are given a graph G, integers p and q and a vertex v in G. The objective is to find a set C *not* containing v such that $d(C \cup \{v\}) \leq q$ and, depending on which problem we are solving, either $|C \cup \{v\}| = \mathsf{size}(C \cup \{v\}) \leq p$, $\mathsf{nonedge}(C \cup \{v\}) \leq p$ or $\mathsf{nondeg}(C \cup \{v\}) \leq p$.

For a set S and vertex v, define $\Delta(S, v)$ to be the set of edges with one endpoint in S and one in $\{v\}$. Define $\overline{\Delta}(S, v)$ to be $\Delta(S) \setminus \Delta(S, v)$, and let $d(S, v) = |\Delta(S, v)|$ and $\overline{d}(S, v) = |\overline{\Delta}(S, v)|$. We will say that a set C is v*-minimal* if $v \notin C$ and $d(C' \cup \{v\}) > d(C \cup \{v\})$ for every $C' \subset C$. As size, nonedge and nondeg are monotone we can focus on v-minimal sets C. The following fact uses that there are no parallel edges:

Observation 14. Let C be a v-minimal set. Then $\overline{d}(C, v) < d(C, v) \leq |C|$

In particular, if $\overline{d}(C, v) \geq d(C, v)$, then $d(v) \leq d(C \cup \{v\})$, contradicting that C is minimal. Since $\overline{d}(C, v) < |C|$, it follows that C must contain a vertex u such that $N[u] \subseteq C \cup \{v\}$. Now we show that there are not too many v-minimal sets C of size at most p such that $G[C]$ is connected.

Lemma 15. *For any graph G, vertex v and integer p, there are at most $4^p|V(G)|$ v-minimal sets C such that $|C| \leq p$ and $G[C]$ is connected. Furthermore, all such sets can be listed in $O(4^p|V(G)|)$ time.*

Proof. By Observation 14, any v-minimal set C of size at most p satisfies $\overline{d}(C, v) < p$. Let S be a set such that $|S| \leq p$ and $G[S]$ is connected. Let F be a subset of $N(S) \setminus \{v\}$ of size at most $p-1$. We prove by downward induction on $|S|$ and $|F|$ that there are at most $2^{2p-|S|-|F|-1}$ v-minimal sets such that $|C| \leq p$, $G[C]$ is connected, $S \subseteq C$, and $F \cap C = \emptyset$. If $|S| = p$ then the only possibility for C is S, while $2^{2p-|S|-|F|-1} \geq 1$. Similarly, consider the case that $|F| = p-1$. Now, every vertex of F has at least one edge into C and hence $\overline{d}(C, v) = p-1$. Hence $N(C) = F \cup \{v\}$ and the only possibility for C is the connected component of $G \setminus (F \cup \{v\})$ that contains S. Hence there is one possibility for C and $2^{2p-|S|-|F|-1} \geq 1$.

For the inductive step, consider a set S such that $|S| \leq p$ and $G[S]$ is connected and a subset F of $N(S) \setminus \{v\}$ of size at most $p-1$. We want to bound the number of v-minimal sets such that $|C| \leq p$ and $G[C]$ is connected, $S \subseteq C$ and $F \cap C = \emptyset$. If $N(S) \setminus (F \cup \{v\})$ is empty, then there is only one choice for C, namely S, and $2^{2p-|S|-|F|-1} \geq 1$. Otherwise, consider a vertex $u \in N(S) \setminus (F \cup \{v\})$. By the induction hypothesis, the number of v-minimal sets such that $|C| \leq p$ and $G[C]$ is connected, $S \cup \{u\} \subseteq C$ and $F \cap C = \emptyset$ is at most $2^{2p-|S|-|F|-2}$. Similarly, the number of v-minimal sets such that $|C| \leq p$ and $G[C]$ is connected, $S \subseteq C$ and $(F \cup \{u\}) \cap C = \emptyset$ is at most $2^{2p-|S|-|F|-2}$. Since either $u \in C$ or $u \notin C$, the two cases cover all possibilities for C and hence there are at most $2 \cdot 2^{2p-|S|-|F|-2} = 2^{2p-|S|-|F|-1}$ possibilities for C.

For a fixed S and F, the above proof can be translated into a procedure which lists all v-minimal sets such that $|C| \leq p$ and $G[C]$ is connected, $S \subseteq C$ and $F \cap C = \emptyset$. We run the procedure for $S = \{u\}$ and $F = \emptyset$ for every possible choice of u. Hence, there are at most $4^p|V(G)|$ v-minimal sets C such that $|C| \leq p$ and $G[C]$ is connected, and the sets can be efficiently listed. This concludes the proof. □

Observation 16. Let C be a v-minimal set of G and $G[S]$ be a connected component of $G[C]$. Then S is a v-minimal set.

In particular, if S is not a v-minimal set, then it contains a v-minimal set $S' \subset S$ and it is easy to see that $d(\{v\} \cup (C \setminus S) \cup S') \leq d(\{v\} \cup C)$, contradicting the minimality of C. Observation 16 tells us that any v-minimal set is the union of connected v-minimal sets. This makes it possible to use Lemma 15. We are now ready to give an algorithm for (size, p, q)-Cluster, the easiest of the three clustering problems. Our algorithm is based on a combination of *color coding* [2] with a dynamic programming algorithm which uses the observations made in this section.

Proposition 17 ([16]). *For every n, k there is a family of functions $\mathcal{F}$ of size $O(e^k \cdot k^{O(\log k)} \cdot \log n)$ such that every function $f \in \mathcal{F}$ is a function from $\{1, \ldots, n\}$ to $\{1, \ldots, k\}$ and for every subset S of $\{1, \ldots, n\}$ there is a function $f \in \mathcal{F}$ that*

is bijective when restricted to S. Furthermore, given n and k, $\mathcal{F}$ can be computed in time $O(e^k \cdot k^{O(\log k)} \cdot \log n)$.

Lemma 18. (size, p, q)-CLUSTER *can be solved in time* $2^{O(p)}|V(G)|^{O(1)}$.

Proof. We are given as input a graph G together with a vertex v and integers p and q. The task is to find a vertex set C of size at most $p-1$ such that $d(\{v\} \cup C) \leq q$. It is sufficient to search for a v-minimal set C satisfying these properties. By Observation 16, C can be decomposed into $C = S_1 \cup S_2 \ldots \cup S_t$ such that S_i is a connected v-minimal set for every i, $S_i \cap S_j = \emptyset$ for every $i \neq j$ and no edge of G has one endpoint in S_i and the other in S_j for every $i \neq j$. The algorithm of Lemma 15 can be used to list all connected v-minimal sets $S_1 \ldots S_n$; we have $n \leq 4^p|V(G)|$. For a subset Z of $\{1, \ldots, n\}$, define $C(Z) = \{v\} \cup \bigcup_{i \in Z} S_i$. Let $Z \subseteq \{1, \ldots, n\}$ be such that for every $i, j \in Z$ with $i \neq j$, we have $S_i \cap S_j = \emptyset$. We have that $|C(Z)| = 1 + \sum_{i \in Z} |S_i|$ and

$$d\big(C(Z)\big) \leq d(v) + \sum_{i \in Z} (\overline{d}(S_i, v) - d(S_i, v)).$$

If there is no edge with one endpoint in S_i and the other in S_j for some $i \neq j$, $i, j \in Z$, then the inequality above holds with equality. Our algorithm will select a Z such that $C = \bigcup_{i \in Z} S_i$. To ensure that the algorithm picks Z such that the sets S_i and S_j will be disjoint for every pair of distinct integers $i, j \in Z$ we will use color coding. In particular, we construct a family $\mathcal{F}$ of functions from $V(G) \setminus \{v\}$ to $\{1, \ldots, p-1\}$ as described in Proposition 17. The family $\mathcal{F}$ has size $O(e^p \cdot p^{O(\log p)} \cdot \log |V(G)|)$.

For each function $f \in \mathcal{F}$ we will think of the function as a coloring of $V(G) \setminus \{v\}$ with colors from $\{1, \ldots, p-1\}$. We will only look for a v-minimal set C whose vertices have different colors. This will not only ensure that any two sets S_i and S_j that we pick will be disjoint, it also automatically ensures that the size of the set C we return is at most $p-1$. If the input instance was a yes-instance then a solution set C exists, and the construction of $\mathcal{F}$ ensures that there will be a function $f \in \mathcal{F}$ which colors all vertices in C with different colors.

When considering a particular coloring f, we discard all sets from $S_1, \ldots S_n$ which have two vertices of the same color, so from this point, without loss of generality, all sets in $S_1, \ldots S_n$ have at most one vertex of each color. For a vertex set S, define $\mathsf{colors}(S)$ to be the set of colors occuring on vertices on G. For every $0 \leq i \leq n$, $0 \leq j \leq |E(G)|$ and $R \subseteq \{1, \ldots, p-1\}$, we define $T[i, j, S]$ to be true if there is a set $Z \subseteq \{1, \ldots, i\}$ such that all vertices of $C(Z)$ have distinct colors, $d(v) + \sum_{i \in Z} (\overline{d}(S_i, v) - d(S_i, v)) = j$ and $\mathsf{colors}(C(Z)) \subseteq R$. Clearly, there is a v-minimal set C such that $d(\{v\} \cup C) \leq q$ and all vertices of C have different color if and only if $T[n, j, \{1, \ldots, p-1\}]$ is true for some $j \leq q$. We can fill the table T using the following recurrence.

$$T[i, j, R] = \begin{cases} T[i-1, j, R] & \text{if } \mathsf{colors}(S_i) \setminus R \neq \emptyset \\ T[i-1, j, R] \vee T[i-1, & \\ j + d(S_i, v) - \overline{d}(S_i, v), R \setminus \mathsf{colors}(S_i)] & \text{otherwise} \end{cases} \quad (1)$$

Here we initialize $T[0, d(v), \emptyset]$ to true. The table has size $4^p|V(G)|^{O(1)} \cdot 2^p|V(G)|^{O(1)} = 8^p|V(G)|^{O(1)}$ and can be filled in time proportional to its size. Hence the total running time for the algorithm is $(8e)^{p+o(p)}|V(G)|^{O(1)}$. □

For (size, p, q)-Cluster the size of the set C we look for is already bounded by p. For $(\mathsf{nonedge}, p, q)$-Cluster and (nondeg, p, q)-Cluster, we cannot make this assumption, thus further arguments are needed to obtain Theorem 13.

4.1 Hardness Results

The algorithmic results in Section 3 still hold when parallel edges are allowed. Interestingly, the positive results in Section 4 do not: in particular, Observation 14 breaks done if there are parallel edges. The following hardness result shows that allowing parallel edges indeed make the problems more difficult:

Theorem 19 ($\star$). $(\mathsf{nonedge}, p, q)$-Partition *and* (nondeg, p, q)-Partition *are NP-complete for* $p = 0$ *on graphs with parallel edges.* (size, p, q)-Partition *is W[1]-hard parameterized by* p *on graphs with parallel edges.*

References

1. Ailon, N., Charikar, M., Newman, A.: Aggregating inconsistent information: ranking and clustering. In: STOC 2005, pp. 684–693 (2005)
2. Alon, N., Yuster, R., Zwick, U.: Color-coding. J. ACM 42(4), 844–856 (1995)
3. Bansal, N., Blum, A., Chawla, S.: Correlation clustering. Machine Learning 56(1-3), 89–113 (2004)
4. Bellman, R.: Dynamic programming treatment of the travelling salesman problem. J. ACM 9(1), 61–63 (1962)
5. Chen, J., Liu, Y., Lu, S.: An improved parameterized algorithm for the minimum node multiway cut problem. In: Dehne, F., Sack, J.-R., Zeh, N. (eds.) WADS 2007. LNCS, vol. 4619, pp. 495–506. Springer, Heidelberg (2007)
6. Chen, J., Liu, Y., Lu, S., O'Sullivan, B., Razgon, I.: A fixed-parameter algorithm for the directed feedback vertex set problem. J. ACM 55(5) (2008)
7. Cormen, T., Leiserson, C., Rivest, R., Stein, C.: Introduction to algorithms (2001)
8. Downey, R.G., Fellows, M.R.: Parameterized Complexity. Springer, Heidelberg (1999)
9. Heggernes, P., Lokshtanov, D., Nederlof, J., Paul, C., Telle, J.A.: Generalized graph clustering: Recognizing (o,q)-cluster graphs. In: Thilikos, D.M. (ed.) WG 2010. LNCS, vol. 6410, pp. 171–183. Springer, Heidelberg (2010)
10. Langston, M.A., Plaut, B.C.: On algorithmic applications of the immersion order: An overview of ongoing work presented at the third slovenian international conference on graph theory. Discrete Mathematics 182(1-3), 191–196 (1998)
11. Lokshtanov, D., Marx, D.: Clustering with local restrictions. In: preparation, http://www.ii.uib.no/~daniello/papers/clusteringLocal.pdf
12. Marx, D.: Parameterized graph separation problems. Theoret. Comput. Sci. 351(3), 394–406 (2006)
13. Marx, D., Razgon, I.: Fixed-parameter tractability of multicut parameterized by the size of the cutset. To appear in STOC (2011)

14. Mathieu, C., Sankur, O., Schudy, W.: Online correlation clustering. In: STACS, pp. 573–584 (2010)
15. Mathieu, C., Schudy, W.: Correlation clustering with noisy input. In: SODA, pp. 712–728 (2010)
16. Naor, M., Schulman, L.J., Srinivasan, A.: Splitters and near-optimal derandomization. In: FOCS, pp. 182–191 (1995)
17. Razgon, I., O'Sullivan, B.: Almost 2-sat is fixed-parameter tractable (extended abstract). In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part I. LNCS, vol. 5125, pp. 551–562. Springer, Heidelberg (2008)

Author Index