

Jianer Chen
S. Barry Cooper (Eds.)

LNCS 5532

Theory and Applications of Models of Computation

6th Annual Conference, TAMC 2009
Changsha, China, May 2009
Proceedings

 Springer

Commenced Publication in 1973

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison

Lancaster University, UK

Takeo Kanade

Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler

University of Surrey, Guildford, UK

Jon M. Kleinberg

Cornell University, Ithaca, NY, USA

Alfred Kobsa

University of California, Irvine, CA, USA

Friedemann Mattern

ETH Zurich, Switzerland

John C. Mitchell

Stanford University, CA, USA

Moni Naor

Weizmann Institute of Science, Rehovot, Israel

Oscar Nierstrasz

University of Bern, Switzerland

C. Pandu Rangan

Indian Institute of Technology, Madras, India

Bernhard Steffen

University of Dortmund, Germany

Madhu Sudan

Massachusetts Institute of Technology, MA, USA

Demetri Terzopoulos

University of California, Los Angeles, CA, USA

Doug Tygar

University of California, Berkeley, CA, USA

Gerhard Weikum

Max-Planck Institute of Computer Science, Saarbruecken, Germany

Jianer Chen S. Barry Cooper (Eds.)

Theory and Applications of Models of Computation

6th Annual Conference, TAMC 2009
Changsha, China, May 18-22, 2009
Proceedings

Volume Editors

Jianer Chen
Department of Computer Science and Engineering
Texas A&M University
Texas, USA
E-mail: chen@cs.tamu.edu

S. Barry Cooper
School of Mathematics
University of Leeds
Leeds, U.K.
E-mail: pmt6sbc@maths.leeds.ac.uk

Library of Congress Control Number: Applied for

CR Subject Classification (1998): F.2, F.3, F.4, G.2.2, H.1.1, G.4, I.1

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

ISSN 0302-9743
ISBN-10 3-642-02016-X Springer Berlin Heidelberg New York
ISBN-13 978-3-642-02016-2 Springer Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

springer.com

© Springer-Verlag Berlin Heidelberg 2009
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12679799 06/3180 5 4 3 2 1 0

Preface

Theory and Applications of Models of Computation (TAMC) is an international conference series with an interdisciplinary character, bringing together researchers working in computer science, mathematics (especially logic) and the physical sciences. This crossdisciplinary character, together with its focus on algorithms, complexity and computability theory, gives the conference a special flavor and distinction.

TAMC 2009 was the sixth conference in the series. The previous five meetings were held during May 17–19, 2004 in Beijing, May 17–20, 2005 in Kunming, May 15–20, 2006 in Beijing, May 22–25, 2007 in Shanghai, and April 25–29, 2008 in Xi'an. TAMC 2009 was held in ChangSha, during May 18–22, 2009. Next year will see a new departure, namely, the first TAMC conference to be held outside of Asia. TAMC 2010 will be held in Prague, capital of the Czech Republic.

At TAMC 2009 we had three plenary speakers, Leslie Valiant (Harvard University, USA), Moshe Vardi (Rice University, USA) and Matthew Hennessy (Trinity College, Ireland), giving one-hour talks each. Professor Valiant spoke on “Neural Computations That Support Long Mixed Sequence of Knowledge Acquisition Tasks,” Professor Vardi on “Constraints, Graphs, Algebra, Logic, and Complexity,” and Professor Hennessy on “Distributed Systems and Their Environments.” Their respective abstracts accompanying the talks are included in these proceedings.

In addition, there were two special sessions organized by S. Barry Cooper on “Models of Computation” and by Iyad A. Kanj on “Algorithms and Complexity.” The invited speakers in the first session were Dan Browne (Imperial College, London, UK), Alessandra Carbone (University Pierre et Marie Curie, France), Barry Cooper (University of Leeds, UK) and Andrea Sorbi (University of Siena, Italy). Invited speakers in the second session were Jiong Guo (Friedrich-Schiller-Universität Jena, Germany), Iyad Kanj (DePaul University, USA), Henning Fernau (University of Trier, Germany), and Binhai Zhu (Montana State University, USA). The Respective papers accompanying seven of the invited talks are included in these proceedings.

The TAMC conference series arose naturally in response to important scientific developments affecting how we compute in the twenty-first century. At the same time, TAMC is already playing an important regional and international role, and promises to become a key contributor to the scientific resurgence seen throughout China and other parts of Asia.

The TAMC 2009 Program Committee selected 39 papers from 86 submissions for presentation at the conference and inclusion in this LNCS volume.

We are very grateful to the Program Committee, and the many outside referees they called on, for the hard work and expertise which they brought to the difficult selection process. We also wish to thank all authors who submitted

their work for our consideration. The submissions for TAMC 2009 were of a particularly high standard, and inevitably many good-quality papers had to be excluded.

Finally, we would like to thank the members of the Editorial Board of *Lecture Notes in Computer Science* and the editors at Springer for their encouragement and cooperation throughout the preparation of this conference.

Of course TAMC 2009 would not have been possible without the support of our sponsors, Central South University, China, and the National Science Foundation of China, and we therefore gratefully acknowledge their help in the realization of this conference.

ChangSha 2009

Jianer Chen
S. Barry Cooper

Organization

Program Committee

Marat Arslanov	Kazan State University, Russia
Giorgio Ausiello	University of Rome, Italy
Hans Bodlaender	University of Utrecht, The Netherlands
Liming Cai	University of Georgia, USA
Cristian S. Calude	University of Auckland, New Zealand
Alessandra Carbone	University Pierre et Marie Curie, France
Jianer Chen, PC Co-chair	Central South University, China, and Texas A&M University, USA
Xi Chen	Princeton University, USA
Bob Coecke	Oxford University, UK
S. Barry Cooper, PC Co-chair	University of Leeds, UK
Vincent Danos	University of Edinburgh, UK
Anuj Dawar	Cambridge University, UK
Frank Dehne	Carleton University, Canada
Xiaotie Deng	City University of Hong Kong, China
Rod Downey	Victoria University, New Zealand
Mike Fellows	University of Newcastle, Australia
Fedor Fomin	University of Bergen, Norway
Lane A. Hemaspaandra	University of Rochester, USA
Kazuo Iwama	Kyoto University, Japan
Iyad Kanj	DePaul University, USA
Mike Langston	University of Tennessee, USA
Angsheng Li	The Institute of Software, Chinese Academy of Sciences, China
Ming Li	University of Waterloo, Canada
Wei Li	Beihang University, China
Giuseppe Longo	Ecole Normale Supérieure, France
Johann Makowsky	Technion, Israel
Luay Nakhleh	Rice University, USA
Luke Ong	Oxford University, UK
Venkatesh Raman	The Institute of Mathematical Sciences, India
Kenneth Regan	University at Buffalo - SUNY, USA
Rudiger Reischuk	University of Lubeck, Germany
Miklos Santha	CNRS, University Paris-Sud, France
Ivan Soskov	Sofia University, Bulgaria
Peter van Emde Boas	University of Amsterdam, The Netherlands

VIII Organization

Jianxin Wang	Central South University, China
Osamu Watanabe	Tokyo Institute of Technology, Japan
Ke Xu	Beihang University, China
Chee Yap	New York University, USA

Organizing Committee

Weihua Gui, Co-chair	Central South University, China
Shuquan Liang, Co-chair	Central South University, China
Jianer Chen	Texas A&M University, USA
S. Barry Cooper	University of Leeds, UK
Zhaohui Dai	Central South University, China
Angsheng Li	Chinese Academy of Sciences, China
Ming Liu	Central South University, China
Mingming Lu	Central South University, China
Yu Sheng	Central South University, China
Jianxin Wang	Central South University, China
Beiji Zou	Central South University, China

Sponsoring Institutions

South Central University, China
The National Natural Science Foundation of China

TAMC Steering Committee

Manindra Agrawal	IIT Kanpur, India
Jin-Yi Cai	University of Wisconsin-Madison, USA
S. Barry Cooper	University of Leeds, UK
Angsheng Li	Chinese Academy of Sciences, China

External Reviewers

Amano, Kazuyuki	Canetti, Ran
Baramidze, Gregory	Carrault, Guy
Batyrrshin, Ilnur	Che, Dongsheng
Bentz, Cédric	Chen, Jing
Bezakova, Ivona	Corruble, Vincent
Birov, Dimiter	Creignou, Nadia
Blakey, Ed	Cui, Peng
Bonizzoni, Paola	Dai, Decheng
Bose, Prosenjit	Demetrescu, Camil
Bouyer, Patricia	Desharnais, Josee
Bulatov, Andrei	Durr, Christophe

Eblen, John
Fragoudakis, Christodoulos
Franceschini, Gianni
Gasarch, William
Giannopoulos, Panos
Hazay, Carmit
Hoogeveen, Han
Hueffner, Falk
Itoh, Toshiya
Jay, Jeremy
Kalimullin, Iskander
Kari, Jarkko
Keliher, Liam
Kerenidis, Iordanis
Klaudel, Hanna
Kosub, Sven
Kullmann, Oliver
Lenisa, Marina
Liu, Chunmei
Lozin, Vadim
Maddy, Penny
Mahajan, Meena
Manyem, Prabhu
Mathelier, Anthony
Meister, Daniel
Moser, Hannes
Moss, Larry
Naswa, Sudhir
Ng, Keng Meng (Selwyn)

Okamoto, Yoshio
Ouaknine, Joel
Phillips, Charles
Regev, Oded
Richerby, David
Robertson, Joseph
Rogers, Gary
Rosen, Adi
Rotics, Udi
Sadrzadeh, Mehrnoosh
Segev, Danny
Shareghi, Pooya
Sikdar, Somnath
Slissenko, Anatoly
Soskova, Mariya
Speidel, Ulrich
Stefanescu, Gheorghe
Stephan, Frank
Tanaka, Keisuke
Tiomkin, Michael
Tsianos, Konstantinos
Vigliotti, Maria Grazia
Villanger, Yngve
Wang, Yingfeng
Wong, Duncan
Worrell, James
Yamamoto, Masaki
Zheng, Ying
Zimand, Marius

Table of Contents

Plenary Talks

Neural Computations That Support Long Mixed Sequences of Knowledge Acquisition Tasks	1
<i>Leslie G. Valiant</i>	
Constraints, Graphs, Algebra, Logic, and Complexity	3
<i>Moshe Y. Vardi</i>	
Distributed Systems and Their Environments	4
<i>Matthew Hennessy</i>	

Invited Special Session: Models of Computation

Co-evolution and Information Signals in Biological Sequences	6
<i>Alessandra Carbone and Linda Dib</i>	
The Extended Turing Model as Contextual Tool	18
<i>S. Barry Cooper</i>	
Strong Positive Reducibilities	29
<i>Andrea Sorbi</i>	

Invited Special Session: Algorithms and Complexity

Fixed-Parameter Algorithms for Graph-Modeled Date Clustering	39
<i>Jiong Guo</i>	
On Spanners of Geometric Graphs	49
<i>Iyad A. Kanj</i>	
Searching Trees: An Essay	59
<i>Henning Fernau and Daniel Raible</i>	
Approximability and Fixed-Parameter Tractability for the Exemplar Genomic Distance Problems	71
<i>Binhai Zhu</i>	

Contributed Papers

A Quadratic Kernel for 3-Set Packing	81
<i>Faisal N. Abu-Khzam</i>	

Quantitative Aspects of Speed-Up and Gap Phenomena.....	88
<i>Klaus Ambos-Spies and Thorsten Kräling</i>	
Computing the Exact Distribution Function of the Stochastic Longest Path Length in a DAG	98
<i>Ei Ando, Hirotaka Ono, Kunihiko Sadakane, and Masafumi Yamashita</i>	
On the Connection between Interval Size Functions and Path Counting	108
<i>Evangelos Bampas, Andreas-Nikolas Göbel, Aris Pagourtzis, and Aris Tentes</i>	
On the Red/Blue Spanning Tree Problem	118
<i>Sergey Bereg, Minghui Jiang, Boting Yang, and Binhai Zhu</i>	
Undecidability of Cost-Bounded Reachability in Priced Probabilistic Timed Automata	128
<i>Jasper Berendsen, Taolue Chen, and David N. Jansen</i>	
A Computational Proof of Complexity of Some Restricted Counting Problems	138
<i>Jin-Yi Cai, Pinyan Lu, and Mingji Xia</i>	
Block-Graph Width	150
<i>Maw-Shang Chang, Ling-Ju Hung, Ton Kloks, and Sheng-Lung Peng</i>	
Minimum Vertex Ranking Spanning Tree Problem on Permutation Graphs	158
<i>Ruei-Yuan Chang, Guanling Lee, and Sheng-Lung Peng</i>	
On Parameterized Exponential Time Complexity	168
<i>Jianer Chen, Iyad A. Kanj, and Ge Xia</i>	
Best-Order Streaming Model	178
<i>Atish Das Sarma, Richard J. Lipton, and Danupon Nanongkai</i>	
Behavioral and Logical Equivalence of Stochastic Kripke Models in General Measurable Spaces.....	192
<i>Ernst-Erich Doberkat</i>	
Influence of Tree Topology Restrictions on the Complexity of Haplotyping with Missing Data	201
<i>Michael Elberfeld, Ilka Schnoor, and Till Tantau</i>	
Improved Deterministic Algorithms for Weighted Matching and Packing Problems	211
<i>Qilong Feng, Yang Liu, Songjian Lu, and Jianxin Wang</i>	

Parameterized Complexity of Coloring Problems: Treewidth versus Vertex Cover (Extended Abstract)	221
<i>Jiří Fiala, Petr A. Golovach, and Jan Kratochvíl</i>	
Discovering Almost Any Hidden Motif from Multiple Sequences in Polynomial Time with Low Sample Complexity and High Success Probability	231
<i>Bin Fu, Ming-Yang Kao, and Lusheng Wang</i>	
A Complete Characterisation of the Linear Clique-Width of Path Powers	241
<i>Pinar Heggernes, Daniel Meister, and Charis Papadopoulos</i>	
Preserving Privacy versus Data Retention	251
<i>Markus Hinkelmann and Andreas Jakoby</i>	
Kolmogorov Complexity and Combinatorial Methods in Communication Complexity	261
<i>Marc Kaplan and Sophie Laplante</i>	
An Almost Totally Universal Tile Set	271
<i>Grégory Lafitte and Michael Weiss</i>	
Linear Kernel for Planar Connected Dominating Set	281
<i>Daniel Lokshantov, Matthias Mnich, and Saket Saurabh</i>	
A Simple Greedy Algorithm for the k -Disjoint Flow Problem	291
<i>Maren Martens</i>	
Minimizing AND-EXOR Expressions for Multiple-Valued Two-Input Logic Functions (Extended Abstract)	301
<i>Takaaki Mizuki, Hitoshi Tsubata, and Takao Nishizeki</i>	
Exact and Experimental Algorithms for a Huffman-Based Error Detecting Code	311
<i>Paulo Eustáquio Duarte Pinto, Fábio Protti, and Jayme Luiz Szwarcfiter</i>	
Terminal Coalgebras for Measure-Polynomial Functors	325
<i>Christoph Schubert</i>	
High Minimal Pairs in the Enumeration Degrees	335
<i>Andrea Sorbi, Guohua Wu, and Yue Yang</i>	
Searching a Circular Corridor with Two Flashlights	345
<i>Bo Jiang and Xuehou Tan</i>	
On the Complexity of the Multiple Stack TSP, k STSP	360
<i>Sophie Toulouse and Roberto Wolfier Calvo</i>	

Linear Programming Based Approximation Algorithms for Feedback Set Problems in Bipartite Tournaments	370
<i>Anke van Zuylen</i>	
An Online Algorithm for Applying Reinforcement Learning to Handle Ambiguity in Spoken Dialogues	380
<i>Fangju Wang and Kyle Swegles</i>	
A Fixed-Parameter Enumeration Algorithm for the Weighted FVS Problem	390
<i>Jianxin Wang and Guohong Jiang</i>	
On the Tractability of Maximal Strip Recovery	400
<i>Lusheng Wang and Binhai Zhu</i>	
Greedy Local Search and Vertex Cover in Sparse Random Graphs (Extended Abstract)	410
<i>Carsten Witt</i>	
Embedding the Diamond Lattice in the c.e. tt -Degrees with Superhigh Atoms	420
<i>Douglas Cenzer, Johanna N.Y. Franklin, Jiang Liu, and Guohua Wu</i>	
Feasibility of Motion Planning on Directed Graphs	430
<i>Zhilin Wu and Stéphane Grumbach</i>	
Polynomial-Time Algorithm for Sorting by Generalized Translocations	440
<i>Xiao Yin and Daming Zhu</i>	
The Two-Guard Polygon Walk Problem (Extended Abstract)	450
<i>John Z. Zhang</i>	
Approximation and Hardness Results for Label Cut and Related Problems	460
<i>Peng Zhang, Jin-Yi Cai, Linqing Tang, and Wenbo Zhao</i>	
An Observation on Non-Malleable Witness-Indistinguishability and Non-Malleable Zero-Knowledge	470
<i>Zongyang Zhang, Zhenfu Cao, and Rong Ma</i>	
Author Index	481

Neural Computations That Support Long Mixed Sequences of Knowledge Acquisition Tasks

Leslie G. Valiant*

School of Engineering and Applied Sciences
Harvard University
valiant@seas.harvard.edu

In this talk we shall first give a brief review of a quantitative approach to understanding neural computation [4-6]. We target so-called *random access* tasks, defined as those in which one instance of a task execution may need to access arbitrary combinations of items in memory. Such tasks are communication intensive, and therefore the known severe constraints on connectivity in the brain can inform their analysis.

Crucial to any theory of neural computation is the set of basic *task types* that are to be realized. We consider a set of task types that enable hierarchical structures to be built and a rich set of operations executed on them. The set consists of the four operations: *hierarchical memory formation*, *association*, *supervised memorization of conjunctions*, and *inductive learning of certain threshold functions*. Our choice of task types is different from the classical so-called associative memories, where a set of fixed strings of symbols are to be stored and accessed in a flat dictionary [2].

Also crucial to any analysis is the *model of computation* considered. Ours is the *neuroidal* model, which is designed to *underestimate* the capabilities of neurons. It consists of a network of nodes each realizing a threshold element but also having some additional states that allow for some programmability. The model has numerical *parameters* that aim to capture fundamental quantitative constraints on biological neurons. It has three such parameters, n the number of neurons, d the number of connections from or to each neuron, and k the inverse of the maximum strength of a synapse in comparison with the total strength needed to cause a threshold firing. We assume that for each real-world *item* there corresponds a set of neurons that are active when that item is being computed on. We therefore have a fourth parameter r for the number of neurons that are used typically to represent such an item.

Our algorithms for realizing these tasks are entirely distributed. The algorithmic style is called *vicinal*, as neighborly communication realizes the needed communication in a particularly simple way. In general, depending on the task types to be implemented, the network has to be able to realize certain combinatorial properties. We call these *vicinal graph properties* for the similar reason that they make possible direct interactions, via immediate neighborly connections, among the sets of neurons that represent concepts. The following are examples of these properties and refer to n node directed graphs where every node has

* This work was supported in part by NSF-CCF-04-27129.

indegree and outdegree about d : For the hierarchical memory formation task we need the property that for every two sets A and B of r nodes each, there are about r other nodes u in the graph that have the property that there are edges to u from at least k nodes in $A \cup B$. For association to be done without intermediate nodes, we need that for every two sets A and B of r nodes each, for every node u in B there are edges directed to u from at least k elements of A . As for other previously studied combinatorial properties, such as expansion [1], for the properties needed here random graphs are sufficient, while in principle those produced by deterministic or pseudorandom processes may also suffice.

The question of whether a random network with certain parameters has certain combinatorial properties can be often determined by analysis. However, to gain an understanding of the cumulative effect of the updates made by sequences of task instances executed on such a network, we resort to computer simulations.

We shall describe some recent computer simulation results, obtained jointly with Vitaly Feldman [3], that show that, for biologically realistic ranges of our four parameters n , d , k , and r , our model has substantial capacity for realizing large numbers of instances of all our four task types together. These simulations demonstrate that sequences of thousands of task instances, that intermingle the first three types (association, supervised memorization of conjunctions, and inductive learning of certain threshold functions) on items allocated by the first type (hierarchical memory formation), can be executed without the later task instances substantially degrading the functionality of the earlier ones. These results were obtained for two distinct general regimes that had been found through earlier analysis to be effective for realizing this set of tasks. The first, *regime alpha* [5], needs only the simplest algorithms and the weaker synapses, but apparently has the smaller capacity. The second, *regime beta* [4], uses slightly more complex algorithms and strong synapses, but can achieve extremely high capacities.

We conjecture that because of its simplicity, regime alpha is pervasive in neural computation and underlies the distributed representations often observed experimentally. For human performance over a lifetime, capacities of the order of several hundreds of thousands or a few million need to be explained. Our simulations show that regime beta offers an explanation, the first to our knowledge, of how this remarkable feat can be accomplished at all with realistic parameters.

References

- [1] Hoory, S., Linial, N., Wigderson, A.: Expander graphs and their applications. Bull. Amer. Math. Soc. 43, 439–561 (2006)
- [2] Graham, B., Willshaw, D.: Capacity and information efficiency of the associative net. Network: Comput. Neural Syst. 8, 35–54 (1997)
- [3] Feldman, V., Valiant, L.G.: Experience-induced neural circuits that achieve high capacity. Neural Computation (to appear, 2009)
- [4] Valiant, L.G.: Circuits of the Mind. Oxford University Press, Oxford (1994, 2000)
- [5] Valiant, L.G.: Memorization and association on a realistic neural model. Neural Computation 17(3), 527–555 (2005)
- [6] Valiant, L.G.: A quantitative theory of neural computation. Biological Cybernetics 95(3), 205–211 (2006)

Constraints, Graphs, Algebra, Logic, and Complexity*

Moshe Y. Vardi

Department of Computer Science
Rice University
Houston, TX 77251-1892, USA
vardi@cs.rice.edu
<http://www.cs.rice.edu/~vardi>

Abstract. A large class of problems in AI and other areas of computer science can be viewed as constraint-satisfaction problems. This includes problems in database query optimization, machine vision, belief maintenance, scheduling, temporal reasoning, type reconstruction, graph theory, and satisfiability. All of these problems can be recast as questions regarding the existence of homomorphisms between two directed graphs. It is well-known that the constraint-satisfaction problem is NP-complete. This motivated an extensive research program into identify tractable cases of constraint satisfaction.

This research proceeds along two major lines. The first line of research focuses on non-uniform constraint satisfaction, where the target graph is fixed. The goal is to identify those target graphs that give rise to a tractable constraint-satisfaction problem. The second line of research focuses on identifying large classes of source graphs for which constraint-satisfaction is tractable. We show in how tools from graph theory, universal algebra, logic, and complexity theory, shed light on the tractability of constraint satisfaction.

Reference

- [1] Kolaitis, P.G., Vardi, M.Y.: A logical approach to constraint satisfaction. In: Complexity of Constraints. LNCS, vol. 5250, pp. 125–155. Springer, Heidelberg (2008)

* Work supported in part by NSF grants CCR-0311326, CCF-0613889, ANI-0216467, and CCF-0728882.

Distributed Systems and Their Environments

Matthew Hennessy*

Trinity College Dublin, Ireland

Abstract. Process description languages, such as the picalculus [SW01], offer the possibility of formally describing system behaviour at varying levels of abstraction, and applying logical techniques to verify this behaviour.

But system behaviour often depends on environmental considerations. What a system can do depends on the current context in which it finds itself. It is this context which determines what information is available to the system, and therefore affects its future evolution. In a dual manner the current context determines the knowledge of the system which is available to its environment, and thus affects the use which can be made of the system. Moreover this interplay between a system and its environment is dynamic, changing as either or both evolve.

In this talk I will offer a survey of recent work on behavioural theories of systems in which their environments play a crucial role. We will see three instances in which environmental knowledge involves key features of *distributed* systems.

- *Access control*: Capabilities on resources and access rights to sites are determined by a static type system, [HRY05]; only partial knowledge of these types are available to the environment.
- *Network failure*: Systems run on a dynamically changing network of inter-connected nodes, where both the nodes and the connections are subject to failure, [FH08]; this network is shared between the system and its environment.
- *Resource cost*: Use of resources entail a cost, which must be borne by the processes responsible, [HG08]; the environment determines the overall funds available to processes for access to resources.

The focus will be on a particular process description language called Dpi [Hen07], oriented towards distributed systems.

References

- [FH08] Francalanza, A., Hennessy, M.: A theory of system behaviour in the presence of node and link failures. *Information and Computation* 206, 711–759 (2008)
- [Hen07] Hennessy, M.: *A distributed picalculus*. Cambridge University Press, Cambridge (2007)

* The financial support of Science Foundation Ireland is gratefully acknowledged.

- [HG08] Hennessy, M., Gaur, M.: Counting the cost in the picalculus (extended abstract). *Electr. Notes Theor. Comput. Sci.* (to appear) (2008); preliminary version presented at First Interaction and Concurrency Experience (ICE 2008), Reykjavik (July 2008)
- [HRY05] Hennessy, M., Rathke, J., Yoshida, N.: Safedpi: A language for controlling mobile code. *Acta Informatica* 42, 227–290 (2005)
- [SW01] Sangiorgi, D., Walker, D.: *The π -calculus: a Theory of Mobile Processes*. Cambridge University Press, Cambridge (2001)

Co-evolution and Information Signals in Biological Sequences

Alessandra Carbone and Linda Dib

¹ Département d'Informatique, Université Pierre et Marie Curie-Paris 6

² Génomique Analytique, FRE3214 CNRS-UPMC,

15, Rue de l'École de Médecine, 75005, Paris

Alessandra.Carbone@lip6.fr, Linda.Dib@gmail.com

Abstract. Information content of a pool of sequences has been defined in information theory through entropic measures aimed to capture the amount of variability within sequences. When dealing with biological sequences coding for proteins, a first approach is to align these sequences to estimate the probability of each amino-acid to occur within alignment positions and to combine these values through an "entropy" function whose minimum corresponds to the case where for each position, each amino-acid has the same probability to occur. This model is too restrictive when the purpose is to evaluate sequence constraints that have to be conserved to maintain the function of the proteins under random mutations. In fact, co-evolution of amino-acids appearing in pairs or tuplets of positions in sequences constitutes a fine signal of important structural, functional and mechanical information for protein families. It is clear that classical information theory should be revisited when applied to biological data. A large number of approaches to co-evolution of biological sequences have been developed in the last seven years. We present a few of them, discuss their limitations and some related questions, like the generation of random structures to validate predictions based on co-evolution, which appear crucial for new advances in structural bioinformatics.

1 Introduction

Protein sequences are chains of amino acids folding into a 3-dimensional structure and forming functionally active organic compounds in the cell. Protein sequences have evolved along billions of years and generated a number of *homologous* sequences, that is sequences with a same common ancestor, that are found today in genomes of living species. Ancestral sequences mutated through substitution, insertion or deletion of residues. It has been noticed that within a family of homologous protein sequences observed today, not all positions in the sequence have mutated with the same rate and that certain parts of the sequence (typically those playing a structural or functional role for the protein) are more conserved than others. Signals of conservation have been extracted from aligned homologous protein sequences (where the multiple sequence alignment, in short MSA, aims at piling up similar sub-sequences in the best way) from more than

forty years. In fact, these signals help to detect important functional or structural properties of proteins. Analysis have been based on the classical notion of information content, and captured numerically the residue variability in a single position of the MSA, by providing a global numerical score representing the entropy of the set of sequences through the combination of local information on alignment positions [25,14,2,41,47,32,33,48]. Extra information, such as physico-chemical properties of the residues and the local preservation of these properties along the MSA were also integrated in the analysis. Numerous interesting predictions of functional interaction sites on the surface of proteins have been obtained [26,27,3,37,23,4,15]. Yet, these methods demonstrated to work well for detecting interaction sites of a protein with other molecules (that is, other proteins, RNA, DNA, small molecules, etc) but not for detecting residues involved in allosteric conformations (these are structural changes affecting the activation or the inactivation of a protein function) or in certain structural features. This biological information is also coded in protein sequences.

The information content of a biological molecule corresponds to the number of sequence constraints that have to be conserved to maintain its function under random mutations [1,9]. If a protein sequence occurs in the genome of a species, it is likely to be also present in the genomes of phylogenetically close species and that its function is maintained. To study homologous sequences across species through their MSA, means to analyze residue conservation within single alignment positions, but also to analyze pairs, triplets or blocks of positions and check whether they have been mutating in concert or not. In biology, the phenomenon of “parallel” mutation is called *co-evolution* and experimental evidence supports the idea that this signal is related to structural changes in proteins.

Co-evolution is a measure to describe the impact of functional and structural constraints between residues on a pool of protein sequences. In the last ten years, many different methods have been proposed to study this signal in sequences. In this abstract we shall give an account of some of these computational approaches. The reason for doing this is that a convincing mathematical definition of co-evolution is missing, but several different methods to detect some form of co-evolution are proposed. These methods do not necessarily agree in their results and the computer science community might be interested to pick up the challenge of clearing out this goal by proposing a well founded framework where to decompose biological signals in sequences. Sequences hiding correlations between symbols other than positional conservation, like co-evolution of positions in biological sequences, demand to revisit the classical notion of entropy given in information theory. In [8], a first step has been made to adapt the classical notion of entropy and information content to protein sequences by taking into explicit account the distance tree describing similarity between sequences. By so doing, functional signals are shown to be more sharply detected.

On the methodological level, an original effort could be made in the unification of the approaches towards an appropriate definition of co-evolution and an understanding of the mathematical principles governing it. We shall rapidly discuss the fundamental problem of validating the methods analyzing co-evolution.

The bibliography is not exhaustive but, hopefully, representative of the work done in this field in these last years.

2 Basic Notions and Motivations

A biological sequence is a string of letters in a fixed alphabet of variable length. This can be a 20 letters alphabet (where a letter corresponds to an amino-acid, in short aa, also called *residue*) for proteins, or a 4 letters alphabet (where a letter corresponds to a nucleotide *A, T, C, G*) for DNA sequences. There is a map between the two alphabets that allows us to translate a DNA sequence into a protein sequence. This map is surjective but not injective. Protein sequences have an average length of 300-400aa, varying in an interval going from 30aa up to approximately 2000aa. These sequences, during millions of years of evolution transforming and creating new species, have been changing considerably. Today, within the species that we can observe, we find similar sequences, which diverged more or less radically. Assuming that we can reconstruct the common ancestor of a set of sequences, we are interested to study the changes within different positions in the sequences. These signals provide useful information on the evolution of the species. Sequences can evolve by three different operations: substitution, insertion and deletion of residues. *Substitution* means that a residue is replaced by another residue at a given position, *insertion* means that at a given position the insertion of a new residue is allowed, and *deletion* means that a given residue is lost from a sequence. Through these operations, sequences evolve by changing positional content and length. We call them *mutational changes*.

Today, given a sequence S , we can find in available databases, sequences S' which are homologous to S (that is, sequences S' which share a common ancestor with S , from which they both have evolved). The set of homologous sequences can be organized into a metric tree where leaves are labeled by sequences (each sequence labels exactly one leaf) and where the length of branches illustrates the



Fig. 1. Example of multiple sequence alignment. Point mutations appear as different letters (amino acids) in an alignment column, and insertions or deletions (gaps) appear as hyphens in the alignment sequences. Fully (partially) conserved positions are characterized by a single residue occurring everywhere (with high frequency) in a column, and co-evolved positions are characterized by several residues within columns such that a mutation in one column records a mutation in another column.

proximity or the divergence between sequences induced by mutational changes. These trees turn out to be very useful for studying mutational processes affecting several positions in a sequence [18]. Families of protein sequences display regions which are more conserved (less mutated) than others as well as sets of conserved positions, not necessarily close in sequence, which are close in 3-dimensions. These latter often correspond to residues involved in the interaction of the protein with other proteins, DNA, RNA, ligands, small molecules. The nature of these interactions might be multiple, spanning among binding specificity, allosteric regulation and conformational change of the protein. They are important for the function and for the mechanical properties of the protein.

With the large amount of undergoing sequencing, no more positions in sequences are fully conserved (see Fig. II) and it is interesting not only to provide a fine measure of conservation but also to give an appropriate one of co-evolution. It is believed that mutations in non-conserved positions can occur because they are either accompanied or preceded by compensatory changes in other variable positions. Such compensations would result in a coupling between changes in the two positions [19]. Clearly, not all mutated positions are under the pressure of co-evolution and the question is to find which ones are. In particular, several positions might be co-evolving together. Pairs of physically non contiguous sites that evolve in a similar way so to insure and maintain protein functionality and structure, are expected to belong to some network of co-evolving positions. This network is made of a set of residues that are physically connected to each other within the 3-dimensional structure of the protein. These residues might be surface residues or buried in the structure. It is the entire ensemble of residues in the network that will connect together non-contiguous co-evolved sites.

3 Approaches Detecting Residue Co-evolution

A number of different approaches to identify conserved and co-evolving positions have been developed. Below, we shall explain several methods attempting to define conservation and co-evolution. They use sequence information, at times phylogenetic information coming from trees of distances between sequences, and at times structural information coming from 3-dimensional structures known for a protein family. The methods are organized into two groups, a large one based on statistical approaches and another based on combinatorics. We consider only a few representative methods here to highlight a few important conceptual steps.

3.1 Statistical Approaches

Mutual Information Methods. Shannon's entropy H for column i in a MSA of protein sequences is a measure of the randomness of the residues in the column. It is calculated as $H(i) = -\sum_x p(x, i) \log_{20} p(x, i)$, where $p(x, i)$ is the frequency of occurrences of each residue x in column i . The resulting value varies from 0, in the case of complete conservation, to 1, which occur when all residues are equally distributed. The observed *joint entropy* of a pair of positions i, j , is calculated

similarly except that a pair of residues is used and that the sum extends over all possible combinations: $H(i, j) = -\sum_{x,y} p(x, i, y, j) \log_{20} p(x, i, y, j)$, where $p(x, i, y, j)$ is the probability of the pair of outcomes x, y at positions i, j respectively. The joint entropy values vary from 0 to 2. *Mutual information MI* measures the reduction of uncertainty about one position given information about the other. This can be thought of as the degree of correlation between two positions i and j in a MSA, $MI(i, j) = H(i) + H(j) - H(i, j)$. By definition, MI can vary between 1 and 0, with larger values reflecting a greater level of interdependence between positions.

In the context of MSAs, MI is an attractive metric because it explicitly measures the dependence of one position on another. Its usefulness has been limited by three factors though:

- positions with high variability or entropy will tend to have higher levels of both random and non-random MI than positions of lower entropy, even though the latter are more constraints and they might seem more likely to suffer of the neighboring positions [20,29].
- random MI arises because the alignments do not contain enough sequences for background noise to be negligible. Careful analysis show that a minimum amount of sequences should be guaranteed to ensure that random noise cannot compete with non-random signals [29].
- all position pairs present a (part of) MI which is due to the phylogenetic relationships of the organisms represented in the MSA. This source can be limited if very similar sequences are excluded from the MSA but it cannot be eliminated completely [29,42].

At the biological level, MI seems to be characterized by several signals, coming from structural-interaction, functional constraints, random noise and shared ancestry [49]. Thus the challenge is to separate the signal caused by structural and functional constraints from the others listed above. The effort should go into the removal of interference factors.

It has been observed that by using the correcting factor MI_r defined as $MI(a, b)/H(a, b)$, the influence of entropy can be partially removed [29]. In [13], the problems caused by phylogenetic closeness have been also partially solved. After removing the background noise due to phylogenetic and entropic signals, this method confirms that co-evolved positions are observed in sites proximal to regions with critical functions for the protein, where co-evolution occurs to maintain the structural characteristics around these regions and consequently to maintain the protein conformational and functional stability [21]. The estimation of MI related to the background signal is computationally cheap and accurate in contrast with other approaches based on bootstrapping for estimation of the background that are computationally very expensive [16,49]. A significant number of sequences in the protein family is still required though.

Another conceptual improvement for methods based on covariance detection of aligned sequences was highlighted in [16]. Here, the authors propose two types of co-evolved sites to be distinguished during the analysis: pairs of sites that

are spatially proximal, where compensatory mutations could maintain the local structural stability, and clusters of distant sites that are located in functional domains, suggesting a functional dependency between them. They observe that all sites detected under adaptive evolution in proteins belong to co-evolution groups, further underlining the importance of testing for co-evolution in selective constraints analysis.

Statistical Coupling Analysis method. The ability to efficiently propagate energy through tertiary structure is a fundamental property of many proteins and is the physical basis for key biological properties such as allostery and signal transmission. Pathways of co-evolved residues, also called coupled pathways, may represent conduits along which energy distributes through a protein structure to generate functional features. Based on this idea, a method detecting networks of co-evolved residues has been introduced by Ranganathan in [28,40].

The method starts from the idea that a protein family is represented by the associated MSA describing evolutionary constraints on the family. It demands the MSA to be evolutionarily well sampled, that is the MSA be sufficiently large and diverse where additional sequences do not significantly change the distribution of amino-acids in an alignment position. Under this assumption, two definitions guide the development of statistical parameters used by the method:

(i) (conservation) conservation at a given site in a MSA is defined as the overall deviance of amino acid frequencies at that site from their mean values. This means that if a site l contributes nothing to either the folding or function of the protein, the corresponding amino acid frequencies in the MSA should be unconstrained and, therefore, should approach their mean values in all proteins (in a database). However, if two (possibly more) sites i and j make some contribution, the amino acid distributions at these sites should deviate from these mean values, and the extent of this deviation should provide a quantitative measure of the underlying evolutionary constraint (that is, conservation).

(ii) (co-evolution) statistical coupling of two sites, i and j , is defined as the degree to which amino acid frequencies at site i change in response to a “perturbation” of frequencies at another site j . This definition of coupling does not require that the overall conservation of site i changes upon perturbation at j , but only that the amino acid population be rearranged. In other words, the functional coupling of two sites i and j should exert a mutual evolutionary constraint between these sites, which should be encoded in the statistical coupling of the underlying distributions of amino acids. That is, the distribution of residues at site j should depend on those at site i . It then follows that a lack of functional interaction between two sites i and k should, regardless of conservation at both sites, result in independence of their amino acid distributions.

Given a set S of homologous aligned sequences, to measure the degree of co-evolution, the method computes the distribution of amino acids at each position of S and determines the set of positions P , that are highly conserved without being fully conserved. For each position i of P , it induces a perturbation, that is it keeps only the subset S' of aligned sequences that contain the most conserved

amino acids at positions i , and looks at all the other positions of S' . It checks the new distribution of positions in S' at positions i , and deduces energetically connected positions with i based on the idea that connected positions are expected to share a similar distribution, that is to change similarly.

For example, extracting only the sequences that contain amino-acid a at position i results in a subalignment in which position i has experienced a substantial statistical perturbation (the fraction of a changes from 0.6 to 1.0 for instance). If the subalignment still retains sufficient size and diversity so that it remains a representative ensemble of the fold family, then the following properties should hold. First, sites l , which were not conserved in the parent alignment, should still show an amino acid distribution near the mean in all proteins. Second, sites k , which were conserved but not coupled to site i , should remain unchanged in their amino acid distribution. Finally, the coupling of sites i and j should induce a change in the observed distribution at site j upon perturbation at i . The magnitude of this change can be quantitatively measured as a statistical coupling energy between position j and the perturbation at i (with the notion of $\Delta\Delta G_{j,i}^{stat}$ defined in [28,40]). Several examples demonstrated the accuracy of the method to detect networks of functional and allosteric sites.

The Statistical Coupling Analysis method might miss some signals on the functional or mechanical properties of residues because of the hypothesis on the number of aligned sequences and of sequence divergence which are required to be both high by the statistical approach for a protein family. In general, these constraints limit the domain of applicability of the method to well-described families. By exploiting combinatorial information on the distance trees associated to a protein family, an alternative method (described below) has been proposed to drop divergence constraints [5].

3.2 Combinatorial Approaches

The Maximal SubTrees method. A sequence-based combinatorial alternative to statistical approaches has been proposed in [5] for the detection of functionally important co-evolved residue networks using phylogenetic information. This combinatorial approach is based on the analysis of a set of aligned sequences, on the associated distance tree and on the combinatorics of its subtrees and does not need structural data nor the knowledge of functional residues as the ATD method. The method is based on four main steps:

1. it selects conserved positions based on the scattering of residues (within the position) in the tree. The combinatorics of the tree plays a crucial role here in contrast with the analysis done on statistical approaches which are concerned on computing the entropy of aligned columns. For this step, a novel notion of rank for aligned positions in a MSA is used. It is defined to be the number of Maximal SubTrees (MST) observed at the position, where a MST is the largest subtree conserving a residue at the given position in all sequences labeling the nodes of the subtree. This notion is purely based on combinatorial information extracted from the distance tree.

2. it evaluates all pairs of selected conserved positions accordingly to the distribution of their residues in the tree. Namely, for each selected position, we parse the distance tree and apply numerical criteria to score co-evolution between pairs of residues conserved on subtrees and identify positions with similar residue distribution. We construct a matrix, called correspondence matrix, that collects scores between all pairs of positions in the MSA.
3. it identifies pairs of conserved and co-evolved positions during evolution, based on new numerical criteria adapted to tree and subtree analysis. Intuitively, the criteria compare and compute the difference between the correspondence matrix and the “ideal” matrix, observable in case of “perfect” co-evolution.
4. it clusters co-evolved residue networks reconstruction through an ad hoc designed clusterisation algorithm.

The method has been applied on the haemoglobin example studied in [40] and new co-evolved positions were found that have not been observed previously due to percentage identity constraints of Ranganathan method. It identifies pairs of coupled “residues” and not only pairs of positions. This property is of interest for determining meaningful functional signatures specific to the protein family.

Residual co-evolution detected by blocks. Compensation often involves residues that are proximal in the folded structure [19,36,46]. Based on this observation and on the analysis coming from [5] finding blocks of consecutive residues with high index of co-evolution, we decided to look at co-evolved blocks and check whether the analysis of their co-evolution can bring some information of functional origin. We propose a method to detect co-evolved blocks of residues, define an appropriate score of co-evolution between pairs of blocks, numerically rank them depending on their level of co-evolution, and clusterize them to obtain networks of co-evolved blocks (Dib and Carbone, manuscript, 2009). In contrast to Ranganathan method and to the other statistical methods proposed in the literature, where co-evolution of alignment columns is analyzed through a comparison of their residue distributions, our main focus is on groups of successive positions in the aligned homologous proteins, called *blocks*.

Given a MSA of n sequences, we consider groups of m consecutive positions in the MSA, where $m \geq 1$. For each group of consecutive positions and for each sequence in the MSA, there is a uniquely identified word that appears as a subword in the sequence and occupies the given consecutive positions. We look at the set of n words associated to a group of positions and study the combinatorial properties of the distribution of words in the group to establish whether a group is a block or not. Intuitively, we look at the space of all words of length m and check the variability of the words associated to a group. There are $n + 1$ different dimensions that are used to evaluate the space of words, and they correspond to the number of “errors” or “exceptions” that we want to accept: dimension 0 is the most restrictive one and it accepts no error, while dimension n accepts errors to occur in all sequences. For dimension i , we call a *block*, every maximal group of positions that contains at least two occurrences of each word, with the exception of at most i words (that might occur only once). Two blocks co-evolve when the associated distribution of words satisfy certain combinatorial thresholds (aimed

to ensure the mutual variability of words in the blocks) within a given dimension. For each dimension, we evaluate the score of co-evolution between blocks and predict, by a transitive rule, networks of co-evolved blocks. When applied to a large number of sequences, the algorithm might have a very costly computational time. A randomized version of the algorithm allows us to compute sets of co-evolved residues in high dimensions, and to handle more easily sets of divergent sequences typically displaying a high number of small blocks. The method has been applied to different protein families, among which, the haemoglobin family and a large scale analysis of the performance of the method is underway.

4 Random Generation of Sequences to Test the Method

Predictions should be tested against real biological data and at the moment experimental evidence on the role of networks of co-evolved residues is limited to less than ten protein families for which we know quite precisely, about functional interactions and allosteric properties. This experimental limitation demands for a theoretical validation of the predictions. A possible approach is to determine appropriate statistical tests based on random generation of sets of sequences. Co-evolved residue prediction could be made on randomly generated sets and compared, at a large scale, to predictions on real data. If predictions on real data could provide a number of residues (covering the 20-30% of the amino-acids in the protein as estimated on experimental data) that would appear in clusters within the 3-dimensional structure of the protein while the random generation could not, this would be an evidence of the significativity of the results.

Also, some of the signals detected by co-evolutionary methods might be artifacts of the phylogenetic tree [16], that is they might be signals coming from a niche of the large evolutionary tree of species. Niches might induce the detection of stronger signals of conservation (due to the higher similarity of the sequences) and inappropriate predictions on the maintaining of functional properties in protein families could be inferred. In order to make sure that positions in aligned sequences are co-evolving for functional reasons, again, we can model a random set of aligned sequences, sharing a number of suitable properties with the original ones, and test whether the same amount of important residues can be detected from both sets as an effect of the intrinsic properties of the sequences.

The generation of random sequences which seems appropriate needs to explicitly preserve:

- the percentage of gaps (corresponding to insertions or deletions in sequences, see Figure 1 for an example) in real sequences
- the percentage of sequence identity between sequences in the set
- the phylogenetic relationship between sequences in the set
- hydrophobic blocks in sequences, that is consecutive highly hydrophobic blocks
- amino-acid frequency usage
- the frequency of residue types (aromatic, aliphatic, neutral ...)

Attempts to generate sets of random protein sequences have been made in [38,39], where some of the conditions above have been taken into account. To

study co-evolutionary signals, a new and different approach to the random generation of sets of sequences is required. In fact, it is not conservation of residues on single positions that we need to capture but co-evolutionary relationships between several positions. A novel idea for doing this should be proposed and would be most welcome in structural bioinformatics. An approach to the testing of co-evolution would be to produce a random set of aligned sequences, respecting the phylogenetic tree, that would maintain the co-evolution signal in the protein family. Then the methodology could be tested to verify the robustness of it and the probability for a signal to be weakened by noise.

5 Conclusions

Conservation and mutual conservation, or co-evolution, might appear at first to be distinguished concepts but the combinatorial approach in [5] exploits the idea that along time evolution, conservation “comes before” co-evolution, in the sense that before two positions start to co-evolve together both they are conserved, and that conservation occupies a specific position within the continuum spectrum where to measure different degrees of co-evolution. A unifying theory explaining this intuition is missing.

Acknowledgments. We thank Anthony Mathelier for helpful comments.

References

1. Adami, C., Cerf, N.J.: Physical complexity of symbolic sequences. *Physica D* 137, 62–69 (2000)
2. Altschul, S.F., Madden, T.L., Schaffer, A.A., Zhang, J., Zhang, Z., Miller, W., Lipman, D.J.: Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Res.* 25, 3389–3402 (1997)
3. Armon, A., Graur, D., Ben-Tal, N.: ConSurf: An Algorithmic Tool for the Identification of Functional Regions in Proteins by Surface Mapping of Phylogenetic Information. *J. Mol. Biol.* 307, 447–463 (2001)
4. Cheng, G., Qian, B., Samudrala, R., Baker, D.: Improvement in protein functional site prediction by distinguishing structural and functional constraints on protein family evolution using computational design. *Nucleic Acids Res.* 33, 5861–5867 (2005)
5. Baussand, J., Carbone, A.: A combinatorial approach to detect co-evolved amino-acid networks in protein families with variable divergence (submitted manuscript) (2009)
6. Bickel, P.J., Kechris, K.J., Spector, P.C., Wedemayer, G.J., Glazer, A.N.: Finding important sites in protein sequences. *Proceedings of the National Academy of Sciences USA* 99, 14764–14771 (2002)
7. Capra, J.A., Singh, M.: Predicting functionally important residues from sequences conservation. *Bioinformatics* 23, 1875–1882 (2007)
8. Carbone, A., Engelen, S.: Information content of sets of biological sequences revisited. In: Condon, A., Harel, D., Kok, J.N., Salomaa, A., Winfree, E. (eds.) *Algorithmic Bioprocesses*. Natural Computing Series. Springer, Heidelberg (2008)
9. Carothers, J.M., Oestreich, S.C., Davis, J.H., Szostak, J.W.: Informational complexity and functional activity of RNA structures. *J. Am. Chem. Soc.* 126, 5130–5137 (2004)

10. Chang, M.S.S., Benner, S.A.: Empirical analysis of protein insertions and deletions determining parameters for the correct placement of gaps in protein sequence alignments. *J. Mol. Biol.* 341, 617–631 (2004)
11. Cheng, G., Qian, B., Samudrala, R., Baker, D.: Improvement in protein functional site prediction by distinguishing structural and functional constraints on protein family evolution using computational design. *Nucleic Acids Research* 33, 5861–5867 (2005)
12. del Alamo, M., Mateu, M.G.: Electrostatic repulsion, compensatory mutations, and long-range non-additive effects at the dimerization interface of the HIV capsid protein. *J. Mol. Biol.* 345, 893–906 (2005)
13. Dunn, S.D., Wahl, L.M., Gloor, G.B.: Mutual Information Without the Influence of Phylogeny or Entropy Dramatically Improves Residue Contact Prediction. *Bioinformatics* 24, 333–340 (2008)
14. Duret, L., Abdeddaim, S.: Multiple alignment for structural functional or phylogenetic analyses of homologous sequences. In: Higgins, D., Taylor, W. (eds.) *Bioinformatics sequence structure and databanks*. Oxford University Press, Oxford (2000)
15. Engelen, S., Trojan, L.A., Sacquin-Mora, S., Lavery, R., Carbone, A.: Joint Evolutionary Trees: detection and analysis of protein interfaces. *PLoS Computational Biology* 5(1), e1000267 (2009)
16. Fares, M.A., Travers, S.A.A.: A Novel Method for Detecting Intramolecular Coevolution: Adding a Further Dimension to Selective Constraints Analyses. *Genetics* 173, 9–23 (2006)
17. Fares, M.A., McNally, D.: CAPS: coevolution analysis using protein sequences. *Bioinformatics* 22, 2821–2822 (2006)
18. Felsenstein, J.: *Inferring Phylogenies*. Sinauer Associates, Sunderland (2004)
19. Fitch, W.M., Markowitz, E.: An improved method for determining codon variability in a gene and its application to the rate of fixation of mutations in evolution. *Biochem Genet.* 4, 579–593 (1970)
20. Fodor, A.A., Aldrich, R.W.: Influence of conservation on calculations of amino acid covariance in multiple sequence alignments. *Proteins* 56, 211–221 (2004a)
21. Gloor, G.B., Martin, L.C., Wahl, L.N., Dunn, S.D.: Mutual information in protein multiple sequence alignments reveals two classes of coevolving positions. *Biochemistry* 44, 7156–7165 (2005)
22. Halperin, I., Wolfson, H., Nussinov, R.: Correlated mutations: advances and limitations. A study on fusion proteins and on the Cohesin/Dockerin families. *Proteins* 63, 832–845 (2006)
23. Innis, C.A.: siteFiNDER-3D: a web-based tool for predicting the location of functional sites in proteins. *Nucleic Acids Res.* 35(Web-Server-Issue), 489–494 (2007)
24. Kass, I., Horovitz, A.: Mapping pathways of allosteric communication in GroEL by analysis of correlated mutations. *Proteins: Structure, Function, and Bioinformatics* 48, 611–617 (2002)
25. Lecompte, O., Thompson, J.D., Plewniak, F., Thierry, J., Poch, O.: Multiple alignment of complete sequences (MACS) in the post-genomic era. *Gene* 270, 17–30 (2001)
26. Lichtarge, O., Bourne, H.R., Cohen, F.E.: An evolutionary trace method defines binding surfaces common to protein families. *J. Mol. Biol.* 257, 342–358 (1996)
27. Lichtarge, O., Sowa, M.E.: Evolutionary predictions of binding surfaces and interactions. *Current Opinions in Structural Biology* 12, 21–27 (2002)
28. Lockless, S.W., Ranganathan, R.: Evolutionary conserved pathways of energetic connectivity in protein families. *Science* 286, 295–299 (1999)
29. Martin, L.C., Gloor, G.B., Dunn, S.D., Wahl, L.M.: Using information theory to search for co-evolving residues in proteins. *Bioinformatics* 21, 4116–4124 (2005)

30. Mateu, M.G., Fersht, A.R.: Mutually compensatory mutations during evolution of the tetramerization domain of tumor suppressor p53 lead to impaired hetero-oligomerization. *Proc. Natl. Acad. Sci. USA* 96, 3595–3599 (1999)
31. Mintseris, J., Weng, Z.: Structure, function, and evolution of transient and obligate proteinprotein interactions. *Proc. Natl. Acad. Sci. USA* 102, 10930–10935 (2005)
32. Notredame, C.: Recent progresses in multiple sequence alignment: a survey. *Pharmacogenomics* 31, 131–144 (2002)
33. Notredame, C.: Recent evolutions of multiple sequence alignment algorithms. *PLOS Computational Biology* 8, e123 (2007)
34. Pazos, F., Helmer-Citterich, M., Ausiello, G., Valencia, A.: Correlated mutations contain information about proteinprotein interaction. *J. Mol. Biol.* 271, 511–523 (1997)
35. Pazos, F., Valencia, A.: In silico two-hybrid system for the selection of physically interacting protein pairs. *Proteins* 47, 219–227 (2002)
36. Poon, A., Chao, L.: The rate of compensatory mutation in the DNA bacteriophage X174. *Genetics* 170, 989–999 (2005)
37. Pupko, T., Bell, R.E., Mayrose, I., Glaser, F., Ben-Tal, N.: Rate4Site: an algorithmic tool for the identification of functional regions in proteins by surface mapping of evolutionary determinants within their homologues. *Bioinformatics* 18, S71–S77 (2002)
38. Rambaut, A., Grassly, N.C.: Seq-Gen: An application for the Monte Carlo simulation of DNA sequence evolution along phylogenetic trees. *Comput. Appl. Biosci.* 13, 235–238 (1997)
39. Strophe, C.L., Scott, S.D., Moriyama, E.N.: indel-Seq-Gen: A new protein family simulator incorporating domains, motifs, and indels. *Mol. Biol. Evol.* 24, 640–649 (2007)
40. Suel, G.M., Lockless, S.W., Wall, M.A., Ranganathan, R.: Evolutionary conserved networks of residues mediate allosteric communication in proteins. *Nature Struct. Biol.* 23, 59–69 (2003)
41. Thompson, J.D., Plewniak, F., Poch, O.: A comprehensive comparison of multiple sequence alignment programs. *Nucleic Acids Research* 27, 12682–12690 (1999)
42. Tillier, E.R., Lui, T.W.: Using multiple interdependency to separate functional from phylogenetic correlations in protein alignments. *Bioinformatics* 19, 750–755 (2003)
43. Tress, M., de Juan, D., Grana, O., Gomez, M.J., Gomez-Puertas, P., Gonzalez, J.M., Lopez, G., Valencia, A.: Scoring docking models with evolutionary information. *Proteins* 60, 275–280 (2005)
44. Yang, Z.: Adaptive molecular evolution. In: Balding, D., Bishop, M., Cannings, C. (eds.) *Handbook of statistical genetics*, pp. 327–350. Wiley, New York (2001)
45. Yang, Z., Swanson, W.J., Vacquier, V.D.: Maximum likelihood analysis of molecular adaptation in abalone sperm lysin reveals variable selective pressures among lineages and sites. *Mol. Biol. Evol.* 17, 1446–1455 (2000)
46. Yanofsky, C., Horn, V., Thorpe, D.: Protein Structure Relationships Revealed by Mutational Analysis. *Science* 146, 1593–1594 (1964)
47. Wallace, I.M., Blackshields, G., Higgins, D.G.: Multiple sequence alignments. *Curr. Opin. Struct. Biol.* 15, 261–266 (2005)
48. Watson, J.D., Laskowski, R.A., Thornton, J.M.: Predicting protein function from sequence and structural data. *Curr. Opin. Struct. Biol.* 15, 275–284 (2005)
49. Wollenberg, K.R., Atchley, W.R.: Separation of phylogenetic and functional associations in biological sequences by using the parametric bootstrap. *Proc. Natl. Acad. Sci. U S A* 97, 3288–3291 (2000)

The Extended Turing Model as Contextual Tool

S. Barry Cooper*

School of Mathematics, University of Leeds, Leeds LS2 9JT, U.K.

pmt6sbc@leeds.ac.uk

<http://www.amsta.leeds.ac.uk/~pmt6sbc/>

Abstract. Computability concerns information with a causal – typically algorithmic – structure. As such, it provides a schematic analysis of many naturally occurring situations. We look at ways in which computability-theoretic structure emerges in natural contexts. We will look at how algorithmic structure does not just emerge mathematically from information, but how that emergent structure can model the emergence of very basic aspects of the real world.

1 Introduction

The adequacy of the classical Turing model of computation — as first presented in [19] — is in question in many contexts. There is widespread doubt concerning the reducibility to this model of a broad spectrum of real-world processes and natural phenomena, from basic quantum mechanics to aspects of evolutionary development, or human mental activity.

In 1939 Turing [20] described an extended model providing mathematical form to the algorithmic content of structures which are presented in terms of real numbers. Most scientific laws with a computational content can be framed in terms of appropriate Turing reductions. This can be seen in implicit form in Newton's *Principia* [15], published some 272 years before Turing's paper. Newton's work was formative in established a more intimate relationship between mathematics and science, and one which held the attention of Turing, in various guises, throughout his short life (see Hodges [11]).

Just as the history of arithmetically-based algorithms, underlying many human activities, eventually gave rise to models of computation such as the Turing machine, so the oracle Turing machine schematically addresses the scientific focus on the extraction of predictions governing the form of computable relations over the reals. Whereas the inputting of data presents only time problems for the first model, the second model is designed to deal with possibly incomputable inputs, or at least inputs for which we do not have available an algorithmic presentation. One might reasonably assume that data originating from observation of the real world carries with it some level of computability, but we are yet to agree a mathematical model of physical computation which dispenses with

* Preparation of this article supported by E.P.S.R.C. Research Grant No. EP/G000212.

the relativism of the oracle Turing machine. In fact, even as the derivation of recognisable incomputability in mathematics arises from quantification over algorithmic objects, so definability may play an essential role in fragmenting and structuring the computational content of the real world. The Turing model of computability over the natural numbers appears to many people to be a poor indicator of what to expect in science.

2 The Turing Landscape: From Local to Global

The oracle Turing machine, which made its first appearance in Turing [20], should be familiar enough. The details are not important, but can be found in most reasonable introductions to computability (see for instance [4]).

The basic form of the questioning permitted is modelled on that of everyday scientific practice. This is seen most clearly in today's digital data gathering, whereby one is limited to receiving data which can be expressed, and transmitted to others, as information essentially finite in form. But with the model comes the capacity to collate data in such a way as enable us to deal with arbitrarily close approximations to infinitary inputs and hence outputs, giving us an exact counterpart to the computing scientist working with real-world observations. If the different number inputs to the oracle machine result in 0-1 outputs from the corresponding Turing computations, one can collate the outputs to get a binary real computed from the oracle real, the latter now viewed as an input. This gives a partial computable functional Φ , say, from reals to reals.

As usual, one cannot computably know when the machine for Φ computes on a given natural number input, so Φ may not always give a fully defined real output. So Φ may be partial. One can computably list all oracle machines, and so index the infinite list of all such Φ , but one cannot computably sift out the partial Φ 's from the list.

Anyway, put \mathbb{R} together with this list, and we get the Turing Universe. Depending on one's viewpoint, this is either a rather reduced scientific universe, or a much expanded one. The familiar mathematical presentation of it is due to Emil Post, in his search for the informational underpinnings of computational structure.

Post's first step was to gather together binary reals which are computationally indistinguishable from each other, in the sense that they are mutually Turing computable from each other. Mathematically, this delivered a more standard mathematical structure to investigate — the familiar upper semi-lattice of the *degrees of unsolvability*, or *Turing degrees*. There is no simple scientific counterpart of the mathematical model, or any straightforward justification for what Post did with the Turing universe for perfectly good mathematical reasons — if one wants to get a material avatar of the Turing landscape one needs both a closer and a more comprehensive view of the physical context.

In approaching with this, we are presented with very real and inescapable causal structure, accompanied by the information content of its particular instantiations, and the problem is to explain and characterise this connection. The

difficulty is that recognition of these causal structures entails us taking a global view of an environment of which we ourselves are a component. When we look at the mysterious emergence of structure in nature, either subatomic laws, or the richness of life forms, or large-scale galactic or super-galactic structures, we are not just looking at information, but at expressions of patterns of a universal nature. And patterns the origins of which science is as yet unable to explain.

When we inspect the intricacies of the Cat's Eye Nebula, say, as revealed by the Hubble Space Telescope, we feel we should be able to explain the remarkable complexity observed on the basis of our understanding of the local physics. The intuition is that it should be possible to describe global relations in terms of local structure, so capturing the emergence of large-scale structure. The mathematics pertaining to any particular example will be framed in terms of the specific interactive structure on which it is based. But if one wants to reveal general characteristics, and approach deep problems around the emergence of physical laws and constants, which current theory fails to do, one needs something more fundamental.

Schematically, as we have argued, any causal context framed in terms everyday computable mathematics can be modelled in terms of Turing reductions. Then emergence can be formalised as definability over the appropriate substructure of the Turing universe; or more generally, as invariance under automorphisms of the Turing universe. Simple and fundamental as the notions of definability and invariance are, and basic as they are to everyday thought and discourse, as concepts they are not well understood outside of mathematics.

This is seen most strikingly in the physicists' apparent lack of awareness of the concept in interpreting the collapse of the wave function. Quantum decoherence and the many-worlds hypothesis comprise a far more outlandish interpretive option than does speculating that measurements, in enriching an environment, merely lead to an assertion of invariance. It appears a sign of desperation to protect consistent histories by inventing new universes, when the mathematics of our observable universes already contains a straightforward explanation. We argue that many scientific puzzles can be explained in terms of failures of invariance in different contexts, and that the key task is to identify useful theoretical models within which to investigate the nature of invariance more fully. One of the most relevant of these models has to be that of Turing, based as it is on a careful analysis of the characteristics of algorithmic computation.

This brings us to a well-known research programme, initiated by Hartley Rogers in his 1967 paper [17], in which he drew attention to the fundamental problem of characterising the Turing invariant relations. Again, the intuition is that these are key to pinning down how basic laws and entities emerge as mathematical constraints on causal structure. It is important to notice how the richness of Turing structure discovered so far becomes the raw material for a multitude of non-trivially definable relations, matching in its complexity what we attempt to model.

Unfortunately, the current state of Rogers' programme is not good. For a number of years research in this area was dominated by a proposal originating

with the Berkeley mathematician Leo Harrington, which can be (very) roughly stated:

Bi-interpretability Conjecture: *The Turing definable relations are exactly those with information content describable in second-order arithmetic.*

Most importantly, bi-interpretability is not consistent with the existence of non-trivial Turing automorphisms. Despite decades of work by a number of leaders in the field, the exact status of the conjecture is still a matter of controversy.

For those of us who have grown up with Thomas Kuhn's 1962 book [14] on the structure of scientific revolutions, such difficulties and disagreements are not seen as primarily professional failures, or triggers to collective shame (although they may be that too), but rather signs that something scientifically important is at stake.

3 Foundational Problems in Physics

A far more public controversy currently shapes developments around important issues affecting theoretical physics — see, for example the recent books of Lee Smolin [18] and Peter Woit [22].

As Peter Woit [22, p.1] describes, according to purely pragmatic criteria particle physics has produced a standard model which is remarkably successful, and has great predictive power:

By 1973, physicists had in place what was to become a fantastically successful theory of fundamental particles and their interactions, a theory that was soon to acquire the name of the standard model. Since that time, the overwhelming triumph of the standard model has been matched by a similarly overwhelming failure to find any way to make further progress on fundamental questions.

The reasons why people are dissatisfied echo misgivings going back to Einstein himself [9, p.63]:

... I would like to state a theorem which at present can not be based upon anything more than upon a faith in the simplicity, i.e. intelligibility, of nature: ... nature is so constituted that it is possible logically to lay down such strongly determined laws that within these laws only rationally completely determined constants occur (not constants, therefore, whose numerical value could be changed without destroying the theory) ...

If one really does have a satisfying description of how the universe is, it should not contain arbitrary elements with no plausible explanation. In particular, a theory containing arbitrary constants, which one adjusts to fit the intended interpretation of the theory, is not complete.

And as Woit observes [22]:

One way of thinking about what is unsatisfactory about the standard model is that it leaves seventeen non-trivial numbers still to be explained

...

At one time, it had been hoped that string theory would supply a sufficiently fundamental framework to provide a much more coherent and comprehensive description, in which such arbitrary ingredients were properly pinned down. But despite its mathematical attractions, there are growing misgivings about its claimed status as “the only game in town” as a unifying explanatory theory. Here is how one time string theorist Daniel Friedan [10] combatively puts it:

The longstanding crisis of string theory is its complete failure to explain or predict any large distance physics. ... String theory is incapable of determining the dimension, geometry, particle spectrum and coupling constants of macroscopic spacetime. ... The reliability of string theory cannot be evaluated, much less established. String theory has no credibility as a candidate theory of physics.

Smolin starts his book [18]:

From the beginning of physics, there have been those who imagined they would be the last generation to face the unknown. Physics has always seemed to its practitioners to be almost complete. This complacency is shattered only during revolutions, when honest people are forced to admit that they don't know the basics.

He goes on to list what he calls the “five great [unsolved] problems in theoretical physics”. Gathering these together, and slightly editing, they are [18, pp.5-16]:

1. Combine general relativity and quantum theory into a single theory that can claim to be the complete theory of nature.
2. Resolve the problems in the foundations of quantum mechanics.
3. The unification of particles and forces problem: Determine whether or not the various particles and forces can be unified in a theory that explains them all as manifestations of a single, fundamental entity.
4. Explain how the values of the free constants in the standard model of physics are chosen in nature.
5. Explain dark matter and dark energy. Or, if they don't exist, determine how and why gravity is modified on large scales.

Problems also occur with the picture that general relativity provides us with of the early stages of cosmology. This is Martin Bojowald writing [1, p.383]:

The beginning was extremely violent with conditions such as diverging energy densities and tidal forces under which no theory can prevail ...there are situations in the universe which ...can be described only

by solutions to general relativity which . . . must have a singularity in the past or future . . . From the observed expansion of our current universe one can conclude . . . that according to general relativity there must have been such a singularity in the past . . . but . . . it is clear that the singularity does not so much present a boundary to the universe as a boundary to the classical theory: The theory predicts conditions under which it has to break down and is thus incomplete. . . . A definitive conclusion about a possible beginning can therefore be reached only if a more complete theory is found which is able to describe these very early stages meaningfully.

That each of the above problems can be framed in terms of definability is not so surprising, since that is exactly how, essentially, they are approached by researchers. The question is the extent to which progress is impeded by a lack of consciousness of this fact, and an imperfect grip of what is fundamental. Quoting Einstein again (from a letter to Robert Thornton, dated 7 December 1944, Einstein Archive 61-754), this time on the relevance of a philosophical approach to physics:

So many people today – and even professional scientists – seem to me like someone has seen thousands of trees but has never seen a forest. A knowledge of the historical and philosophical background gives that kind of independence from prejudices of his generation from which most scientists are suffering. This independence created by philosophical insight is – in my opinion – the mark of distinction between a mere artisan or specialist and a real seeker after truth.

Smolin's comment [18, p.263] is in the same direction, though more specifically directed at the string theorists:

The style of the string theory community . . . is a continuation of the culture of elementary-particle theory. This has always been a more brash, aggressive, and competitive atmosphere, in which theorists vie to respond quickly to new developments . . . and are distrustful of philosophical issues. This style supplanted the more reflective, philosophical style that characterized Einstein and the inventors of quantum theory, and it triumphed as the center of science moved to America and the intellectual focus moved from the exploration of fundamental new theories to their application.

So what is it that is fundamental that is being missed? For Smolin [18, p.241], it is *causality*:

It is not only the case that the spacetime geometry determines what the causal relations are. This can be turned around: Causal relations can determine the spacetime geometry . . . Its easy to talk about space or spacetime emerging from something more fundamental, but those who have tried to develop the idea have found it difficult to realize in practice. . . . We now believe they failed because they ignored the role that

causality plays in spacetime. These days, many of us working on quantum gravity believe that *causality itself is fundamental* – and is thus meaningful even at a level where the notion of space has disappeared.

Citing Penrose as an early champion of the role of causality, Smolin also mentions Rafael Sorkin, Fay Dowker, and Fotini Markopoulou, known in this context for their interesting work on causal sets (see [2]), which abstract from causality relevant aspects of its underlying ordering relation. Essentially, causal sets are partial orderings which are locally finite, providing a model of spacetime with built-in discreteness. Despite the apparent simplicity of the mathematical model, it has had striking success in approximating the known characteristics of spacetime. An early prediction, in tune with observation, concerned the value of Einstein’s cosmological constant.

Of course, this preoccupation with causality might suggest to a logician a need to also look at its computational content. Smolin’s comment that “Causal relations can determine the spacetime geometry” touches on one of the biggest disappointments with string theory, which turns out to be a ‘background dependant’ theory with a vengeance — one has literally thousands of candidate Calabi-Yau spaces for shaping the extra dimensions of superstring theory. In current superstring models, Calabi-Yau manifolds are those qualifying as possible space formations for the six hidden spatial dimensions, their undetected status explained by the assumption of their being smaller than currently observable lengths.

Ideally, a truly fundamental mathematical model should be background independent, bringing with it a spacetime geometry arising from within.

4 Turing Invariance and the Laws of Physics

There are obvious parallels between the Turing universe and the material world. Each of which in isolation, to those working with specific complexities, may seem superficial and unduly schematic. But the lessons of the history of mathematics and its applications are that the simplest of abstractions can yield unexpectedly far-reaching and deep insights into the nature of the real world.

Most basic, science describes the world in terms of real numbers. This is not always immediately apparent, any more that the computer on ones desk is obviously an avatar of a universal Turing machine. Nevertheless, scientific theories consist, in their essentials, of postulated relations upon reals. These reals are abstractions, and do not come necessarily with any recognisable metric. They are used because they are the most advanced presentational device we can practically work with. There is no faith that reality itself consists of information presented in terms of reals. In fact, those of us who believe that mathematics is indivisible, no less in its relevance to the material world, have a due humility about the capacity for our science to capture more than a surface description of reality.

Some scientists would take us in the other direction, and claim that the universe is actually finite, or at least countably discrete. We have argued elsewhere

(see for example [8]) that to most of us a universe without algorithmic content is inconceivable. And that once one has swallowed that bitter pill, infinitary objects are not just a mathematical convenience (or inconvenience, depending on ones viewpoint), but become part of the mathematical mold on which the world depends for its shape. As it is, we well know how essential algorithmic content is to our understanding of the world. The universe comes with recipes for doing things. It is these recipes which generate the rich information content we observe, and it is reals which are the most capacious receptacles we can humanly carry our information in, and practically unpack.

Globally, there are still many questions concerning the extent to which one can extend the scientific perspective to a comprehensive presentation of the universe in terms of reals — the latter being just what we need to do in order to model the immanent emergence of constants and natural laws from an entire universe. Of course, there are many examples of presentations entailed by scientific models of particular aspects of the real world. But given the fragmentation of science, it is fairly clear that less natural presentations may well have an explanatory role, despite their lack of a role in practical computation.

The natural laws we observe are largely based on algorithmic relations between reals. Newtonian laws of motion will computably predict, under reasonable assumptions, the state of two particles moving under gravity over different moments in time. And, as previously noted, the character of the computation involved can be represented as a Turing functional over the reals representing different time-related two-particle states. One can point to physical transitions which are not obviously algorithmic, but these will usually be composite processes, in which the underlying physical principles are understood, but the mathematics of their workings outstrip available analytical techniques.

Over forty years ago, Georg Kreisel [12] distinguished between classical systems and *cooperative phenomena* not known to have Turing computable behaviour, and proposed [13, p.143, Note 2] a collision problem related to the 3-body problem, which might result in “an analog computation of a non-recursive function (by repeating collision experiments sufficiently often)”. However, there is a qualitatively different apparent breakdown in computability of natural laws at the quantum level — the *measurement problem* challenges us to explain how certain quantum mechanical probabilities are converted into a well-defined outcome following a measurement. In the absence of a plausible explanation, one is denied a computable prediction. The physical significance of the Turing model depends upon its capacity for explaining what is happening here. If the phenomenon is not composite, it does need to be related in a clear way to a Turing universe designed to model computable causal structure. We will need to talk more about definability and invariance.

For the moment, let us think in terms of what an analysis of the automorphisms of *any* sufficiently comprehensive, sufficiently fundamental, mathematical model of the material universe might deliver.

Let us first look at the relationship between automorphisms and many-worlds. When one says “I tossed a coin and it came down heads, maybe that means

there is a parallel universe where I tossed the coin and it came down tails”, one is actually predicating a large degree of correspondence between the two parallel universes. The assumption that *you* exist in the two universes puts a huge degree of constraint on the possible differences — but nevertheless, some relatively minor aspect of our universe has been rearranged in the parallel one. There are then different ways of relating this to the mathematical concept of an automorphism.

One could say that the two parallel worlds are actually isomorphic, but that the structure was not able to *define* the outcome of the coin toss. So it and its consequences appear differently in the two worlds. Or one could say that what has happened is that the worlds are *not* isomorphic, that actually we were able to change quite a lot, without the parallel universe looking very different, and that it was these fundamental but hidden differences which forces the worlds to be separate and not superimposed, quantum fashion. The second view is more consistent with the view of quantum ambiguity displaying a failure of definability. The suggestion here being that the observed existence of a particle (or cat!) in two different states at the same time merely exhibits an automorphism of our universe under which the classical level is rigid (just as the Turing universe displays rigidity above $0''$) but under which the sparseness of defining structure at the more basic quantum level enables the automorphism to re-represent our universe, with everything at our level intact, but with the particle in simultaneously different states down at the quantum level. And since our classical world has no need to decohere these different possibilities into parallel universes, we live in a world with the automorphic versions superimposed. But when we make an observation, we establish a link between the undefined state of the particle and the classical level of reality, which destroys the relevance of the automorphism.

To believe that we now get parallel universes in which the alternative states are preserved, one now needs to decide how much else one is going to change about our universe to enable the state of the particle destroyed as a possibility to survive in the parallel universe — and what weird and wonderful things one must accommodate in order to make that feasible. It is hard at this point to discard the benefits brought by a little mathematical sophistication. Quantum ambiguity as a failure of definability is a far more palatable alternative than the invention of new worlds of which we have no evidence or scientific understanding.

Another key conceptual element in the drawing together of a global picture of our universe with a basic mathematical model is the correspondence between emergent phenomena and definable relations. This gives us a framework within which to explain the particular forms of the physical constants and natural laws familiar to us from the standard model science currently provides. It goes some way towards substantiating Penrose’s [16, pp.106-107] ‘strong determinism’, according to which “all the complication, variety and apparent randomness that we see all about us, as well as the precise physical laws, are all exact and unambiguous consequences of one single coherent mathematical structure” — and repairs the serious failure of the standard model pointed to by researchers such

as Smolin and Woit. It also provides a hierarchical model of the fragmentation of the scientific enterprise.

This means that despite the causal connections between say particle physics and the study of living organisms, the corresponding disciplines are based on quite different basic entities and natural laws, and there is no feasible and informative reduction of one to another. The entities in one field may emerge through phase transitions characterised in terms of definable relations in the other, along with their distinct causal structures. In this context, it may be that the answer to Smolin's first 'great unsolved problem in theoretical physics' consists of an explanation of why there is no single theory (of the kind that makes useful predictions) combining general relativity and quantum theory.

The following table provides a summary of some of the main features of the Turing interpretation, drawing out parallels between scientific activity and what the Turing model provides. For further discussion of such issues, see [3], [5], [6], [7] and [8].

Science	Turing landscape
Physical entities treated as information	Structures information
Theories describing relations over the reals, enabling calculations	Functionals over the reals modelled on real computational capabilities
An extensive basic causal structure which is algorithmic	Models computable causal relations over the reals
Descriptions of globally emerging laws and constants elusive	Problems pinning down the nature of Turing invariance and definability
Features quantum ambiguity and non-locality	Explanation in terms of putative breakdown in Turing definability
Theoretical fragmentation involving phase transitions	Incomputability, and algorithmic relations over emergent objects

References

1. Bojowald, M.: Loop quantum cosmology. In: Ashtekar, A. (ed.) 100 Years of Relativity — Space-Time Structure: Einstein and Beyond, pp. 382–414. World Scientific, Singapore (2006)
2. Bombelli, L., Lee, J., Meyer, D., Sorkin, R.D.: Spacetime as a causal set. *Phys. Rev. Lett.* 59, 521–524 (1987)
3. Cooper, S.B.: Clockwork or Turing U/universe? - Remarks on causal determinism and computability. In: Cooper, S.B., Truss, J.K. (eds.) *Models and Computability*. London Mathematical Society Lecture Notes Series, vol. 259, pp. 63–116. Cambridge University Press, Cambridge (1999)
4. Cooper, S.B.: *Computability Theory*. Chapman & Hall/CRC, Boca Raton (2004)
5. Cooper, S.B.: Definability as hypercomputational effect. *Applied Mathematics and Computation* 178, 72–82 (2006)
6. Cooper, S.B.: How Can Nature Help Us Compute? In: Wiedermann, J., Tel, G., Pokorný, J., Bieliková, M., Štuller, J. (eds.) *SOFSEM 2006*. LNCS, vol. 3831, pp. 1–13. Springer, Heidelberg (2006)

7. Cooper, S.B.: Computability and emergence. In: Gabbay, M., Goncharov, S.S., Zakharyashev, M. (eds.) *Mathematical Problems from Applied Logic I. Logics for the XXIst Century*. Springer International Mathematical Series, vol. 4, pp. 193–231 (2006)
8. Cooper, S.B., Odifreddi, P.: Incomputability in Nature. In: Cooper, S.B., Goncharov, S.S. (eds.) *Computability and Models*, pp. 137–160. Kluwer Academic/Plenum, Dordrecht/New York (2003)
9. Einstein, A.: *Autobiographical Notes*. In: Schilpp, P. (ed.) *Albert Einstein: Philosopher-Scientist*. Open Court Publishing (1969)
10. Friedan, D.: A Tentative Theory of Large Distance Physics. *J. High Energy Phys.* JHEP10, 063 (2003)
11. Hodges, A., Turing, A.: *The Enigma*, Vintage, London, Melbourne, Johannesburg (1992)
12. Kreisel, G.: Mathematical logic: What has it done for the philosophy of mathematics? In: Schoenman, R. (ed.) *Bertrand Russell, Philosopher of the Century*, Allen and Unwin, London, pp. 201–272 (1967)
13. Kreisel, G.: Church's Thesis: a kind of reducibility axiom for constructive mathematics. In: Kino, A., Myhill, J., Vesley, R.E. (eds.) *Intuitionism and proof theory: Proceedings of the Summer Conference at Buffalo N.Y. 1968*, pp. 121–150. North-Holland, Amsterdam (1970)
14. Kuhn, T.S.: *The Structure of Scientific Revolutions*, 3rd edn. University of Chicago Press, Chicago (1996)
15. Newton, I.: *Philosophiæ Naturalis Principia Mathematica*, London (1687)
16. Penrose, R.: Quantum physics and conscious thought. In: Hiley, J., Peat, F.D. (eds.) *Quantum Implications: Essays in honour of David Bohm*, pp. 105–120. Routledge & Kegan Paul, London (1987)
17. Rogers Jr., H.: Some problems of definability in recursive function theory. In: Crossley, J.N. (ed.) *Sets, Models and Recursion Theory*. Proceedings of the Summer School in Mathematical Logic and Tenth Logic Colloquium, Leicester, pp. 183–201. North-Holland, Amsterdam (1965)
18. Smolin, L.: *The Trouble With Physics: The Rise of String Theory, the Fall of Science and What Comes Next*. Allen Lane/Houghton Mifflin, London, New York (2006)
19. Turing, A.: On computable numbers, with an application to the Entscheidungsproblem. Proceedings of the London Mathematical Society 42, 230–265 (1936-1937); Turing, A.M.: *Collected Works: Mathematical Logic*, 18–53 (reprint)
20. Turing, A.: Systems of logic based on ordinals. Proceedings of the London Mathematical Society 45, 161–228 (1939); Turing, A.M.: *Collected Works: Mathematical Logic*, 81–148 (reprint)
21. Turing, A.M.: *Collected Works: Mathematical Logic*. In: Gandy, R.O., Yates, C.E.M. (eds.). Elsevier, Amsterdam (2001)
22. Woit, P.: *Not Even Wrong: The Failure of String Theory and the Continuing Challenge to Unify the Laws of Physics*, Jonathan Cape, London (2006)

Strong Positive Reducibilities

Andrea Sorbi

University of Siena, 53100 Siena, Italy

sorbi@unisi.it

<http://www.dsmi.unisi.it/~sorbi/>

Abstract. The need of formalizing a satisfactory notion of relative computability of partial functions leads to enumeration reducibility, which can be viewed as computing with nondeterministic Turing machines using positive information. This paper is dedicated to certain reducibilities that are stronger than enumeration reducibility, with emphasis given to s -reducibility, which appears often in computability theory and applications. We review some of the most notable properties of s -reducibility, together with the main differences distinguishing the s -degrees from the e -degrees, both at the global and local level.

1 Motivations and Historical Background

Relative computability models computations using auxiliary information which is stored and made available by some external resource, or “oracle”. The most popular model of relative computability is provided by oracle Turing machines, introduced by Turing, [36]. One could think of the following variation on the classical model of an oracle Turing machine. The oracle stores information about a *total* function f : the n -th cell of the oracle tape contains the value $m = f(n)$; the computing agent from time to time needs auxiliary information of the form “Is $f(n) = m$?”, which is retrieved by accessing the n -th cell of the oracle tape. This provides a way of computing a partial function relatively to f : in this model of relative computability, access to the oracle database is immediate and complete.

We propose an immediate generalization of the above model, in the case when the oracle database stores data about a partial function ψ . Suppose that the computing agent needs to know “Is $\psi(n) = m$?”; in order to answer this question the computing agent gets access to the n -th cell of the oracle tape: how the computation proceeds from now on depends on whether the cell stores m (affirmative answer: it is indeed $\psi(n) = m$), or some $m' \neq m$ (negative answer: we definitely know $\psi(n) \neq m$), or no information is provided (i.e. $n \notin \text{domain}(\psi)$): in the last case the computation is stuck. This provides a very interesting model of relative computability, with a corresponding reducibility notion on partial functions, known as *Sasso reducibility* \leq_T ([31]; see also [32]): keeping Sasso’s notation, we use the same symbol as for Turing reducibility, since, as we will see, on total functions the two reducibilities agree. There is a reasonable machine model that goes with this idea: the way the information is organized and

made available by the oracle takes the form of an enumeration of the graph of ψ . Upon accessing the n -cell of the oracle tape, we wait for the oracle to enumerate a pair $\langle n, m' \rangle$; thus we wait forever if $n \notin \text{domain}(\psi)$. Notice that in this model, the computing agent has access to *positive* information (referring only to data that *are* in ψ), in that *negative* information is available only when it can be translated into positive information. However Sasso's model does not do justice to our intuitive idea of relative computability of partial functions. As argued by Myhill, [23], let A be any noncomputable set; then in Sasso reducibility the characteristic function c of A is not reducible to the semicharacteristic function of $A \oplus \bar{A}$, $s = \{(2n, 0) : n \in A\} \cup \{(2n + 1, 0) : n \in \bar{A}\}$ (where \bar{X} denotes the complement of a given set X) as if this were not the case then by monotonicity we would also have that c is reducible to $\lambda x. 0$, and then c would be computable. On the other hand, the function c is intuitively computable relative to s : on input n , it is enough to search the oracle database for the information $s(2n) = 0$ or $s(2n + 1) = 0$. It is then clear that in order to allow for situations like this one has to introduce some nondeterministic ingredients. This led Myhill [23] to suggest the following reducibility notion on partial functions: $\phi \leq_e \psi$ if there exists a c.e. set W (of coded triples), such that

$$\phi(n) = m \Leftrightarrow (\exists \text{ finite } D)[\langle \langle n, m \rangle, D \rangle \in W \text{ and } D \subseteq \psi].$$

Notice that in this case for the two functions above we have that $c \leq_e s$ as is easily seen: the reason for this lies in the fact that, in Sasso's model, an oracle Turing machine might get stuck after asking, for instance, the question "Is $s(2n) = 0$?" if $n \in \bar{A}$. In Myhill's model, on the contrary, the computing agent can be viewed as performing two computations in parallel: when needing information on a particular n , the two different possibilities, i.e. both $s(2n) = 0$ and $s(2n + 1) = 0$ are tested at the same time, only one computation eventually giving the output. In fact, Myhill reducibility can be characterized as *nondeterministic* Sasso reducibility, as observed by Sasso, Cooper and McEvoy (see [21] for a detailed proof): $\phi \leq_e \psi$ if and only if there is a nondeterministic oracle Turing machine which in Sasso's model computes ϕ from oracle ψ .

It must be observed that Davis, [10], had previously taken yet another approach to relative computability of partial functions, introducing some ingredients of nondeterminism. He defines ϕ to be computable relative to ψ if $\phi \leq_e \psi$ via a c.e. set W of triples satisfying the following consistency condition: if $\langle \langle n, m \rangle, D \rangle, \langle \langle n, m' \rangle, D' \rangle \in W$ and $m \neq m'$ then $D \cup D'$ is not single-valued, i.e. W maps partial functions to partial functions. However Davis reducibility \leq_{WT} does not pass Myhill's counterexample either, by an argument similar to the one given for Sasso reducibility above.

Another advantage of Myhill's definition is that it immediately extends to sets, ([29]). Any computably enumerable (c.e.) set W defines an *enumeration operator* (or e-operator), i.e. a mapping $\Phi : \mathcal{P}(\omega) \rightarrow \mathcal{P}(\omega)$, where ω denotes the set of natural numbers, and $\mathcal{P}(\omega)$ its power set: if $A \in \mathcal{P}(\omega)$ then

$$\Phi(A) = \{x : (\exists \text{ finite } D)[\langle x, u \rangle \in W \text{ and } D \subseteq A]\}.$$

Definition 1. *If $A = \Phi(B)$ for some e -operator Φ then we say that A is enumeration reducible to B (abbreviated by: A is e -reducible to B ; in symbols: $A \leq_e B$).*

Enumeration reducibility is what Polyakov and Rozinas, [28], call a “positive” reducibility, i.e. a reducibility in which the computing agent is typically allowed to access only positive information about the oracle set. In particular, to an oracle question like “ $n \in B?$ ”, one may in general expect an answer *only* if $n \in B$. If $n \notin B$ then the computing agent might get stuck waiting forever for this information. One can convincingly argue that e -reducibility is the most comprehensive positive reducibility, giving a formal counterpart to the idea of a set A being “computably enumerable relatively to” a set B , i.e. there is an effective procedure for enumerating A , given *any* enumeration of B . Here is a suitable machine model for e -reducibility. The computing agent is a Turing machine M , which step by step computably enumerates pairs $\langle x, D \rangle$ where x is a number and D is a finite set; the oracle is independently providing positive information about B by enumerating B (in an order and with a timing which only depend on the oracle); if among the pairs enumerated by M by step s there is a pair $\langle x, D \rangle$ and the elements of D have been already enumerated by the oracle, then (if not already done) the computing agent enumerates x at stage s : the set A consists exactly of the numbers that are enumerated in this way.

Our notations and terminology for computability theory are standard, and can be found for instance in [8], which contains also an excellent introduction to positive reducibilities. If \leq_r is a reducibility, we denote by \equiv_r the equivalence relation generated by \leq_r ; the r -degree of a set A , denoted by $\text{deg}_r(A)$, is the equivalence class of A under \equiv_r ; we order r -degrees by $\text{deg}_r(A) \leq_r \text{deg}_r(B)$, if $A \leq_r B$. All reducibilities \leq_r considered in this paper give rise to an upper semilattice \mathfrak{D}_r with least element given by $\mathbf{0}_r = \{W : W \text{ c.e.}\}$; the least upper bound is given by $\text{deg}_r(A) \vee \text{deg}_r(B) = \text{deg}_r(A \oplus B)$. Any positive reducibility \leq_r studied here can be viewed also as a reducibility on partial functions, via identification of partial functions with their graphs. Whether one works with sets or partial functions, the corresponding degree structures are isomorphic as for every set A , $A \equiv_r$ to the semicharacteristic function of A . The degree structure originated by a reducibility \leq_r will be denoted by $\langle \mathfrak{D}_r, \leq_r \rangle$; the Turing degrees will be denoted simply by $\langle \mathfrak{T}, \leq \rangle$.

2 Strong Enumeration Reducibilities

Whether we work with sets or partial functions, by a *strong enumeration reducibility*, we usually mean a positive reducibility that is stronger than enumeration reducibility. Accordingly, Sasso and Davis reducibilities can be viewed as strong enumeration reducibilities. Indeed, it immediately follows from the definitions that $\leq_T \subseteq \leq_{WT} \subseteq \leq_e$: the former inclusion is proper as follows from item [2] below; the latter inclusion is proper by a result due to Myhill and Shepherdson (see [29, Theorem 13.XIX]). Not much is known about the degree structures corresponding to these reducibilities, with the following notable exceptions. For a poset \mathfrak{P} , define $Th(\mathfrak{P})$ to denote the collection of all first order sentences σ (in

the language of partial orders with identity), which are true in \mathfrak{P} . Two posets (\mathfrak{P}_1, \leq_1) , (\mathfrak{P}_2, \leq_2) , are *elementarily equivalent* if $Th(\mathfrak{P}_1) = Th(\mathfrak{P}_2)$. Then:

1. \mathfrak{D}_T and \mathfrak{T} are not elementarily equivalent, ([5]): in \mathfrak{T} there is a cone made up of joins of minimal degrees, whereas this is not the case of \mathfrak{D}_T .
2. \mathfrak{D}_{WT} is elementarily equivalent neither with the Turing degrees \mathfrak{T} , nor with \mathfrak{D}_T , since in \mathfrak{D}_{WT} there is no minimal element (Gutteridge's proof, [14], for the e-degrees works fine here), although the structure is not dense (see proof of Theorem 1 below). On the other hand, one can embed in \mathfrak{D}_T every countable distributive lattice with 0 as an initial segment ([5]). As a consequence, $Th(\mathfrak{D}_T)$ is computably isomorphic to second order arithmetic ([5]).

Moreover, see ([20]), $Th(\mathfrak{D}_{WT})$ is computably isomorphic to second order arithmetic. For the proof of this claim, one may use the same coding techniques as the ones used in [33] for the e-degrees, by showing that every countable relation is uniformly definable with parameters, thus reducing second order quantification to quantification on elements of the structure.

Embedding the Turing degrees. If f and g are total functions then, see for instance [29, Corollary 9.XXIV],

$$f \leq g \Leftrightarrow f \leq_T g \Leftrightarrow f \leq_{WT} g \Leftrightarrow f \leq_e g$$

(recall that \leq denotes Turing reducibility). This shows that the Turing degrees \mathfrak{T} embed into the three degree structures $\mathfrak{D}_T, \mathfrak{D}_{WT}, \mathfrak{D}_e$ preserving 0 and suprema: the embedding identifies the T-degrees as the r-degrees of total functions for $r \in \{T, WT, e\}$.

Typically, strong enumeration reducibilities arise when there are constraints or bounds to how the computing agent can retrieve oracle information. A particularly interesting example is provided by *s-reducibility*: an *s-operator* is an enumeration operator containing only elements of the form $\langle x, D \rangle$, where D is empty or a singleton. In many practical situations in which we have $A \leq_e B$, we have in fact $A \leq_s B$ (including Myhill's example quoted above, in which $c \leq_s s$).

This gives a very nice and natural reducibility \leq_s , which is now being actively studied. It is worth noticing that $A \leq_s B$ if and only if there exists a computable function f such that (if $B \neq \omega$)

$$x \in \overline{A} \Leftrightarrow W_{f(x)} \subseteq \overline{B},$$

which takes us to the definition of Q-reducibility \leq_Q (introduced by Tennenbaum: see [29, p. 159]): in other words $A \leq_s B$ if and only if $\overline{A} \leq_Q \overline{B}$. Several interesting applications of Q-reducibility and s-reducibility to algebra and word problems of groups, [3], are known. Q-reducibility has also been studied in connection with abstract complexity theoretic questions: see for instance [13]. Contrary to Sasso and Davis reducibilities, s-reducibility does not agree with Turing reducibility on total functions: the only implication to hold is

$$f \leq_s g \Rightarrow f \leq g$$

thus the Π_1^0 s-degrees (isomorphic with the c.e. Q-degrees) are not a copy of the c.e. Turing degrees, as on Π_1^0 sets s-reducibility only implies Turing reducibility.

3 The Structure of the s-Degrees

The reducibility \leq_s is properly contained in \leq_e : as shown by Theorem 6 below, every nonzero e-degree contains at least two s-degrees.

Theorem 1. *In \mathcal{D}_s there is no atom, although the structure is not dense (in fact nontrivial empty intervals exist, in which the endpoints are Π_2^0 s-degrees).*

Proof. Gutteridge’s proof of the nonexistence of atoms in \mathcal{D}_e carries over directly to the s-degrees. As to nondensity, in their proof for the e-degrees Calhoun and Slaman, [4], build a nontrivial empty interval, whose endpoints have as representatives Π_2^0 sets A and B such that in fact $A \leq_1 B$. \square

If one has in mind to use \leq_s as a reducibility on partial functions, then one might ask what is the role of the total s-degrees. Well, they “determine” the whole structure, as shown by the following theorem.

Theorem 2. *The total s-degrees generate \mathcal{D}_s under infima. In other words, every s-degree is the infimum of two bigger total s-degrees.*

Proof. The proof is virtually the same as for the e-degrees, see [35]. In fact, if $\mathbf{a} \leq_s \mathbf{b}$ then we can find a total $\mathbf{c} \geq_s \mathbf{a}$ such that $\mathbf{a} = \mathbf{b} \wedge \mathbf{c}$. \square

Although they share the above features, \mathcal{D}_s and \mathcal{D}_e are in many respects very different from each other. A useful tool to measure how different two structures of the same signature are is to exhibit elementary differences:

Theorem 3. [38] *There is a nonzero s-degree \mathbf{a} such that*

$$(\forall \mathbf{b}, \mathbf{c})[\mathbf{a} \leq_s \mathbf{b} \vee \mathbf{c} \Rightarrow \mathbf{a} \leq_s \mathbf{b} \text{ or } \mathbf{a} \leq_s \mathbf{c}].$$

As a consequence, \mathcal{D}_e and \mathcal{D}_s are not elementarily equivalent.

Proof. Take the s-degree of any non c.e. retraceable set. Recall that a set $A = \{a_0 < a_1 < \dots\}$ is *retraceable* if there exists a partial computable function ψ such that for all n , $\psi(a_{n+1}) = a_n$ and $\psi(a_0) = a_0$. Then, [38], [2], if A is retraceable and $B \subseteq A$ is an infinite subset, then $A \leq_s B$, via the s-operator $\Gamma = \{\langle x, \{y\} \rangle : (\exists k)[\psi^k(y) \downarrow = x]\}$, where $\psi^k(y)$ denotes the k -th iteration of ψ starting from y . It follows, [38], that if A is infinite retraceable then for every B, C ,

$$A \leq_s B \oplus C \Rightarrow A \leq_s B \text{ or } A \leq_s C.$$

To see this, suppose that $A \leq_s B \oplus C$. Then by the s-operator witnessing the reduction, we can construct a computable function f such that, for every x ,

$$x \in A \Leftrightarrow W_{f(x)} \cap (B \oplus C) \neq \emptyset.$$

Let B_0, C_0 be the subsets of A consisting of those x for which $W_{f(x)}$ intersects $B \oplus \emptyset$ and $\emptyset \oplus C$, respectively. Then $B_0 \leq_s B$ and $C_0 \leq_s C$ and since A is infinite we have that either B_0 is infinite or C_0 is infinite. If for instance B_0 is infinite then since $B_0 \subseteq A$ we have $A \leq_s B_0 \leq_s B$.

The above property does not hold in \mathcal{D}_e , as can be seen by standard forcing techniques, and thus gives an elementary difference between \mathcal{D}_e and \mathcal{D}_s . \square

4 The Local Structure of the s-Degrees

We deal here with two degree structures (the e-degrees and the s-degrees) which are endowed with a jump operation, and so by *local structure* we mean the degrees below the jump of the least element: call this first jump $\mathbf{0}'_e$ or $\mathbf{0}'_s$ according to whether we are in the e-degrees or the s-degrees. (The e-jump was introduced and studied in [6] and [22]; for the s-jump see [25].) Without giving detailed definitions of the jump operations, suffice it to say that in both cases the local structures partition the Σ_2^0 sets: in particular $\mathbf{0}'_e = \text{deg}_e(\overline{K})$ and $\mathbf{0}'_s = \text{deg}_s(\overline{K})$. For this reason we often refer to the elements of these local structures as Σ_2^0 *degrees*. We recall that a set A is Σ_2^0 if and only if $A = \{x : (\exists t)(\forall s \geq t)[x \in A_s]\}$ for some computable sequence of sets $\{A_s\}$, called a Σ_2^0 *approximation to A*; a stage s in the approximation is *good* if $A_s \subseteq A$; the approximation is *good* if it has infinitely many good stages. For $r \in \{e, s\}$, we denote by \mathfrak{L}_r the local structure of \mathfrak{D}_r . We note that s-degrees consisting entirely of properly Σ_2^0 sets do exist. In fact, by the existence of e-degrees that are *downwards properly* Σ_2^0 (i.e. nonzero e-degrees \mathbf{a} such that for all nonzero $\mathbf{b} \leq_e \mathbf{a}$ we have that \mathbf{b} consists entirely of properly Σ_2^0 sets), and dually of e-degrees that are *upwards properly* Σ_2^0 , and even of e-degrees that are simultaneously downwards and upwards properly Σ_2^0 (first exhibited in [9]), one concludes that similar results hold in \mathfrak{L}_s .

It is still an open question which lattices can be embedded into \mathfrak{L}_s . On the positive side, it is possible to show that every countable distributive lattice is embeddable. As every countable distributive lattice is embeddable in the free Boolean algebra \mathfrak{B} on ω generators, it suffices to show that \mathfrak{B} is embeddable in \mathfrak{L}_s . We can present \mathfrak{B} as a Boolean algebra of infinite computable sets (except for the element 0, which is presented by \emptyset) and construct a Π_1^0 set A such that the assignment $S \mapsto \text{deg}_s(A_S)$ is the desired lattice embedding, where $S \in \mathfrak{B}$ and $A_S = \{\langle n, x \rangle \in A : n \in S\}$. This shows in fact:

Theorem 4. [25] *Every countable distributive lattice is embeddable in the Π_1^0 s-degrees, preserving 0.*

The local structure \mathfrak{L}_s is not distributive, though:

Theorem 5. *The nonmodular five elements lattice N_5 is embeddable in the Π_1^0 s-degrees, via an embedding that preserves the least element.*

The latter theorem was announced in [12], in the context of Q-reducibility. A proof using s-reducibility can be found in [25].

5 Structure of the s-Degrees within the e-Degrees

It is known that in every e-degree there is a greatest s-degree, [38]: if A is any set then $A \equiv_e K_A$ (where $K_A = \{x : x \in \Phi_x(A)\}$ and we refer to some effective listing $\{\Phi_x\}$ of the enumeration operators), and if $B \leq_e A$ then $B \leq_s K_A$ (in fact $B \leq_1 K_A$). Hence $\top_{\mathbf{a}} = \text{deg}_s(K_A)$ is the greatest s-degree within $\text{deg}_e(A)$. Moreover, if G_A is the set of good stages of some good Σ_2^0 approximation of a

Σ_2^0 set A , then $K_A \equiv_s G_A$, and G_A is hyperimmune, see [15]. We recall that a sequence of finite sets $\{F_n\}$ is a *disjoint weak (strong) array* if there exists a computable function f such that $F_n = W_{f(n)}$ ($F_n = D_{f(n)}$) and on distinct indices f lists disjoint sets. An infinite set X is *hyperhyperimmune (hyperimmune)* if for every disjoint weak (strong) array $\{F_n\}$, we have that $F_n \subseteq \overline{X}$ for some n ; if X contains no infinite c.e. set then X is said to be *immune*.

Theorem 6. [38] *Every nonzero e-degree $\mathbf{a} = \text{deg}_e(A)$ contains a set $B \leq_s K_A$: thus there exist at least two s-degrees within \mathbf{a} .*

Proof. It can be shown that no non-c.e. semirecursive set nor any immune set can be s-reduced to a superset of a simple set. Given a non c.e. A , take $B = S \cup \bigcup_{n \in K_A} F_n$, where S is Post’s simple set (a c.e. set which is coimmune but not cohyperrimmune), and $\{F_n\}$ is a disjoint strong array such that $F_n \cap \overline{S} \neq \emptyset$ for each n . Then $B \equiv_e K_A$, hence $B \leq_s K_A$, but on the other hand $K_A \not\leq_s B$: if $A \in \Sigma_2^0$, this follows from the fact that $G_A \leq_s K_A$, and G_A is immune; otherwise by [17, Theorem 3.6] take a semirecursive set R such that $R \leq_e K_A \leq R$ (recall that \leq denotes Turing reducibility), hence $R \leq_s K_A$; but since we assume $K_A \notin \Delta_2^0$, R is not c.e., and $K_A \leq_s B$ would imply $R \leq_s B$, a contradiction. \square

The situation is much clearer if we limit our attention to Σ_2^0 e-degrees.

Theorem 7. [18] *Every nonzero Σ_2^0 e-degree \mathbf{a} consists of infinitely many s-degrees. In fact within \mathbf{a} there is no least s-degree, and one can embed any countable partial order.*

Particular cases of this result had been shown by Watson, [37] using the priority method. Recently Kent, [18], has given a priority-free uniform proof that holds of every nonzero Σ_2^0 e-degree. Harris, [15], has further clarified the situation. Using the fact that for every Σ_2^0 set A , $K_A \equiv_s G_A$, by an argument similar to the density proof for the Σ_2^0 e-degrees as presented in [19], he shows the following density result which extends the second claim of Theorem 7 and also gives, as a corollary, upwards density of the Σ_2^0 s-degrees, first proved in [25]:

Theorem 8. *In any nonempty interval of Σ_2^0 s-degrees $(\mathbf{b}, \top_{\mathbf{a}})$, one can embed any countable partial order.*

6 The First Order Theory of the Σ_2^0 s-Degrees

The local structure \mathcal{L}_s is not elementarily equivalent to \mathcal{L}_e . If we look at Theorem 3 again, we see that property therein pointed out gives indeed a difference also at the level of local structures: for instance, it is not difficult to see that for every total function f there is a retraceable B such that $B \equiv_e f$ and $f \leq_s B$: take $B = \{\langle f(0), \dots, f(n-1) \rangle : n \in \omega\}$. Applying this to the characteristic function $c_{\overline{K}}$ we have that $\overline{K} \equiv_s B$ for some retraceable B . Thus:

Corollary 1. *\mathcal{O}'_s is join-irreducible. Hence $\text{Th}(\mathcal{L}_s) \neq \text{Th}(\mathcal{L}_e)$.*

We now turn to show that $Th(\mathfrak{L}_s)$ is undecidable. We do this by showing how to code any Σ_4^0 set in an independent family of Σ_2^0 s-degrees. Recall that in an upper semilattice $\langle U, \leq, \vee \rangle$, a countable $A \subseteq U$ is called *independent* if for every $a \in A$ and any finite $F \subseteq A$, we have that $a \leq \vee F$ implies $a \in F$. The key result is:

Theorem 9. [1] *There is an independent set of Σ_2^0 s-degrees that is first-order definable with parameters in \mathfrak{L}_s . In fact, there exist 2-c.e. s-degrees (i.e. degrees containing sets of the form $X \setminus Y$, where X and Y are c.e.) $\{\mathbf{a}_i\}_{i \in \omega}$, \mathbf{a} , \mathbf{b} , \mathbf{c} , such that the \mathbf{a}_i 's form an independent set and are the minimal solutions of the inequalities*

$$\mathbf{x} \leq_s \mathbf{a} \text{ and } \mathbf{b} \leq_s \mathbf{x} \vee \mathbf{c},$$

i.e. $\mathbf{b} \leq_s \mathbf{a}_i \vee \mathbf{c}$ for every i , and for every \mathbf{x} ,

$$\mathbf{x} \leq_s \mathbf{a} \text{ and } \mathbf{b} \leq_s \mathbf{x} \vee \mathbf{c} \Rightarrow (\exists i)[\mathbf{a}_i \leq_s \mathbf{x}].$$

The proof is by an infinite priority argument.

Consider now such an independent antichain, and call $\alpha(v, \bar{\mathbf{p}})$, with parameters $\bar{\mathbf{p}}$, the first order formula defining the antichain. By the Exact Degree Theorem for the Σ_2^0 e-degrees ([24]) which after inspection holds for the s-degrees as well, every Σ_4^0 set S can be uniformly associated with an s-degree \mathbf{d} such that $S = S_{\mathbf{d}} = \{i : \mathbf{a}_i \leq_s \mathbf{d}\}$ and thus (with $\bar{\mathbf{p}}$ being the relevant list of parameters)

$$S_{\mathbf{d}} \subseteq S_{\mathbf{e}} \Leftrightarrow \mathfrak{L}_s \models (\forall a) ((\alpha(a, \bar{\mathbf{p}}) \wedge a \leq \mathbf{d}) \rightarrow a \leq \mathbf{e}).$$

Therefore the first order theory of the poset $\langle \Sigma_4^0\text{-sets}, \subseteq \rangle$ (which is known to be hereditarily undecidable, see [16]) is elementarily definable with parameters in the Σ_2^0 s-degrees, giving that the first order theory of \mathfrak{L}_s is undecidable.

Inspection of the proof, and the fact that for every 2-c.e. set C there exists a Π_1^0 -set D such that $C \equiv_s D$, ([25]), allow to conclude that there is an independent set of Π_1^0 s-degrees that is first-order definable with parameters. Together with a suitable version of the Exact Degree Theorem for Π_1^0 s-degrees we thus obtain a different proof of a result in [11], stating that the first order theory of the c.e. Q-degrees is undecidable:

Corollary 2. *The first order theory of the Π_1^0 s-degrees is undecidable.*

7 s-Degrees and Immunity Properties

s-reducibility and its sibling Q-reducibility can be fruitfully used to study immunity properties of sets (see definitions at the beginning of Section 5). Following previous work of other authors (see for instance [13] and [34]: a set A is hyperhypersimple, i.e. c.e. and cohyperhyperimmune, if and only no c.e. superset of B is Q-complete), Omanadaze and Sorbi, [26], prove that a Δ_2^0 set A is hyperhyperimmune if and only if $\bar{K} \not\leq_s B$ for every infinite Δ_2^0 subset B of A . (Harris, personal communication, has pointed out that this characterization extends to the Σ_2^0 hyperhyperimmune sets as well.) Moreover, as already observed, [15], the top s-degree in any nonzero e-degree is hyperimmune (i.e. it contains an hyperimmune set), hence $\text{deg}_s(\bar{K})$ is hyperimmune. However:

Theorem 10. [26] $\text{deg}_s(\overline{K})$ does not contain any Δ_2^0 hyperhyperimmune set.

It is known that the immune and hyperimmune e-degrees are upwards closed, [30]. The following shows that neither the immune nor the hyperimmune s-degrees are upwards closed:

Theorem 11. [27] There are Δ_2^0 s-degrees $\mathbf{a} \leq_s \mathbf{b}$ such that \mathbf{a} is hyperimmune, but \mathbf{b} does not contain any immune set.

8 Conclusions

We have focussed our attention on some features of the strong form of enumeration reducibility known as s-reducibility, which we think are specially attractive and interesting. A lot has still to be done towards a better understanding of the corresponding degree structure, both at the global and local level. For instance: Does every nonzero e-degree contain infinitely many s-degrees? Are the Σ_2^0 s-degrees dense (raised by [7]; the Π_1^0 s-degrees are dense by [11])?

References

1. Affatato, M.L., Kent, T.F., Sorbi, A.: Undecidability of local structures of s-degrees and Q-degrees. Tbilisi Mathematical Journal 1, 15–32 (2008)
2. Arslanov, M.M.: On a class of hypersimple incomplete sets. Mat. Zametki 38, 872–874, 984–985 (1985) (English translation)
3. Belegradek, O.: On algebraically closed groups. Algebra i Logika 13(3), 813–816 (1974)
4. Calhoun, W.C., Slaman, T.A.: The Π_2^0 e-degrees are not dense. J. Symbolic Logic 61, 1364–1379 (1996)
5. Casalegno, P.: On the T-degrees of partial functions. JSL 50, 580–588 (1985)
6. Cooper, S.B.: Partial degrees and the density problem. Part 2: The enumeration degrees of the Σ_2 sets are dense. J. Symbolic Logic 49, 503–513 (1984)
7. Cooper, S.B.: Enumeration reducibility, nondeterministic computations and relative computability of partial functions. In: Ambos-Spies, K., Müller, G., Sacks, G.E. (eds.) Recursion Theory Week, Oberwolfach 1989. Lecture Notes in Mathematics, vol. 1432, pp. 57–110. Springer, Heidelberg (1990)
8. Cooper, S.B.: Computability Theory. Chapman & Hall/CRC Mathematics, Boca Raton/London (2003)
9. Cooper, S.B., Copestate, C.S.: Properly Σ_2 enumeration degrees. Z. Math. Logik Grundlag. Math. 34, 491–522 (1988)
10. Davis, M.: Computability and Unsolvability. Dover, New York (1982)
11. Downey, R.G., Laforte, G., Nies, A.: Computably enumerable sets and quasi-reducibility. Ann. Pure Appl. Logic 95, 1–35 (1998)
12. Fischer, P., Ambos-Spies, K.: Q-degrees of r.e. sets. J. Symbolic Logic 52(1), 317 (1985)
13. Gill III, J.T., Morris, P.H.: On subcreative sets and S-reducibility. J. Symbolic Logic 39(4), 669–677 (1974)
14. Gutteridge, L.: Some Results on Enumeration Reducibility. PhD thesis, Simon Fraser University (1971)

15. Harris, C.M.: Good Σ_2^0 singleton degrees and density (to appear)
16. Herrmann, E.: The undecidability of the elementary theory of the lattice of recursively enumerable sets. In: Frege conference, 1984, Schwerin, pp. 66–72. Akademie-Verlag, Berlin (1984)
17. Jockusch Jr., C.G.: Semirecursive sets and positive reducibility. *Trans. Amer. Math. Soc.* 131, 420–436 (1968)
18. Kent, T.F.: s-degrees within e-degrees. In: Agrawal, M., Du, D.-Z., Duan, Z., Li, A. (eds.) TAMC 2008. LNCS, vol. 4978, pp. 579–587. Springer, Heidelberg (2008)
19. Lachlan, A.H., Shore, R.A.: The n -REA enumeration degrees are dense. *Arch. Math. Logic* 31, 277–285 (1992)
20. Marsibilio, D., Sorbi, A.: Global properties of strong enumeration reducibilities (to appear)
21. McEvoy, K.: The Structure of the Enumeration Degrees. PhD thesis, School of Mathematics, University of Leeds (1984)
22. McEvoy, K.: Jumps of quasi-minimal enumeration degrees. *J. Symbolic Logic* 50, 839–848 (1985)
23. Myhill, J.: A note on degrees of partial functions. *Proc. Amer. Math. Soc.* 12, 519–521 (1961)
24. Nies, A.: A uniformity of degree structures. In: Sorbi, A. (ed.) *Complexity, Logic and Recursion Theory*, pp. 261–276. Marcel Dekker, New York (1997)
25. Omanadze, R.S., Sorbi, A.: Strong enumeration reducibilities. *Arch. Math. Logic* 45(7), 869–912 (2006)
26. Omanadze, R.S., Sorbi, A.: A characterization of the Δ_2^0 hyperhyperimmune sets. *J. Symbolic Logic* 73(4), 1407–1415 (2008)
27. Omanadze, R.S., Sorbi, A.: Immunity properties of s-degrees (to appear)
28. Polyakov, E.A., Rozinas, M.G.: Enumeration reducibilities. *Siberian Math. J.* 18(4), 594–599 (1977)
29. Rogers Jr., H.: *Theory of Recursive Functions and Effective Computability*. McGraw-Hill, New York (1967)
30. Rozinas, M.G.: Partial degrees of immune and hyperimmune sets. *Siberian Math. J.* 19, 613–616 (1978)
31. Sasso, L.P.: Degrees of Unsolvability of Partial Functions. PhD thesis, University of California, Berkeley (1971)
32. Sasso, L.P.: A survey of partial degrees. *J. Symbolic Logic* 40, 130–140 (1975)
33. Slaman, T., Woodin, W.: Definability in the enumeration degrees. *Arch. Math. Logic* 36, 225–267 (1997)
34. Solov'ev, V.D.: Q-reducibility and hyperhypersimple sets. *Veroyatn. Metod. i Kibern.* 10-11, 121–128 (1974)
35. Sorbi, A.: Sets of generators and automorphism bases for the enumeration degrees. *Ann. Pure Appl. Logic* 94(3), 263–272 (1998)
36. Turing, A.M.: Systems of logic based on ordinals. *Proc. London Math. Soc.* 45, 161–228 (1939)
37. Watson, P.: On restricted forms of enumeration reducibility. *Ann. Pure Appl. Logic* 49, 75–96 (1990)
38. Zacharov, S.D.: e - and s -degrees. *Algebra and Logic* 23, 273–281 (1984)

Fixed-Parameter Algorithms for Graph-Modeled Data Clustering

Jiong Guo

Institut für Informatik, Friedrich-Schiller-Universität Jena
Ernst-Abbe-Platz 2, D-07743 Jena, Germany
jiong.guo@uni-jena.de

Abstract. We survey some practical techniques for designing fixed-parameter algorithms for NP-hard graph-modeled data clustering problems. Such clustering problems ask to modify a given graph into a union of dense subgraphs. In particular, we discuss (polynomial-time) kernelizations and depth-bounded search trees and provide concrete applications of these techniques. After that, we shortly review the use of two further algorithmic techniques, iterative compression and average parameterization, applied to graph-modeled data clustering. Finally, we address some challenges for future research.

1 Introduction

Clustering a given set of objects according to a given similarity or distance measure requires to partition the input objects into homogeneous and well-separated subsets. This problem finds applications in many areas of computational biology, such as analyzing gene expression [22, 23], proteins [19], gene networks [24], etc. Using graph vertices to represent objects—such as genes or proteins—and adding edges between two vertices iff the interrelation between the two corresponding objects exceeds some threshold value, a clustering with respect to such a graph is a disjoint union of dense subgraphs, also called clusters, such that there are few or no edges between the clusters. A graph is dense if the vertices in it are highly connected. Here, a dense group represents a set of highly similar objects. When formulated as a graph modification problem, one thus asks for a minimum-cardinality set of edge modifications such that in the resulting graph every connected component is a cluster:

H-CLUSTER EDITING:

Input: An undirected graph $G = (V, E)$, a density measure H , and an integer $k \geq 0$.

Task: Decide whether there is a set of at most k edge modifications (insertions or deletions) to transform G into a H -cluster graph, that is, a graph in which every connected component satisfies H .

If the given objects admit a perfect cluster structure, namely, they form well-separated groups and, inside each group, the objects are highly interrelated,

then the graph G should be a disjoint union of dense subgraphs. However, for biological data, the input graph G usually is corrupted and we have to “correct” it under the parsimony criterion to reconstruct the “best” clustering. This is exactly what Π -Cluster Editing asks for.

One of the most prominent problems in this context is the NP-hard CLUSTER EDITING problem (also known as CORRELATION CLUSTERING) [3, 23], where the required density measure is $\Pi :=$ “being a clique”. CLUSTER EDITING has been intensively studied from the viewpoints of polynomial-time approximability as well as parameterized algorithmics [2, 5, 13]. Since the density requirement of “being a clique” has been often criticized for its overly restrictive nature and modeling disadvantages [9, 21], several other natural relaxations of cliques have been considered in the literature, including s -defective cliques [25], s -plexes [21], μ -cliques [1], etc. Already CLUSTER EDITING being NP-hard [3, 23], we cannot expect algorithms that can solve Π -Cluster Editing for more general Π ’s efficiently. In practice, heuristic algorithms, approximation algorithms or similar techniques have been employed to cope with the computational intractability of NP-hard problems. However, there are many scenarios where the aforementioned techniques cannot provide satisfactory solutions. For instance, they typically fail when an optimal solution of a computationally hard problem is sought and the solving algorithm should be reasonably efficient. Here, *fixed-parameter algorithms* come into play, where one basically asks for exact algorithms whose running time is exponential only with respect to a certain parameter k ; in our setting k denotes the number of modification operations. Thus, a Π -CLUSTER EDITING problem is called *fixed-parameter tractable (FPT)* if it can be solved in $f(k) \cdot |V|^{O(1)}$ time [20].

In this survey, several practically relevant techniques for designing fixed-parameter algorithms for Π -CLUSTER EDITING are addressed, in particular, *kernelization* (polynomial-time data reduction with provable performance guarantee) and *depth-bounded search trees* that are based on *forbidden subgraph characterizations*. Later, we briefly review two further algorithmic techniques and conclude with some directions for future research.

Basic graph notations. We only consider *undirected* graphs $G = (V, E)$, where $n := |V|$ and $m := |E|$. The (*open*) *neighborhood* $N(v)$ of a vertex $v \in V$ is the set of vertices that are adjacent to v in G and $N^2(v)$ is the set of vertices that have distance exactly two to v . The *degree* of a vertex v , denoted by $\deg(v)$, is the cardinality of $N(v)$. For a set U of vertices, $N(U) := \bigcup_{v \in U} N(v) \setminus U$. We use $N[v]$ to denote the *closed* neighborhood of v , that is, $N[v] := N(v) \cup \{v\}$. For a set of vertices $V' \subseteq V$, the *induced subgraph* $G[V']$ is the graph over the vertex set V' with edge set $\{\{v, w\} \in E \mid v, w \in V'\}$. For $V' \subseteq V$, we use $G - V'$ as an abbreviation for $G[V \setminus V']$ and for a vertex $v \in V$ let $G - v$ denote $G - \{v\}$.

2 Kernelizations

To solve NP-hard problems, polynomial-time preprocessing is a natural and promising approach. Preprocessing is based on data reduction techniques that

take a problem's input instance and try to perform a reduction to a smaller, equivalent problem instance, called *reduced* instance. Many practical examples have demonstrated the usefulness of the data reduction techniques [15, 17]. Data reduction and problem kernelization results can explain, and *prove*, why preprocessing works so well in many practical applications. The idea is to prove upper bounds on the size of reduced problem instances, which are then called *problem kernels*. These upper bounds shall be functions solely depending on a parameter that typically is related with the size of a solution set of the problem under study. The formal definition of problem kernels and kernelization reads as follows: Let (G, k) be an instance of a parameterized problem where k denotes the parameter. A *reduction to a problem kernel* (or *kernelization*) means to replace (G, k) by a *reduced* instance (G', k') called *problem kernel* in polynomial time such that (1) $k' \leq k$, (2) $|G'| \leq g(k)$ for some function g only depending on k , and (3) (G, k) is a yes-instance if and only if (G', k') is a yes-instance.

In the following, we give two concrete applications of kernelization techniques.

General Data Reduction Rules and Average-s-Plex Editing. First, we present two general data reduction rules that apply to Π -CLUSTER EDITING for all considered density measures Π but do not necessarily give a problem kernel. Then, we show by the example of AVERAGE- s -PLEX EDITING that these rules, however, can lead to fixed-parameter tractability results.

General rules. The following rule is clearly true and can be carried out in polynomial time:

Rule 1. *Remove connected components that satisfy Π from G .*

The second rule is based on the easy-to-see observation that, if there are more than k edge-disjoint paths between two vertices u and v , then u and v must end up in the same Π -cluster after performing at most k edge modifications to the input graph G . Then, we can merge u and v into a “super-vertex”. To describe this rule, we introduce weight functions for the vertices and edges. For each super-vertex $\mathcal{X} \in V$, we define a set $V_{\mathcal{X}}$ and two weight functions $\sigma(\mathcal{X})$ and $\delta(\mathcal{X})$: $V_{\mathcal{X}}$ denotes the set of vertices merged into \mathcal{X} , $\sigma(\mathcal{X}) := |V_{\mathcal{X}}|$, and $\delta(\mathcal{X})$ is equal to the number of edges between the vertices in $V_{\mathcal{X}}$. Note that, for each of the original vertices u in V , we can set $V_u = \{u\}$, $\sigma(u) := 1$ and $\delta(u) := 0$. Moreover, for an edge e between two vertices u and v , we define $\omega(e)$ to be the number of the edges between V_u and V_v . The next rule reads as follows:

Rule 2. *If G contains two vertices u and v such that $\omega(\{u, v\}) > k$ or u and v have more than k common neighbors, then remove u from G and set*

- $\sigma(v) := \sigma(u) + \sigma(v)$,
- $\delta(v) := \delta(u) + \delta(v) + \omega(\{u, v\})$, and
- $\omega(\{v, w\}) := \omega(\{v, w\}) + \omega(\{u, w\})$ for each $w \in V$.

Observe that this rule does not strictly follow the definition of data reduction rules. Its application to an instance of Π -CLUSTER EDITING results in an instance of some kind of weighted version of Π -CLUSTER EDITING. However, this

rule may not only be very useful in practice, but also leads to fixed-parameter tractability results, as in the case of AVERAGE- s -PLEX EDITING.

Average- s -Plex Editing. A connected graph $H = (V_H, E_H)$ is an *average- s -plex* if the average degree of H is at least $|V_H| - s$. In AVERAGE- s -PLEX EDITING, we are given an undirected graph $G = (V, E)$ and an integer $k \geq 0$, and the task is to decide whether it is possible to transform G into a vertex-disjoint union of average- s -plexes for a fixed integer $s \geq 1$. It is not hard to see that Rule 2 can be applied to AVERAGE- s -PLEX EDITING and results in an instance of a weighted version of this problem. This weighted version is defined as follows.

Extend the function $\sigma(\mathcal{X})$ for super-vertices \mathcal{X} to vertex sets S , namely, $\sigma(S) := \sum_{v \in S} \sigma(v)$. The average degree $\bar{d}(G)$ of a connected graph $G = (V, E)$ with vertex weights and edge weights is defined as follows:

$$\bar{d}(G) = \frac{2 \sum_{v \in V} \delta(v) + \sum_{v \in V} \sum_{u \in N(v)} \omega(\{u, v\})}{\sigma(V)}.$$

We say then that a graph is an average- s -plex graph if for every connected component $C = (V_C, E_C)$ the average degree $\bar{d}(C)$ is at least $\sigma(V_C) - s$. The weighted version of AVERAGE- s -PLEX EDITING, called WEIGHTED AVERAGE- s -PLEX EDITING, is defined as follows: Given a graph $G = (V, E)$ with two vertex weight functions $\sigma : V \rightarrow [1, n]$ and $\delta : V \rightarrow [0, n^2]$, an edge weight function $\omega : E \rightarrow [1, n^2]$, and a nonnegative integer k , one asks whether there is a set of edge modifications S with $|S| \leq k$ such that applying S to G yields an average- s -plex graph.

For such a weighted graph $G = (V, E)$, we have four kinds of edge modifications: we can increase $\omega(\{u, v\})$ by one for an edge $\{u, v\} \in E$, decrease $\omega(\{u, v\})$ by one for an edge $\{u, v\} \in E$, delete $\{u, v\} \in E$ with $\omega(\{u, v\}) = 1$, and add some edge $\{u, v\}$ to E and set $\omega(\{u, v\}) := 1$. Each of these edge operations has cost one, and the overall cost of an edge modification set S is thus exactly $|S|$. We can easily reduce an instance $(G = (V, E), k)$ of AVERAGE- s -PLEX EDITING to an instance of the weighted version, by setting $\sigma(v) := 1$ and $\delta(v) := 0$ for each $v \in V$, and $\omega(e) := 1$ for each edge $e \in E$. Note that this reduction is parameter-preserving, that is, s and k are not changed.

Clearly, Rule 2 is a data reduction rule for WEIGHTED AVERAGE- s -PLEX EDITING, leading to a graph with a bounded number of vertices.

Theorem 1 ([16]). *A yes-instance (G, k) of WEIGHTED AVERAGE- s -PLEX EDITING that is reduced with respect to Rules 1 and 2 contains at most $4k^2 + 8sk$ vertices.*

We obtain a fixed-parameter algorithm for WEIGHTED AVERAGE- s -PLEX EDITING as follows. First, we exhaustively apply the reduction rules, which can clearly be done in polynomial time. If the reduced instance contains more than $4k^2 + 8sk$ vertices, then it is a no-instance. Otherwise, we can solve the problem with running time only depending on s and k , for example, by brute-force generation of all possible partitions of the graph. The fixed-parameter tractability of AVERAGE- s -PLEX EDITING then directly follows from the described reduction to WEIGHTED AVERAGE- s -PLEX EDITING.

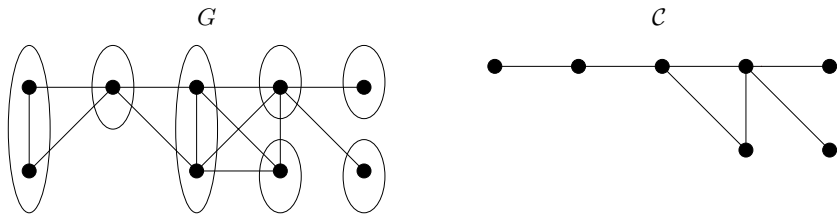


Fig. 1. A graph G and its critical clique graph \mathcal{C} . Ovals denote the critical cliques of G .

Cluster Editing. As mentioned in Section [11](#), CLUSTER EDITING is the special case of Π -CLUSTER EDITING with $\Pi :=$ “being a clique”. This means that, given a graph $G = (V, E)$ and $k \geq 0$, CLUSTER EDITING asks to decide whether G can be transformed by at most k edge deletions and insertions into a cluster graph, that is, a vertex-disjoint union of cliques. We describe a kernelization for this problem which results in a graph with $O(k)$ vertices [\[14\]](#). First, we introduce the concepts of *critical clique* and *critical clique graph*.

Definition 1. A critical clique of a graph G is a clique K where all vertices of K have the same sets of neighbors in $V \setminus K$, and K is maximal under this property. Given a graph $G = (V, E)$, let \mathcal{K} be the collection of its critical cliques. Then the critical clique graph \mathcal{C} is the graph $(\mathcal{K}, E_{\mathcal{C}})$ with $\{K_i, K_j\} \in E_{\mathcal{C}} \iff \forall u \in K_i, v \in K_j : \{u, v\} \in E$. That is, the critical clique graph has the critical cliques as nodes, and two nodes are connected iff the corresponding critical cliques together form a larger clique.

See Figure [1](#) for an example of a graph and its critical clique graph. The critical clique graph can be constructed in $O(m + n)$ time [\[14\]](#). Note that we use the term *nodes* for the vertices in the critical clique graph. Moreover, we use $K(v)$ to denote the critical clique containing vertex v and use $V(K)$ to denote the set of vertices contained in the critical clique $K \in \mathcal{K}$.

The basic idea behind introducing critical cliques is the following: suppose that the input graph $G = (V, E)$ has a solution with at most k edge modifications. Then, at most $2k$ vertices are “affected” by these edge modifications, that is, they are endpoints of edges added or deleted. Thus, in order to give a size bound on V depending only on k , it remains to upper-bound the size of the “unaffected” vertices. The central observation is that, in the cluster graph obtained after making the at most k edge modifications, the unaffected vertices contained in one clique must form a critical clique in the original graph G . By this observation, it seems easier to derive data reduction rules working for the critical cliques and the critical clique graph than to derive rules directly working on the input graph. Concerning critical cliques, one can show that there is always a solution that does not divide critical cliques. Moreover, large critical cliques, together with their neighbors, must form clusters in the final cluster graph.

Rule 3. If a critical clique K contains more vertices than all the critical cliques in $N_{\mathcal{C}}(K) \cup N_{\mathcal{C}}^2(K)$ together, then construct a clique C consisting of $V(K)$

and $V(K')$ for all $K' \in N_C(K)$, remove all vertices in C from G , and decrease the parameter k by the number of edge insertions and deletions needed to construct C and to separate C from the rest of G .

Combining Rules 1 and 3 gives the following problem kernel.

Theorem 2 ([14]). *If a reduced graph for CLUSTER EDITING has more than $6k$ vertices, then it has no solution with at most k edge modifications.*

Independently, Fellows et al. [11] achieved also the $6k$ -vertex kernel by using a different approach. By adding a more intricate data reduction rule, the problem kernel size can be improved to at most $4k$ vertices [14].

3 Forbidden Subgraph Characterization and Search Tree

The basic idea behind the depth-bounded search tree technique is to organize the systematic and exhaustive exploration of the search space in a tree-like manner. More precisely, given an instance (G, k) of Π -CLUSTER EDITING, search tree algorithms replace (G, k) by a finite set \mathcal{C} of instances (G_i, k_i) with $1 \leq i \leq |\mathcal{C}|$ and $k_i < k$ specified by some *branching rules*. If G_i for an i is not a Π -cluster graph, then the algorithm recursively applies this replacing procedure to (G_i, k_i) . The algorithm terminates when the replacing procedure is no longer applicable, that is, there is a G_i being a Π -cluster graph or $k_i < 0$. The recursive application of the replacing procedure can be illustrated in a tree structure, yielding a *search tree*. The depth of the search tree is bounded from above by a function of the parameter and its size is clearly $O(|\mathcal{C}|^k)$.

Many graph properties allow for characterizations in terms of forbidden subgraphs.

Definition 2. *Let \mathcal{F} be a collection of graphs. A graph property Π can be characterized by the forbidden induced subgraphs in \mathcal{F} iff each graph having Π contains no graph in \mathcal{F} as an induced subgraph. A graph having Π is called \mathcal{F} -free.*

It is not hard to observe that *finite* forbidden subgraph characterizations lead to the fixed-parameter tractability of the corresponding graph modification problems: Given a forbidden subgraph G' , a branching rule considers all possible ways to destroy G' ; in each case, an edge is deleted from or added to G' . Therefore, the search tree size depends directly on the size of G' .

For many density measures Π , Π -cluster graphs admit such finite forbidden subgraph characterizations. In the following, we give two applications of the search tree technique that is based on forbidden subgraph characterizations.

Cluster Editing. For $\Pi :=$ “being a clique”, it is easy to observe that the only forbidden subgraph is an induced path of three vertices. An $O(3^k)$ -size search tree follows immediately from this characterization: If the graph G is already a union of disjoint cliques, then we are done: Report the solution and return; otherwise, if $k \leq 0$, then we cannot find a solution in this branch of the search tree. Otherwise,

identify $u, v, w \in V$ with $\{u, v\} \in E$, $\{u, w\} \in E$, but $\{v, w\} \notin E$. Recursively call the branching procedure on the following three instances consisting of graphs $G' = (V, E')$ with nonnegative integer k' : (B1) $E' := E - \{u, v\}$ and $k' := k - 1$, (B2) $E' := E - \{u, w\}$ and $k' := k - 1$, and (B3) $E' := E + \{v, w\}$ and $k' := k - 1$. The search tree size can be significantly reduced. More specifically, a more sophisticated branching strategy gives a search tree size of $O(2.27^k)$ [13]. To this end, we distinguish two cases concerning three vertices $u, v, w \in V$ with $\{u, v\} \in E$, $\{u, w\} \in E$, but $\{v, w\} \notin E$: (C1) Vertices v and w do not share a common neighbor, that is, $\nexists x \in V, x \neq u : \{v, x\} \in E$ and $\{w, x\} \in E$; (C2) Vertices v and w have a common neighbor $x \neq u$.

The key observation for the improvement is the following observation, which implies, regarding case (C1), a branching into two cases (B1) and (B2) suffices.

Lemma 1 ([13]). *Given a graph $G = (V, E)$, a nonnegative integer k and $u, v, w \in V$ with $\{u, v\} \in E$, $\{u, w\} \in E$, but $\{v, w\} \notin E$. If v and w do not share a common neighbor besides u , then branching case (B3) cannot yield a better solution than both cases (B1) and (B2), and can therefore be omitted.*

For case (C2), the standard branching into three subcases can be avoided by taking a common neighbor of v and w into account [13]. Combining the analysis for cases (C1) and (C2), one can show the following.

Theorem 3 ([13]). *CLUSTER EDITING can be solved by a search tree algorithm with search tree size $O(2.27^k)$.*

Note that, by considering more complicated case distinction, Böcker et al. [5] achieved a search tree of size $O(1.82^k)$.

s-Defective Clique Editing. A connected graph $G = (V, E)$ is an s -defective clique if $|E| \geq |V| \cdot (|V| - 1)/2 - s$ for an integer $s \geq 0$. An s -defective clique graphs consists of connected components that are s -defective cliques. In the following, we present a forbidden subgraph characterization of s -defective clique graphs for any $s \geq 0$ as well as a search tree algorithm that makes use of this characterization. First, we show that s -defective clique graphs are characterized by forbidden subgraphs with $O(s)$ vertices. Note that, if $s = 0$, then s -defective clique graphs are equal to cluster graphs and the only forbidden subgraph is a path induced by three vertices, as shown above. Here, a graph is called a *minimal* forbidden subgraph if all its subgraphs are s -defective cliques.

Theorem 4 ([16]). *For $s \geq 1$, any minimal forbidden induced subgraph of s -defective clique graphs contains at most $2(s + 1)$ vertices.*

The forbidden subgraph characterization given in Theorem 4 directly leads to a search tree algorithm for s -DEFECTIVE CLIQUE EDITING: Given a graph $G = (V, E)$ that is not an s -defective clique graph, one can find in $O(nm)$ time a minimal forbidden subgraph [16]. Then, branch into all possibilities of adding or deleting an edge between two vertices contained in the forbidden induced subgraph. Since the number of vertices of a minimal forbidden subgraph is bounded by $2s + 2$, there are at most $\binom{2s+2}{2}$ cases and in each case the parameter k is

decreased by one. Hence, the size of the search tree is bounded by $O\left(\binom{2s+2}{2}^k\right)$. Putting all together leads to the following.

Theorem 5 ([16]). *s -DEFECTIVE CLIQUE EDITING is fixed-parameter tractable with respect to the combined parameter (s, k) .*

4 Further Techniques

Iterative Compression and Cluster Vertex Deletion. CLUSTER VERTEX DELETION is the vertex-deletion version of CLUSTER EDITING. The only difference is the modification allowed. In CLUSTER VERTEX DELETION, the only allowed operation is the deletion of vertices. A solution for this problem is called a “cluster vertex deletion set” (CVD, for short). Hüffner et al. [18] showed that the fairly recently introduced *iterative compression* technique can be applied to this problem, resulting in an algorithm running in $O(2^k k^6 \log k + n^3)$ time. The general idea behind their iterative compression is as follows: Given a graph $G = (V, E)$ and $k \geq 0$, start with $V' = \emptyset$ and $X' = \emptyset$; clearly, X' is a CVD for $G[V']$. Iterating over all graph vertices, step by step add one vertex $v \notin V'$ from V to both V' and X . Then X is still a CVD set for $G[V']$, although possibly not a minimum one. One can, however, obtain a minimum one by applying the *compression routine*. It takes a graph G and a CVD X for G , and returns a minimum CVD for G . Therefore, it is a loop invariant that X is a minimum-size CVD for $G[V']$. Since eventually $V' = V$, one obtains an optimal solution for G once the algorithm returns X . Together with the following lemma, the correctness and overall running time of the iterative compression algorithm follow.

Theorem 6 ([18]). *The compression routine for CLUSTER VERTEX DELETION runs in $O(2^k \cdot m \sqrt{n} \log n)$ time.*

Average Parameterization and Consensus Clustering. At the first sight, CONSENSUS CLUSTERING is not a graph-modeled clustering problem. However, one can reformulate it as a weighted version of CLUSTER EDITING as stated in [8]. In CONSENSUS CLUSTERING, one is given a base set S and a set \mathcal{C} of partitions over S , and asks for a partition C of S minimizing $\sum_{C_i \in \mathcal{C}} d(C, C_i)$. The distance $d(C, C_i)$ between two partitions is defined as follows: we call two elements $a, b \in S$ *co-clustered* with respect to a partition C if a and b occur together in a subset of C and *anti-clustered* if a and b occur in different subsets of C . Define the *distance* $d(C_i, C_j)$ between two partitions C_i and C_j as the number of unordered pairs $\{a, b\}$ of elements from the base set S such that a and b are co-clustered in one of C_i and C_j and anti-clustered in the other. Recently, CONSENSUS CLUSTERING has been shown to be fixed-parameter tractable with respect to the *average distance between the input partitions* [4] by deriving two data reduction rules and showing the existence of a “pseudo-kernel”. By defining the average distance between the input partitions as $d := (\sum_{C_i, C_j \in \mathcal{C}} d(C_i, C_j)) / (n \cdot (n - 1))$, one can show the following:

Theorem 7 ([4]). *Each CONSENSUS CLUSTERING instance can be reduced in polynomial time to an equivalent instance with at most $9d + 4$ elements.*

5 Conclusion

As demonstrated by several experimental studies [6, 10], data reduction proves extremely useful for practically solving hard problems. Even complex and large graphs such as biological instances allow for exact solutions. Therefore, encountering an NP-hard graph clustering problem, one should always start with designing data reduction rules. Even a data reduction that has no provable performance guarantee may turn out to be very effective in practice. Concerning the search tree technique, notice that the running times given here and in the literature are based on worst-case analysis and, thus, often too pessimistic.

There still remain a lot of challenges concerning the design of fixed-parameter algorithms for graph clustering problems: Until now, experimental studies of fixed-parameter algorithms in the field of graph-modeled clustering concentrated on CLUSTER EDITING. There lack efficient implementations for other variants of H -CLUSTER EDITING. For example, for s -DEFECTIVE CLIQUE EDITING, no polynomial-size kernel is known. A more general version of CLUSTER EDITING considers graphs with *don't care*-edges, that is, edges with zero cost [3]. It is open whether this problem is fixed-parameter tractable [7]. In some practical applications, the dense clusters do not need to be disjoint. The graphs resulting by the edge modifications are not necessarily vertex-disjoint dense clusters; some vertices may be contained in more than one dense clusters. A very first step in this direction has been undertaken recently [12], but this field is widely open.

References

- [1] Abello, J., Resende, M.G.C., Sudarsky, S.: Massive quasi-clique detection. In: Rajsbbaum, S. (ed.) LATIN 2002. LNCS, vol. 2286, pp. 598–612. Springer, Heidelberg (2002)
- [2] Ailon, N., Charikar, M., Newman, A.: Aggregating inconsistent information: Ranking and clustering. J. ACM 55(5), Article No. 23 (2008)
- [3] Bansal, N., Blum, A., Chawla, S.: Correlation clustering. Machine Learning 56(1-3), 89–113 (2004)
- [4] Betzler, N., Guo, J., Komusiewicz, C., Niedermeier, R.: Average parameterization for computing medians (submitted) (2009)
- [5] Böcker, S., Briesemeister, S., Bui, Q.B.A., Truß, A.: Going weighted: Parameterized algorithms for cluster editing. In: Yang, B., Du, D.-Z., Wang, C.A. (eds.) COCOA 2008. LNCS, vol. 5165, pp. 1–12. Springer, Heidelberg (2008)
- [6] Böcker, S., Briesemeister, S., Klau, G.W.: Exact algorithms for cluster editing: Evaluation and experiments. In: McGeoch, C.C. (ed.) WEA 2008. LNCS, vol. 5038, pp. 289–302. Springer, Heidelberg (2008)
- [7] Bodlaender, H.L., Fellows, M.R., Heggernes, P., Mancini, F., Papadopoulos, C., Rosamond, F.A.: Clustering with partial information. In: Ochmański, E., Tyszkiewicz, J. (eds.) MFCS 2008. LNCS, vol. 5162, pp. 144–155. Springer, Heidelberg (2008)
- [8] Bonizzoni, P., Vedova, G.D., Dondi, R., Jiang, T.: On the approximation of correlation clustering and consensus clustering. J. Comput. Syst. Sci. 74(5), 671–696 (2008)

- [9] Chesler, E.J., Lu, L., Shou, S., Qu, Y., Gu, J., Wang, J., Hsu, H.C., Mountz, J.D., Baldwin, N.E., Langston, M.A., Threadgill, D.W., Manly, K.F., Williams, R.W.: Complex trait analysis of gene expression uncovers polygenic and pleiotropic networks that modulate nervous system function. *Nature Genetics* 37(3), 233–242 (2005)
- [10] Dehne, F.K.H.A., Langston, M.A., Luo, X., Pitre, S., Shaw, P., Zhang, Y.: The cluster editing problem: Implementations and experiments. In: Bodlaender, H.L., Langston, M.A. (eds.) *IWPEC 2006*. LNCS, vol. 4169, pp. 13–24. Springer, Heidelberg (2006)
- [11] Fellows, M.R., Langston, M.A., Rosamond, F.A., Shaw, P.: Efficient parameterized preprocessing for Cluster Editing. In: Csuhaj-Varjú, E., Ésik, Z. (eds.) *FCT 2007*. LNCS, vol. 4639, pp. 312–321. Springer, Heidelberg (2007)
- [12] Fellows, M.R., Guo, J., Komusiewicz, C., Niedermeier, R., Uhlmann, J.: Graph-based data clustering with overlaps (submitted) (2009)
- [13] Gramm, J., Guo, J., Hüffner, F., Niedermeier, R.: Graph-modeled data clustering: Exact algorithms for clique generation. *Theory Comput. Syst.* 38(4), 373–392 (2005)
- [14] Guo, J.: A more effective linear kernelization for Cluster Editing. *Theor. Comput. Sci.* 410(8-10), 718–726 (2009)
- [15] Guo, J., Niedermeier, R.: Invitation to data reduction and problem kernelization. *ACM SIGACT News* 38(1), 31–45 (2007)
- [16] Guo, J., Kanj, I.A., Komusiewicz, C., Uhlmann, J.: Editing graphs into dense clusters (submitted) (2009)
- [17] Hüffner, F., Niedermeier, R., Wernicke, S.: Techniques for practical fixed-parameter algorithms. *The Computer Journal* 51(1), 7–25 (2008)
- [18] Hüffner, F., Komusiewicz, C., Moser, H., Niedermeier, R.: Fixed-parameter algorithms for cluster vertex deletion. *Theory Comput. Syst.* (to appear, 2009)
- [19] Kawaji, H., Takenaka, Y., Matsuda, H.: Graph-based clustering for finding distant relationships in a large set of protein sequences. *Bioinformatics* 20(2), 243–252 (2004)
- [20] Niedermeier, R.: *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, Oxford (2006)
- [21] Seidman, S.B., Foster, B.L.: A graph-theoretic generalization of the clique concept. *Journal of Mathematical Sociology* 6, 139–154 (1978)
- [22] Seno, S., Teramoto, R., Takenaka, Y., Matsuda, H.: A method for clustering expression data based on graph structure. *Genome Informatics* 15(2), 151–160 (2004)
- [23] Shamir, R., Sharan, R., Tsur, D.: Cluster graph modification problems. *Discrete Appl. Math.* 144(1-2), 173–182 (2004)
- [24] Voy, B.H., Scharff, J.A., Perkins, A.D., Saxton, A.M., Borate, B., Chesler, E.J., Branstetter, L.K., Langston, M.A.: Extracting gene networks for low-dose radiation using graph theoretical algorithms. *PLoS Computational Biology* 2(7), e89 (2006)
- [25] Yu, H., Paccanaro, A., Trifonov, V., Gerstein, M.: Predicting interactions in protein networks by completing defective cliques. *Bioinformatics* 22(7), 823–829 (2006)

On Spanners of Geometric Graphs^{*}

Iyad A. Kanj^{**}

School of Computing, DePaul University, 243 S. Wabash Ave., Chicago, IL 60604
ikanj@cs.depaul.edu

Abstract. We consider the problem of computing spanners of Euclidean graphs embedded in the 2-dimensional Euclidean plane. We present an $O(n \lg n)$ time algorithm that computes a spanner of a Euclidean graph that is of bounded degree and plane, where n is the number of points in the graph. Both upper bounds on the degree and the stretch factor significantly improve the previous bounds. We extend this algorithm to compute a bounded-degree plane *lightweight* spanner of a Euclidean graph.

Our results rely on elegant structural and geometric results that we develop. Moreover, our results can be extended to Unit Disk graphs under the local distributed model of computation.

1 Introduction

Given a Euclidean graph (complete graph) on n points in the plane, we consider the problem of computing a spanner of the graph possessing certain useful properties. The spanner properties we are interested in are: bounded degree, planarity, and lightweight—by which we mean that the total weight of the spanner is at most a constant times that of a Euclidean minimum spanning tree (of the set of points), where the weight of an edge in the graph is the Euclidean distance between its endpoints.

The problem of constructing a bounded degree or lightweight plane geometric spanner has been extensively studied within computational geometry; for example, see [1,3,8,9,12,14,16,20], and the following book on spanners [18]. More recently, wireless network researchers have approached the problem as well. Spanners and lightweight spanners are fundamental to wireless systems because they represent topologies that can be used for efficient unicasting, multicasting, and/or broadcasting (see [3,5,12,13,15,17,19], to name a few). For these applications, spanners are typically required to be planar and have bounded degree: the planarity requirement is for efficient routing, while the bounded degree requirement is caused by interference issues and the physical limitations of wireless devices [3,5,12,13,15,19].

In this paper we study the problem of computing spanners and lightweight spanners of Euclidean graphs. We present state-of-the-art results on these problems that improve the previous work in several aspects. Our work reveals interesting structural results that are of independent interest. We summarize below

^{*} The results in this paper are obtained jointly with Ljubmoir Perković and G. Xia.

^{**} Supported in part by a DePaul University Competitive Research Grant.

the main results of the paper and how they compare to the relevant work in the literature.

We start with the problem of constructing geometric spanners of Euclidean graphs, a well studied problem (see, for example, the recent book [18] for a survey on geometric spanners and their applications in networks). Dobkin et al. [11] showed that the Delaunay graph is a plane geometric spanner of the Euclidean graph with stretch factor $(1 + \sqrt{5})\pi/2 \approx 5.08$. This ratio was improved by Keil et al [14] to $C_{del} = 2\pi/(3 \cos(\pi/6)) \approx 2.42$, which currently stands as the best upper bound on the stretch factor of the Delaunay graph. Many researchers believe, however, that the lower bound of $\pi/2$ shown in [7] is also an upper bound on the stretch factor of the Delaunay graph. While Delaunay graphs are good plane geometric spanners of Euclidean graphs, they may have unbounded degree. Other geometric (sparse) spanners were also proposed in the literature including the Yao graphs [20], the Θ -graphs [14], and many others (see [18]); however, most of these proposed spanners either do not guarantee planarity, or do not guarantee bounded degree.

Bose et al. [2,3] were the first to show how to extract a subgraph of the Delaunay graph that is a bounded-degree, plane geometric spanner of the Euclidean graph (with stretch factor bounded by ≈ 10.02 and degree bounded by 27). Very recently, Bose et. al [6] improved the earlier result in [2,3] and showed how to construct a subgraph of the Delaunay graph that is a geometric spanner of the Euclidean graph with stretch factor: $\max\{\pi/2, 1 + \pi \sin(\alpha/2)\} \cdot C_{del}$ when $\alpha < \pi/2$, and $(1 + 2\sqrt{3} + 3\pi/2 + \pi \sin(\pi/12)) \cdot C_{del}$ when $\pi/2 \leq \alpha \leq 2\pi/3$, and whose degree is bounded by $14 + 2\pi/\alpha$.

In this paper, we prove structural results about Delaunay graphs that allow us to develop a very simple linear-time algorithm that, given a Delaunay graph, constructs a subgraph of the Delaunay graph with stretch factor $1 + 2\pi(k \cos(\pi/k))^{-1}$ (with respect to the Delaunay graph) and degree at most k , for any integer parameter $k \geq 14$. This result immediately implies an $O(n \lg n)$ time algorithm for constructing a plane geometric spanner of a Euclidean graph with stretch factor of $(1 + 2\pi(k \cos(\pi/k))^{-1}) \cdot C_{del}$ and degree at most k , for any integer parameter $k \geq 14$ (n is the number of vertices in the graph). We also show that our spanner includes a Euclidean minimum spanning tree as a subgraph.

This result significantly improves the previous results (described above) in terms of the stretch factor and the degree bound. To show this, we compare our results with previous results in more details. For a degree bound $k = 14$, our result implies a bound of at most 3.54 on the stretch factor. As the degree bound k approaches ∞ , our bound on the stretch factor approaches $C_{del} \approx 2.42$. The very recent results of Bose et al. [6] achieve a lowest degree bound of 17, and that corresponds to a bound on the stretch factor of at least 23. If Bose et al. [6] allow the degree bound to be arbitrarily large (i.e., to approach ∞), their bound on the stretch factor approaches $(\pi/2) \cdot C_{del} > 3.75$.

Regarding the problem of constructing lightweight spanners, Levcopoulos and Lingas [16] developed the first algorithm for this problem on Euclidean graphs. Their $O(n \log n)$ time algorithm, given a rational $\lambda > 2$, produces a plane spanner

with stretch factor $(\lambda - 1) \cdot C_{del}$ and total weight $(1 + \frac{2}{\lambda-2}) \cdot wt(\text{EMST})$, where the constant $C_{del} \approx 2.42$ is the stretch factor of the Delaunay subgraph of the Euclidean graph. Althöfer et al. [1] gave a polynomial time greedy algorithm that constructs a lightweight plane spanner of a Euclidean graph having the same upper bound on the stretch factor and weight as the algorithm by Levcopoulos and Lingas [16]. The degree of the lightweight spanner in both [16] and [1], however, may be unbounded: it is not possible to bound the degree without worsening the stretch factor or the weight. A more recent $O(n \log n)$ time algorithm by Bose, Gudmundsson, and Smid [3], succeeded in bounding the degree of the plane spanner by 27 but at a large cost: the stretch factor of the obtained plane spanner is approximately 10.02, and its weight is $O(wt(\text{EMST}))$, where the hidden constant in the asymptotic notation is undetermined. We note that any algorithm for constructing a Euclidean bounded-degree plane spanner, including the previously mentioned algorithms in [3,4,6], can be converted into an algorithm for constructing a Euclidean bounded-degree plane spanner that is lightweight, using the algorithm described in [12]. However, the multiplicative constant in the upper bound of the weight of the spanner with respect to the weight of the EMST will be undetermined.

Our contribution with regard to this problem is in designing an algorithm that, for any integer constant $\Delta \geq 14$ and constant $\lambda > 2$, constructs a plane spanner of a Euclidean graph having degree at most Δ , stretch factor $(\lambda - 1) \cdot (1 + 2\pi(\Delta \cos \frac{\pi}{\Delta})^{-1}) \cdot C_{del}$, and weight at most $(1 + \frac{2}{\lambda-2}) \cdot wt(\text{EMST})$ (Theorem 4.1). We can compare our algorithm with the algorithm by Bose, Gudmundsson, and Smid [3] if we let $\Delta = 14$ and $\lambda \approx 2.475$ in Theorem 4.1: we obtain an $O(n \log n)$ time algorithm that, given a Euclidean graph on n points, computes a plane spanner of the graph having degree at most 14, stretch factor at most 5.22, and weight at most $5.22 \cdot wt(\text{EMST})$.

Most of the proofs have been omitted for lack of space.

2 Definitions and Background

Given a set of points \mathcal{P} in the 2-dimensional Euclidean plane, the Euclidean graph \mathcal{E} on \mathcal{P} is defined to be the complete graph whose point-set is \mathcal{P} . Each edge AB connecting points A and B is assumed to be embedded in the plane as the straight line segment AB ; we define its *weight* to be the Euclidean distance $|AB|$.

For a subgraph $H \subseteq \mathcal{E}$, we denote by $V(H)$ and $E(H)$ the set of points and the set of edges of H , respectively, and by $wt(H)$ the sum of the weights of all the edges in H , that is, $wt(H) = \sum_{XY \in E(H)} wt(XY)$. The *length* of a path P (resp. cycle C) in a subgraph $H \subseteq \mathcal{E}$, denoted $|P|$ (resp. $|C|$), is the number of edges in P (resp. C).

Let G be a subgraph of \mathcal{E} . The cost of a simple path $A = M_0, M_1, \dots, M_r = B$ in G is $\sum_{j=0}^{r-1} |M_j M_{j+1}|$. Among all paths between A and B in G , a path with the smallest cost is defined to be a *smallest cost path* and we denote its cost as $c_G(A, B)$. A spanning subgraph H of G is said to be a *geometric spanner* of G if there is a constant ρ such that for every two points $A, B \in G$ we have:

$c_H(A, B) \leq \rho \cdot c_G(A, B)$. The constant ρ is called the *stretch factor* of H (with respect to the underlying graph G).

A spanning subgraph of \mathcal{E} is said to have *low weight*, or to be *lightweight*, if its weight is at most $c \cdot wt(T)$ for some constant c , where T is a Euclidean minimum spanning tree (EMST) of $V(\mathcal{E})$.

For three non-collinear points X, Y, Z in the plane we denote by $\bigcirc XYZ$ the circumscribed circle of $\triangle XYZ$. A *Delaunay triangulation* of a set of points \mathcal{P} in the plane is a triangulation of \mathcal{P} in which the circumscribed circle of every triangle contains no point of \mathcal{P} in its interior [10]. It is well known that if the points in \mathcal{P} are *in general position* (i.e., no four points in \mathcal{P} are cocircular) then the Delaunay triangulation of \mathcal{P} is unique [10]. In this paper—as in most papers in the literature—we shall assume that the points in \mathcal{P} are in general position; otherwise, the input can be slightly perturbed so that this condition is satisfied. The *Delaunay graph* of \mathcal{P} is defined as the plane graph whose point-set is \mathcal{P} and whose edges are the edges of the Delaunay triangulation of \mathcal{P} . An alternative equivalent definition that we end up using is:

Definition 2.1. ([10]) An edge XY is in the Delaunay graph of \mathcal{P} if and only if there exists a circle through points X and Y whose interior contains no point in \mathcal{P} .

It is well known that the Delaunay graph of a set of points \mathcal{P} is a spanning subgraph of the Euclidean graph defined on \mathcal{P} whose stretch factor is bounded by $C_{del} = 4\sqrt{3}\pi/9 \approx 2.42$ [14].

Given integer parameter $k > 6$, the *Yao subgraph* [20] of a plane graph G is constructed by performing the following *Yao step* at every point M of G : place k equally-separated rays out of M (arbitrarily defined), thus creating k closed cones of size $2\pi/k$ each, and choose the shortest edge in G out of M (if any) in each cone. The Yao subgraph consists of edges in G chosen by *either* endpoint. Note that the degree of a point in the Yao subgraph of G may be unbounded.

Two edges MX, MY incident on a point M in a graph G are said to be *consecutive* if one of the angular sectors determined by MX and MY contains no neighbors of M .

Let X and Y be two points in the plane and let (O) be any circle passing through X and Y . The chord XY subtends two regions of (O) . If Z is a point in the plane different from X and Y , then one of the two regions of (O) subtended by the chord XY is on the same side of XY as Z , whereas the other is on the opposite side of XY as Z . For convenience, we will refer to the former as the region of (O) subtended by XY and *closer* to Z , and to the latter as the region of (O) subtended by XY and *farther* or *away* from Z .

3 Computing Spanners of Delaunay and Euclidean Graphs

Let \mathcal{P} be a set of points in the plane and let \mathcal{E} be the complete, Euclidean graph defined on point-set \mathcal{P} . Let G be the Delaunay graph of \mathcal{P} . We have the following theorem:

Theorem 3.1. *For every integer $k \geq 14$, there exists a subgraph G' of G such that G' has maximum degree k and stretch factor $1 + 2\pi(k \cos \frac{\pi}{k})^{-1}$.*

A linear time algorithm that computes G' from G is the key component to the proof of the above theorem. This very simple algorithm essentially performs a *modified Yao step* (see Section 2) and selects up to k edges out of every point of G . G' is simply the spanning subgraph of G consisting of edges chosen by *both* endpoints.

In order to describe the modified Yao step, we must first develop a better understanding of the structure of the Delaunay graph G . Let CA and CB be edges incident on point C in G such that $\angle BCA \leq 2\pi/k$ and CA is the shortest edge within the angular sector $\angle BCA$. The above theorem easily follows if, for every such pair of edges CA and CB :

1. we show that there exists a path P from A to B in G such that:
 $|CA| + wt(P) \leq (1 + 2\pi(k \cos \frac{\pi}{k})^{-1})|CB|$, and
2. we modify the standard Yao step to include the edges of this path in G' , in addition to including the edges picked by the standard Yao step, and without choosing more than k edges at any point.

This will ensure that: for any edge $CB \in G$ that is not included in G' by the modified Yao step, there exists a path from C to B in G' , whose edges are all included in G' by the modified Yao step, and whose cost is at most $(1 + 2\pi(k \cos \frac{\pi}{k})^{-1})|CB|$. The lemma below, proves the existence of this path and shows some properties satisfied by edges of this path. We will then modify the standard Yao step to include edges satisfying these properties. We describe how the path in Lemma 3.1 can be constructed, and omit the proof about the properties of the path for lack of space.

Lemma 3.1. *Let $k \geq 14$ be an integer, and let CA and CB be edges in G such that $\angle BCA \leq 2\pi/k$ and CA is the shortest edge in the angular sector $\angle BCA$. There exists a path $P : (A = M_0, M_1, \dots, M_r = B)$ in G such that:*

- (i) $|CA| + \sum_{i=0}^{r-1} |M_i M_{i+1}| \leq (1 + 2\pi(k \cos \frac{\pi}{k})^{-1})|CB|$.
- (ii) *There is no edge in G between any pair M_i and M_j lying in the closed region delimited by CA , CB and the edges of P , for any i and j satisfying $0 \leq i < j - 1 \leq r$.*
- (iii) $\angle M_{i-1} M_i M_{i+1} > (\frac{k-2}{k})\pi$, for $i = 1, \dots, r - 1$.
- (iv) $\angle CAM_1 \geq \frac{\pi}{2} - \frac{\pi}{k}$.

We break down the construction of the path into two separate cases: when $\triangle ABC$ contains no point of G in its interior, and when it does. We define some additional notation and terminology first. We will denote by O the center of $\circ ABC$, and by θ the measure of $\angle BCA$. Note that $\angle AOB = 2\theta \leq 4\pi/k$.

We will use \widehat{AB} to denote the arc of $\circ ABC$ determined by points A and B and facing $\angle AOB$. We will make use of the following easily verified property:

Proposition 3.1. *If there are two circles through C and A and through C and B , respectively, that do not contain any points of G in their interior, then the*

region inside $\odot ABC$ subtended by chord CA and away from B and the region inside $\odot ABC$ subtended by chord CB and away from A contain no points of G in their interior.

3.1 The Outward Path

We consider first the case when no points of G are inside $\triangle ABC$. Since CA and CB are edges in G , by Definition 2.1 and Proposition 3.1, it follows that the closed region $\odot ABC$ subtended by chord AB and containing C is devoid of points of G . Keil and Gutwin [14] showed that, in this case, there exists a path between A and B in G inside the region of $\odot ABC$ subtended by chord AB away from C whose length is bounded by the length of \widehat{AB} (see Lemma 1 in [14]); we call this path the *outward path* between A and B .

For the case when no point of G lies inside $\triangle ABC$, we define the path in Lemma 3.1 to be the outward path between A and B . It can be proved that the outward path enjoys the properties described in Lemma 3.1.

3.2 The Inward Path

We consider now the case when the interior of $\triangle ABC$ contains points of G . Recall that CA and CB are edges of G such that CA is the shortest edge in the angular sector $\angle BCA$, and such that $\angle BCA \leq 2\pi/k$.

Let S be the set of points consisting of points A and B plus all the points interior to $\triangle ABC$ (note that $C \notin S$). Let $CH(S)$ be the set of points on the convex hull of S . Then $CH(S)$ consists of points $N_0 = A$ and $N_s = B$, and points N_1, \dots, N_{s-1} of G interior to $\triangle ABC$.

Proposition 3.2. *The following are true:*

- (a) For every $i = 0, \dots, s-1$, the interior of $\triangle CN_i N_{i+1}$ is devoid of point of G .
- (b) For every $i = 0, \dots, s$, there exists a circle passing through CN_i whose interior is devoid of points of G .

From Proposition 3.2, for every pair of points (N_i, N_{i+1}) , $i = 0, \dots, s-1$, the outward path P_i between points N_i and N_{i+1} is well defined. Let $A = M_0, M_1, \dots, M_r = B$ be the concatenation of the paths P_i , for $i = 0, \dots, s-1$. We call the path $A = M_0, M_1, \dots, M_r = B$ constructed above the *inward path* between A and B .

We can prove that the inward path between A and B satisfies the properties in Lemma 3.1.

3.3 The Modified Yao Step

We now augment the *Yao step* so edges forming the paths described in Lemma 3.1 are included in G' , in addition to the edges chosen in the standard Yao step. Lemma 3.1 says that consecutive edges on such paths form moderately large

Algorithm Modified Yao step

INPUT: A Delaunay graph G ; integer $k \geq 14$

OUTPUT: A subgraph G' of G of maximum degree k

1. define k disjoint cones of size $2\pi/k$ around every point M in G ;
2. in every non-empty cone, select the shortest edge incident on M in this cone;
3. **for** every maximal sequence of $\ell \geq 1$ consecutive empty cones:
 - 3.1. **if** $\ell > 1$ **then** select the first $\lfloor \ell/2 \rfloor$ unselected incident edges on M clockwise from the sequence of empty cones and the first $\lceil \ell/2 \rceil$ unselected edges incident on M counterclockwise from the sequence of empty cones;
 - 3.2. **else** (i.e., $\ell = 1$) let MX and MY be the incident edges on M clockwise and counterclockwise, respectively, from the empty cone; **if** either MX or MY is selected **then** select the other edge (in case it has not been selected); **otherwise** select the shorter edge between MX and MY breaking ties arbitrarily;
4. G' is the spanning subgraph of G consisting of edges selected by both endpoints.

Fig. 1. The modified Yao step

angles. The modified Yao step will ensure that consecutive edges forming large angles are included in G' . The algorithm is described in Figure 1.

Since the algorithm selects at most k edges incident on any point M and since only edges chosen by both endpoints are included in G' , each point has degree at most k in G' .

We argue that the running time of the above algorithm is linear. Note first that all edges incident on point M of degree Δ can be mapped to the k cones around M in linear time in Δ . Then, the shortest edge in every cone can be found in time $O(\Delta)$ (step 2 in the algorithm). Since k is a constant, selecting the $\ell/2$ edges clockwise (or counterclockwise) from a sequence of $\ell < k$ empty cones around M (step 3.1) can be done in $O(\Delta)$ time. Noting that the total number of edges in G is linear in the number of vertices completes the analysis.

To complete the proof of Theorem 3.1, we need the following lemma:

Lemma 3.2. *If edge CB is not selected by the algorithm, let CA be the shortest edge in the cone out of C to which CB belongs. Then the edges of the path described in Lemma 3.1 are included in G' by the algorithm.*

Corollary 3.1. *A Euclidean minimum spanning tree (EMST) on \mathcal{P} is a subgraph of G' .*

Proof. It is well known that a Delaunay graph (G) contains a EMST of its point-set [10]. If an edge CB is not in G' , then, by Lemma 3.2, a path from C to B

is included in G' . All edges on this path are no longer than CB , so there is a EMST not including CB .

Since a Delaunay graph of a Euclidean graph of n points can be computed in time $O(n \lg n)$ [10] and has stretch factor $C_{del} \approx 2.42$, we have the following theorem:

Theorem 3.2. *There exists an algorithm that, given a set \mathcal{P} of n points in the plane, computes a plane geometric spanner of the Euclidean graph on \mathcal{P} that contains a EMST, has maximum degree k , and has stretch factor $(1 + 2\pi(k \cos \frac{\pi}{k})^{-1}) \cdot C_{del}$, where $k \geq 14$ is an integer. Moreover, the algorithm runs in time $O(n \lg n)$.*

4 Computing Lightweight Spanners

In this section we present an algorithm that constructs a bounded-degree plane lightweight spanner of \mathcal{E} . We first need the following structural results.

Let G be a plane graph and let T be a spanning tree of G . Call an edge $e \in E(T)$ a *tree edge* and an edge $e \in E(G) - T$ a *non-tree edge*. Every non-tree edge induces a unique cycle in the graph $T + e$ called the *fundamental cycle* of e . Since T is embedded in the plane, we can talk about the *fundamental region* of e , which is the closed region in the plane enclosed by the fundamental cycle of e (other than the outer face of $T + e$).

Definition 4.1. Define a relationship \preceq on the set $E(G)$ as follows. For every edge e , $e \preceq e$. For two edges e and e' in $E(G)$, $e \preceq e'$ if and only if e is contained in the fundamental region of e' .

It is not difficult to verify that \preceq is a partial order relation on $E(G)$, and hence $(E(G), \preceq)$ is a partially ordered set (POSET). Note that any two distinct tree edges are not comparable by \preceq , and that every tree edge is a minimal element in $(E(G), \preceq)$. Therefore, we can topologically sort the edges in $E(G)$ to form a list $\mathcal{L} = \langle e_1, \dots, e_r \rangle$, in which no non-tree edge appears before a tree edge, and such that if $e_i \preceq e_j$ then e_i does not appear after e_j in \mathcal{L} .

Lemma 4.1. *Let e_i be a non-tree edge. Then there exists a unique face F_i in G such that every edge e_j of F_i satisfies $e_j \preceq e_i$.*

We will call the unique face associated with a non-tree edge e_i , described in Lemma 4.1, the *fundamental face* of e_i . The following corollary can be proved using the same techniques used in the proof of Theorem 2 in [1].

Corollary 4.1. *Let G be a connected weighted plane graph with nonnegative weights, and let T be a spanning tree in G . Let $\lambda > 2$ be a constant. Suppose that for every edge $e \in E(G) - T$ we have $wt(F_e) \geq \lambda \cdot wt(e)$, where F_e is the boundary cycle of the fundamental face of e in G . Then $wt(G) \leq (1 + \frac{2}{\lambda-2}) \cdot wt(T)$.*

Now using the $O(n \log n)$ time algorithm described in Section 3, given a Euclidean graph \mathcal{E} on a set of n points in the plane, and an integer parameter

$\Delta \geq 14$, we can construct a plane spanner G' of \mathcal{E} containing a EMST of $V(\mathcal{E})$, of degree at most Δ , and of stretch factor $\rho = (1 + 2\pi(\Delta \cos \frac{\pi}{\Delta})^{-1}) \cdot C_{del}$, where $C_{del} \approx 2.42$ is the stretch factor of the Delaunay subgraph of \mathcal{E} .

The spanner G' , however, may not be of light weight. Therefore, we need to discard edges from G' so that the resulting subgraph is of light weight, while at the same time not affecting the stretch factor of G' by much. To do so, since G' is a plane graph containing a EMST of $V(G')$, we would like to employ Corollary 4.1. However, there is one technical problem: the fundamental faces of G' may not satisfy the condition of Corollary 4.1, namely that the weight of every fundamental face F_e of a non-EMST edge e in G' satisfies $wt(F_e) \geq \lambda \cdot wt(e)$ ($\lambda > 2$ is a constant). We will show next how to prune the set of edges in G' so that this condition is satisfied.

Let T be a EMST of $V(G')$ contained in G' . As described above, we can order the non-tree edges in G' with respect to the partial order \preceq described in Definition 4.1. Let $\mathcal{L}' = \langle e_1, e_2, \dots, e_s \rangle$ be the sequence of non-tree edges in G' sorted in a non-decreasing order with respect to the partial order \preceq . Note that, by the definition of the partial order \preceq , if we add the edges in \mathcal{L}' to T in the respective order they appear in \mathcal{L}' , once an edge e_i is added to form a fundamental face in the partially-grown graph, this fundamental face will remain a face in the resulting graph after all the edges in \mathcal{L}' have been added to T . That is, the face will not be affected (i.e., changed/split) by the addition of any later edge in this sequence.

Given a constant $\lambda > 2$, to construct the desired lightweight spanner G , we first initialize G to the EMST T . We consider the non-tree edges of G' in the order that they appear in \mathcal{L}' . Inductively, suppose that we have processed the edges e_1, \dots, e_{i-1} in \mathcal{L}' . To process edge e_i , let F_i be the fundamental face of e_i in $G + e_i$. If $wt(F_i) > \lambda \cdot wt(e_i)$, we add e_i to G ; otherwise, e_i is not added to G . Let G be the resulting graph at the end of the construction.

Lemma 4.2. *Given the set of n points $V(\mathcal{E})$ in the plane, the graph G can be constructed in $O(n \log n)$ time.*

Theorem 4.1. *For any integer parameter $\Delta \geq 14$ and any constant $\lambda > 2$, the subgraph G of \mathcal{E} constructed above is a plane spanner of \mathcal{E} containing a EMST of $V(\mathcal{E})$, whose degree is at most Δ , whose stretch factor is $(\lambda - 1) \cdot \rho$, where $\rho = (1 + 2\pi(\Delta \cos \frac{\pi}{\Delta})^{-1}) \cdot C_{del}$, and whose weight is at most $(1 + \frac{2}{\lambda - 2}) \cdot wt(EMST)$. Moreover, G can be constructed in $O(n \log n)$ time.*

References

1. Althöfer, I., Das, G., Dobkin, D., Joseph, D., Soares, J.: On sparse spanners of weighted graphs. *Discrete & Computational Geometry* 9, 81–100 (1993)
2. Bose, P., Gudmundsson, J., Smid, M.: Constructing plane spanners of bounded degree and low weight. In: Möhring, R.H., Raman, R. (eds.) *ESA 2002*. LNCS, vol. 2461, pp. 234–246. Springer, Heidelberg (2002)
3. Bose, P., Gudmundsson, J., Smid, M.: Constructing plane spanners of bounded degree and low weight. *Algorithmica* 42(3-4), 249–264 (2005)

4. Bose, P., Morin, P.: Online routing in triangulations. *SIAM J. Comput.* 33(4), 937–951 (2004)
5. Bose, P., Morin, P., Stojmenovic, I., Urrutia, J.: Routing with guaranteed delivery in ad hoc wireless networks. *Wireless Networks* 7(6), 609–616 (2001)
6. Bose, P., Smid, M., Xu, D.: Diamond triangulations contain spanners of bounded degree. In: Asano, T. (ed.) *ISAAC 2006*. LNCS, vol. 4288, pp. 173–182. Springer, Heidelberg (2006)
7. Chew, P.: There are planar graphs almost as good as the complete graph. *Journal of Computers and System Sciences* 39(2), 205–219 (1989)
8. Das, G., Heffernan, P., Narasimhan, G.: Optimally sparse spanners in 3-dimensional euclidean space. In: *Proceedings of SoCG*, pp. 53–62 (1993)
9. Das, G., Narasimhan, G.: A fast algorithm for constructing sparse euclidean spanners. In: *Proceedings of SoCG*, pp. 132–139 (1994)
10. de Berg, M., van Kreveld, M., Overmars, M., Schwarzkopf, O.: *Computational Geometry: Algorithms and Applications*, 2nd edn. Springer, Heidelberg (2000)
11. Dobkin, D., Friedman, S., Supowit, K.: Delaunay graphs are almost as good as complete graphs. *Discrete Computational Geometry* 5(4), 399–407 (1990)
12. Gudmundsson, J., Levkopoulos, C., Narasimhan, G.: Fast greedy algorithms for constructing sparse geometric spanners. *SIAM J. Comput.* 31(5), 1479–1500 (2002)
13. Kanj, I., Perković, L.: On geometric spanners of euclidean and unit disk graphs. In: *Proceedings of STACS* (2008)
14. Keil, J., Gutwin, C.: Classes of graphs which approximate the complete Euclidean graph. *Discrete & Computational Geometry* 7, 13–28 (1992)
15. Kranakis, E., Singh, H., Urrutia, J.: Compass routing on geometric networks. In: *Proceeding of CCCG*, pp. 51–54 (1999)
16. Levkopoulos, C., Lingas, A.: There are planar graphs almost as good as the complete graphs and almost as cheap as minimum spanning trees. *Algorithmica* 8(3), 251–256 (1992)
17. Li, X.-Y., Calinescu, G., Wan, P.-J., Wang, Y.: Localized delaunay triangulation with application in Ad Hoc wireless networks. *IEEE Trans. on Parallel and Dist. Systems.* 14(10), 1035–1047 (2003)
18. Narasimhan, G., Smid, M.: *Geometric Spanner Networks*. Cambridge University Press, Cambridge (2007)
19. Wang, Y., Li, X.-Y.: Localized construction of bounded degree and planar spanner for wireless ad hoc networks. *Mobile Networks and Applications* 11(2), 161–175 (2006)
20. Yao, A.C.-C.: On constructing minimum spanning trees in k-dimensional spaces and related problems. *SIAM Journal on Computing* 11(4), 721–736 (1982)

Searching Trees: An Essay

Henning Fernau and Daniel Raible

Univ.Trier, FB 4—Abteilung Informatik, 54286 Trier, Germany
{fernau,raible}@uni-trier.de

Abstract. We are reviewing recent advances in the run time analysis of search tree algorithms, including indications to open problems. In doing so, we also try to cover the historical dimensions of this topic.

1 Introduction

Search trees are a basic tool for solving combinatorial problems. The underlying idea is quite simple: decompose a given problem into finer and finer partial problems (applying a suitable branching operation) such that the generated partial problems together solve the original problem. Hence, they have been investigated from various points of views for about 50 years, so nearly through all the history of computer science and its mathematics.

Despite of its long history, there are (to our knowledge) only few attempts to develop a unifying view on this topic, apart from some quite old papers, rarely quoted these days, as discussed below. This essay can be seen as a quest to resume a generic research on search trees, with the specific aim to bring the sometimes very successful applications of search trees in practice closer to the theoretical (worst case) analysis. We believe there are many yet unsolved questions.

2 Historical Notes on Search Trees

Conceptual frameworks for search trees. A nice framework on search trees has been developed by Ibaraki in 1978 [30]; actually, that paper sums up many early works. That paper presented a framework that allows to justify the correctness of branch-and-bound procedures. To this end, combinatorial optimization problems are described in the form of *discrete decision processes* (ddp). This formalization actually goes back to Karp and Held [33]. Branch-and-bound, from its very name, implies (in the case of minimization problems) the existence of a lower-bound test. In fact, this was in the focus of earlier formalizations of the principle as, e.g., the one by Mitten [40] that also sums up and reviews still earlier results from the sixties. In those days, many seemingly different names were in use for search tree algorithms; for example, the *backtrack programming* framework of Golomb and Baumert [22] gives another formalization of search tree algorithms. This allows to prune the search tree at node n , assuming it is known that better solutions are already known, compared to the best possible solution that might be derived upon further branching starting out from n . Good pruning hence necessitates

good estimates on the values that could be obtained from n onwards (without necessarily expanding n), which is a (practical and mathematical) problem on its own right. Besides this test, Ibaraki considers two more types of tests (that are quite related among themselves again): *dominance tests* and *equivalence tests*. Roughly speaking, due to a dominance test (considered in details in [29]), we can prune branches in the search tree, since we are sure that better solutions can be found in other branches. Equivalence tests might prove two nodes to be equivalent in the sense that they yield solutions of the same quality, so that only one of the nodes need to be expanded; which one could be the matter of choice due to other criteria. The mentioned early papers focus on the important issue of correctness of the described methods (and how to ensure correctness in concrete circumstances). Typical conceptual questions include: What are the logical relations between abstract properties of discrete decision processes? or: What kind of properties of dominance relations are needed to ensure correctness of the implied pruning strategy?

Artificial Intelligence. A non-negligible part of the literature on Artificial Intelligence deals with search spaces since the very early days. In this sense, search trees play a prominent role in that area, as well. In particular, this remark is true when considering game strategies (like how to play chess with a computer). There again, many (mostly heuristic) decisions have to be made to find a successful path in the underlying implicit configuration tree. We only mention that there are quite a lot of nice internet resources available, like <http://www.cs.ualberta.ca/~aixplore/>. Also in that area of expertise, search trees in the more technical sense of this essay have been examined. For example, Reiter's *theory of diagnosis* is based upon so-called Hitting Set Trees, see [45].

Specific communities have worked a lot to optimize and analyze their search tree algorithms. Nice tools (and generalizable methodologies) have been developed there. For example, the search tree analysis of Kullmann [36,37] within the Satisfiability Community has been a blueprint of similar approaches elsewhere; it could be seen as one of the fathers of the measure-and-conquer paradigm discussed below. Within the integer linear programming (ILP), and more general, the mathematical programming community, branch-and-bound has been very successfully combined with cutting (hyper-)planes and similar approaches, leading to the so-called branch-and-cut paradigm.¹ Branch-and-cut (and in particular, further refinements like branch-and-cut-price, see [32]) has been a very successful paradigm in industrial practice for solving ILPs, up to the point that solving these (NP-hard) problems is deemed to be practically feasible. However, rather simple examples exist that show that a non-educated use of this approach will lead to quite bad running times, see [31]. The arguably best modern book on this topic [1] focuses on the Traveling Salesman Problem, which is a sort of standard testbed for this approach, starting out from the very origins of branch-and-cut [26]. Another community on its own is dealing with binary decision diagrams [8]; we are not aware of any specific search tree analysis in that area.

¹ We gratefully acknowledge discussions on ILP techniques with Frauke Liers.

3 Estimating Running Times

Most of the theory papers on search trees up to the seventies focus on conceptual properties of search tree algorithms. Only few papers (to our knowledge), e.g., [27,28,34], try to attack the efficiency question of search tree algorithms from an abstract point of view. Interestingly, one paper [28] shows that even in the average case, exponential growth of branch-and-bound algorithms are not avoidable, while Knuth [34] shows that Monte-Carlo-methods can be used to assess search tree sizes. It is not so clear how later research in the Artificial Intelligence Community on backtracking programs fits into this line of research, see [38].

Run-time estimates of exponential-time algorithms (as being typical for search tree algorithms) have only relatively recently found renewed interest, obviously initiated by the steadily growing interest in parameterized complexity theory and parameterized (and exact exponential-time) algorithms. Yet, the basic knowledge in this area is not very new. This is possibly best exemplified by the textbook of Mehlhorn [39]. There, to our knowledge, the very first parameterized algorithm for the vertex cover problem was given, together with its analysis, much predating the advent of parameterized algorithmics. This algorithm is quite simple: if any edge $e = \{v_1, v_2\}$ remains in the graph, produce two branches in the search tree, one putting v_1 into the (partial) cover and the other one putting v_2 into the cover. In both cases, the parameter k upperbounding the cover size is decremented, and we consider $G - v_i$ instead of G in the recursive calls. Given a graph G together with an upperbound k on the cover size, such a search tree algorithm produces a binary search tree of height at most k ; so in the worst case, its size (the number of leaves) is at most 2^k . Namely, if $T(k)$ denotes the number of leaves in a search tree of height k , the described recursion implies $T(k) \leq 2T(k - 1)$, which (together with the anchor $T(0) = 1$) yields $T(k) \leq 2^k$.

This reasoning generalizes when we obtain recurrences of the form:

$$T(k) \leq \alpha_1 T(k - 1) + \alpha_2 T(k - 2) + \dots + \alpha_\ell T(k - \ell) \tag{1}$$

for the size $T(k)$ of the search tree (which can be measured in terms of the number of leaves of the search tree, since that number basically determines the running time of a search tree based algorithm).

More specifically, α_i is a natural number that indicates that in α_i of the $\sum_j \alpha_j$ overall branches of the algorithm, the parameter value k got decreased by i . Notice that, whenever $\ell = 1$, it is quite easy to find an estimate for $T(k)$, namely α_1^k . A recipe for the more general case is contained in Alg. [1]. Why does that algorithm work correctly? Please observe that in the simplest case (when $\ell = 1$), the algorithm does what could be expected. We only mention here that

$$p(x) = x^\ell - \alpha_1 x^{\ell-1} - \dots - \alpha_\ell x^0$$

is also sometimes called the *characteristic polynomial* of the recurrence given by Eq. [1] and the base c of the exponential function that Alg. [1] returns is called the *branching number* of this recurrence. Due to the structure of the characteristic polynomial, c is the dominant positive real root.

Algorithm 1. Simple time analysis for search tree algorithms, called ST-simple

Require: a list $\alpha_1, \dots, \alpha_\ell$ of nonnegative integers, the coefficients of inequality (1)

Ensure: a tight estimate c^k upperbounding $T(k)$

Consider inequality (1) as equation:

$$T(k) = \alpha_1 T(k-1) + \alpha_2 T(k-2) + \dots + \alpha_\ell T(k-\ell)$$

Replace $T(k-j)$ by x^{k-j} , where x is still an unknown to be determined.

Divide the equation by $x^{k-\ell}$.

{This leaves a polynomial $p(x)$ of degree ℓ .}

Determine the largest positive real zero (i.e., root) c of $p(x)$.

return c^k .

Alternatively, such a recursion can be also written in the form

$$T(k) \leq T(k-a_1) + T(k-a_2) + \dots + T(k-a_r). \tag{2}$$

Then, (a_1, \dots, a_r) is also called the *branching vector* of the recurrence.

As detailed in [23, pp. 326ff.], a general solution of an equation

$$T(k) = \alpha_1 T(k-1) + \alpha_2 T(k-2) + \dots + \alpha_\ell T(k-\ell)$$

(with suitable initial conditions) takes the form

$$T(k) = f_1(k)\rho_1^k + \dots + f_\ell(k)\rho_\ell^k,$$

where the ρ_i are the distinct roots of the characteristic polynomial of that recurrence, and the f_i are polynomials (whose degree corresponds to the degree of the roots (minus one)). As regards asymptotics, we can conclude $T(k) \in \mathcal{O}^*(\rho_1^k)$, where ρ_1 is the dominant root.

The exact mathematical reasons can be found in the theory of polynomial roots, as detailed in [23, 13, 24, 36, 37]. It is of course also possible to check the validity of the approach by showing that $T(k) \leq \rho^k$ for the obtained solution ρ by a simple mathematical induction argument.

Due to case distinctions that will play a key role for designing refined search tree algorithms, the recurrences often take the form

$$T(k) \leq \max\{f_1(k), \dots, f_r(k)\},$$

where each of the $f_i(k)$ is of the form

$$f_i(k) = \alpha_{i,1}T(k-1) + \alpha_{i,2}T(k-2) + \dots + \alpha_{i,\ell}T(k-\ell).$$

Such a recurrence can be solved by r invocations of Alg. 1, each time solving $T(k) \leq f_i(k)$. This way, we get r upperbounds $T(k) \leq c_i^k$. Choosing $c = \max\{c_1, \dots, c_r\}$ is then a suitable upper bound.

Eq. (1) somehow suggests that the entities a_j that are subtracted from k in the terms $T(k-a_j)$ in Eq. (2) are natural numbers. However, this need not be the

case, even in the case that the branching process itself suggests this, e.g., taking vertices into the cover to be constructed. How do such situations arise? Possibly, during the branching process we produce situations that appear to be more favorable than the current situation. Hence, we could argue that we take a certain credit on this future situation, this way balancing the current (bad) situation with the future (better) one. Interestingly, this approach immediately leads to another optimization problem: How to choose the mentioned credits to get a good estimate on the search tree size? We will describe this issue in more detail below in a separate section. This sort of generalization is the backbone of the search tree analysis in so-called exact exponential algorithms, where the aim is, say in graph algorithms, to develop non-trivial algorithms for hard combinatorial graph problems with run-times estimated in terms of n (number of vertices) or sometimes m (number of edges). One typical scenario where this approach works is a situation where the problem allows for nice branches as long as large-degree vertices are contained in the graph, as well as for nice branches if all vertices have small degree, assuming that branching recursively generates new instances with degrees smaller than before, see [16,17,44,46]

An alternative generalization is the following one: We arrive at systems of equations. In its most general form, these will again include maximum operators that can be treated as explained above. We are left with solving systems like $T(k, \ell) \leq \max_{i=1}^r f_i(k, \ell)$, where each of the $f_i(k, \ell)$ ($1 \leq i, \ell \leq r$) is of the form

$$f_i(k, \ell) = \alpha_{i,\ell,1}T(k-1, i) + \alpha_{i,\ell,2}T(k-2, i) + \dots + \alpha_{i,\ell,q_\ell}T(k-q_\ell, i).$$

This approach was successfully applied to problems related to HITTING SET, see [9,10,11]. In the case of 3-HITTING SET, the *auxiliary parameter* ℓ counts how many hyperedges of small size have been generated, since branching on them is favorable. We have seen in our examples that an upper bound to the general situation, i.e., a bound on $T(k) = T(k, 0)$, takes again the form c^k . Moreover, the other entities $T(k, \ell)$ are upperbounded by $\beta_\ell c^k$ for suitable $\beta_\ell \in (0, 1)$. Now, if we replace $T(j, \ell)$ by $\beta_\ell T(j)$ in the derived inequality system, we are (after dividing inequalities with left-hand side $T(k, \ell)$ by β_ℓ) back to our standard form discussed above. All inequalities involved now take the form: $T(k) \leq \sum_{j \geq 0} \gamma_j T(k-j)$. Notice that $j = 0$ is feasible as long as $\gamma_0 < 1$. Namely, assuming again an upperbound c^k on $T(k)$, the term $\gamma_0 T(k)$ on the right-hand side can be re-interpreted as $T(k + \log_c(\gamma_0))$. Due to $\gamma_0 < 1$, the logarithmic term is negative, so that the parameter is actually reduced. In fact, the system of inequalities of the form

$$T(k) \leq T(k + \log_c(\gamma_0)) + \sum_{j \geq 1} \gamma_j T(k-j)$$

would yield the same solution. Terms like $\gamma T(k)$ on the right-hand side can be interpreted as a case not yielding direct improvement / reduction of the parameter budget, but the knowledge that the overall search tree size is shrunk by factor γ . Another interpretation is possible via the multivariate analysis of Eppstein [6].

Algorithm 2. A simple algorithm for DOMINATING SET

- 1: **if** possible choose a $v \in \text{BLND}$ such that $|N(v) \cap (\text{BLND} \cup \text{INND})| \geq 2$. **then**
 - 2: Binary branch on v (i.e., set v active in one branch, inactive in the other)
 - 3: **else if** possible choose a $v \in \text{BLDO}$ such that $|N(v) \cap (\text{BLND} \cup \text{INND})| \geq 3$. **then**
 - 4: Binary branch on v .
 - 5: **else**
 - 6: Solve the remaining instance in polynomial time using an EDGE COVER algorithm.
-

4 Measure-and-Conquer

In this separate section, we will discuss one of the most successful approaches to run-time estimation of search trees developed in recent years. With this technique it was possible to prove run time upperbounds $\mathcal{O}^*(c^n)$ with $c < 2$ for several hard vertex selection problems. Among these problems (where for years nothing better than the trivial 2^n -algorithm was known) are many variants of DOMINATING SET [16] like CONNECTED or POWER DOMINATING SET [17,44] and FEEDBACK VERTEX SET [14]. The methodology resulted in simplifying algorithms (INDEPENDENT SET [18]) and in speeding up existent non-trivial ones (DOMINATING SET [16,46], INDEPENDENT DOMINATING SET [21] and MAX-2-SAT [43]). This approach also served for algorithmically proving upper bounds on the number of minimal dominating [19] and feedback vertex sets [14].

In this approach, the complexity of an algorithm is not analyzed with respect to $n = |V|$ (or $m = |E|$) for a graph instance $G = (V, E)$. Rather, one chooses a tailored measure, call it μ , which should reflect the progress of the algorithm. Nevertheless, in the end we desire an upperbound of the form c^n . Hence, we must assure that there is some constant ℓ such that $\mu \leq \ell n$ during the whole algorithm. Then, a proven upperbound c^μ entails the desired upperbound $c^{\ell n}$.

A simple example. We give an algorithm for DOMINATING SET using less than 2^n steps, see Alg. 2. It branches on vertices by deciding whether they should be in the solution (*active*) or not (*inactive*). Regarding this we call them *active* and *inactive*. A vertex for which this decision has not been made is called *blank*. If a vertex is active, then its neighbors are *dominated*. Let $\text{BLND} = \{v \in V \mid v \text{ is blank \& not dominated}\}$, $\text{INND} = \{v \in V \mid v \text{ is inactive \& not dominated}\}$ and $\text{BLDO} = \{v \in V \mid v \text{ is blank \& dominated}\}$. We now define our measure:

$$\mu = |\text{BLND}| + \omega \cdot (|\text{INND}| + |\text{BLDO}|) \leq n$$

In step 6, we create an EDGE COVER instance $G_{EC} = (V(E_{EC}), E_{EC})$: (1) For all $v \in \text{BLND}$ with $N(v) \cap (\text{BLND} \cup \text{INND}) = \{q\}$, adjoin $e = \{v, q\}$ to E_{EC} and let $\alpha(e) = v$; for all $v \in \text{BLDO}$ with $N(v) \cap (\text{BLND} \cup \text{INND}) = \{x, y\}$, $x \neq y$, put $e = \{x, y\}$ into E_{EC} and (re-)define $\alpha(e) = v$. If C is a minimum edge cover of G_{EC} , set all v active where $v = \alpha(e)$ for some $e \in C$. (2) Set all $v \in \text{BLND} \setminus V(E_{EC})$ with $N(v) \cap (\text{BLND} \cup \text{INND}) = \emptyset$ active. (3) Set all $v \in \text{BLDO}$ such that $|N(v) \cap (\text{BLND} \cup \text{INND})| = 0$ inactive. (4) Set all $v \in \text{BLDO}$ such that $N(v) \cap (\text{BLND} \cup \text{INND}) = \{s\} \not\subseteq V(E_{EC})$ active.

Now we come to the analysis of the branching process in steps 2 and 4. Let $n_{bl} = |N(v) \cap \text{BLND}|$ and $n_{in} = |N(v) \cap \text{INND}|$. If we set v to active in **step 2**, we first reduce μ by one as v vanishes from μ . Then all vertices in $N(v) \cap \text{BLND}$ will be dominated and hence moved to the set BLDO. Thus, μ is reduced by an amount of $n_{bl} \cdot (1 - \omega)$. In the same way the vertices in $N(v) \cap \text{INND}$ are dominated. Therefore, these do not appear in μ anymore. Hence, μ is lowered by $n_{in}\omega$. If we set v inactive we reduce μ by $(1 - \omega)$, as v is moved from BLND to INND. Hence, the branching vector is:

$$(1 + n_{bl}(1 - \omega) + n_{in}\omega, (1 - \omega)) \tag{3}$$

where $n_{bl} + n_{in} \geq 2$ due to step 1. In **step 4**, we have chosen $v \in \text{BLDO}$ for branching. Here, we must consider that in the first and second branch we only get ω as reduction from v (v disappears from μ). But the analysis with respect to $N(v) \cap (\text{BLND} \cup \text{INND})$ remains valid. Thus,

$$(\omega + n_{bl}(1 - \omega) + n_{in}\omega, \omega) \tag{4}$$

is the branching vector with respect to $n_{bl} + n_{in} \geq 3$ due to step 3.

Unfortunately, depending on n_{bl} and n_{in} we have infinite number of branching vectors. But it is only necessary to consider the worst case branches. For (3) these are the ones with $n_{bl} + n_{in} = 2$ and for (4) $n_{bl} + n_{in} = 3$. For any other branching vector, we can find one among those giving a worse upperbound. Thus, we have a finite set of recurrences $R_1(\omega), \dots, R_7(\omega)$ depending on ω . The next task is to choose ω in a way such that the maximum root of the evolving characteristic polynomials is minimum. In this case we easily see that $\omega := 0.5$. Then the worst case branching vector for (3) and (4) is $(2, 0.5)$. Thus, the number of leaves of the search tree evolving from this branching vector can be bounded by $\mathcal{O}^*(1.9052^\mu)$. Thus, our algorithm breaks the 2^n -barrier using a simple measure. So, one might argue that we should use a more elaborated measure to get a better upperbound:

$$\mu' = |\text{BLND}| + \omega_1 \cdot (|\text{INND}|) + \omega_2 \cdot (|\text{BLDO}|)$$

Under μ' , (3) becomes $(1 + n_{bl}(1 - \omega_1) + n_{in}\omega_2, (1 - \omega_2))$; (4) becomes $(\omega_1 + n_{bl}(1 - \omega_1) + n_{in}\omega_2, \omega_1)$. The right choice for the weights turns into a tedious task. In fact, if $\omega_1 = 0.637$ and $\omega_2 = 0.363$, then we get an upperbound $\mathcal{O}^*(1.8899^{\mu'})$. Nevertheless, the current best upperbound is $\mathcal{O}(1.5134^n)$, see [1646].

Generally, one wants the measure to reflect the progress made by the algorithm best possible. This leads to more and more complicated measures with lots of weights to be chosen. So at a certain point, this task can not be done by hand and is an optimization problem of its own which can only be reasonable solved with the help of a computer. One way of obtaining good weights is to use local search. Starting from initial weights, we examine the direct neighborhood to see if we can find an weight assignment which provides a better upperbound. In practice, this approach works quite well, especially if we use compiled programs. Then an amount of hundreds of characteristic polynomials and several weights can be handled. There is also a formulation as a convex program [20]. For this problem class there are efficient solvers available. An alternative is the approach of Eppstein [6].

5 Correctness of Search Tree Algorithms

In its most primitive form (making complete case distinction at each branch), proving correctness of a search tree algorithm is not a real issue. However, this does become an issue when less trivial branching rules are involved (designed to improve on the running time of the algorithms). This was already noticed in the early days: as said above, correctness was the key issue dealt with in early theory papers on branch-and-bound algorithms. So the question was (and is again): How can we design correct pruning rules that make the search tree shrink as far as possible, without losing the ability to find an optimum solution?

As long as the trivial search tree algorithm is only modified to incorporate certain heuristic priorities with respect to which the branching is performed, no problem with respect to correctness incurs. Such priorities are mere implementations of the inherent nondeterministic selection of a branching item. To further speed up the search, the instance might be modified using reduction rules. As long as these reduction rules are valid for any instance, this is no problem either; however, sometimes such rules are only valid in combination with branching (and a further interplay with the heuristic priorities is possible).

Finally, one might think about transferring the already mentioned ideas about dominance and equivalence of search tree nodes (and the corresponding instances) into the search tree analysis. Here, things become tricky. Recall that upon running a branch-and-bound algorithm, we have dynamically changing information about hitherto best or most promising (partial) solutions at hand. Moreover, we have a certain direction of time in which the underlying static search tree is processed. Both properties give (additional) possibilities to prune search tree nodes, e.g., by not expanding nodes where no improvement over the best solution found so far is to be expected. Nonetheless, it is tempting to stop branching at certain nodes n when it has become clear that other branches would lead to solutions no worse than the ones expected from n . Instead, we would (conceptually) introduce a reference link from n to other nodes in the search tree. However, we must avoid creating cycles in the graph consisting of the search tree together with the conceptual links (viewed as arcs in the graph). An according model named *reference search tree* was defined (and successfully employed) in [44].

6 List of Questions and Research Topics

Is it possible to somehow **use the basic principle of branch-and-bound**, namely the use of a bounding function, **within the run-time analysis of search tree algorithms**? Possibly, the worst case approach is not suitable here. However, are there any possibilities for a reasonable **average-case analysis**? The main mathematical problem and challenge is that, even if we start out with a random instance (in whatever sense), the graph will not be any longer “random” after the first branching operations, since it will have been modified according to some heuristic strategy. So, combinatorial and statistical properties will be

destroyed by branching. However, any progress in this direction would be highly welcome, since the difference between theoretical worst-case analysis (making many search tree approaches seemingly prohibitive) and the actual (fast) run time in practice is far too large and can be best explained by the facts that (1) many heuristics work very nice in practice and (2) a good branching strategy enables to bound the search quite efficiently at a very early stage.

How can **search trees techniques be applied in combination with other algorithmic paradigms**? This idea has been quite successful in connection with dynamic programming (DP). With DP (when used for solving hard combinatorial problems), often the main problem is the exponential space requirement. In general form, these ideas seem to go back to Morin and Marsten [41], exemplified with the Traveling Salesman Problem (TSP). The simple idea is to trade time and space, more specifically, to use search trees to lower the prohibitive space requirements of DP. More recently, such space saving ideas have been employed to obtain good (not necessarily best) solutions [2], as well as (again) for TSP in [3]. From the early days of research onwards, the connection between search tree algorithms and DP was seen; e.g., in [33]. Karp and Held investigate a kind of reverse question: which discrete decision processes can be solved by DP? One can think to combine search tree algorithms with other paradigms, as, e.g., iterative compression / expansion as known from parameterized algorithms, see [42] for a textbook explanation. However, to the knowledge of the authors, no such research has been carried out yet.

Explore the **measure-and-conquer paradigm within parameterized algorithms**. Only few examples of applying measure-and-conquer to (classical) parameterization are known, see [12] for one such example. However, in those examples, there is always a very close link between the non-parameterized view (say, measured in terms of the number m of edges) and the parameterization (e.g., the number of edges k found in an acyclic subgraph). The finite automata approach of Wahlström [47], as well as the ideas expressed in Sec. 3 that offer the re-interpretation of weights as search-tree reduction might show a way to a broader application of measure-and-conquer to (classical) parameterization. Here, Eppstein's quasiconvex method [6] could be also of interest. Other forms of amortized analysis (using potential functions that are analyzed on each path of the search tree) are reported in [4] and could also find broader applicability.

For specific forms of recurrent (e.g., divide-and-conquer) algorithms, connections between the **run time estimation and fractal geometry** have been shown [5]. How does this setting generalize towards time estimates of search-tree algorithms as obtained by branch-and-bound? Notice that those fractal geometric objects in turn are quite related to finite automata, and there are many connections between finite automata and search trees, see [30,47].

Is there a broader theory behind the idea of **automization of search tree analysis**? Several attempts in the direction of employing computers to do the often nasty and error-prone case-analysis have been reported [7,25,35]. Can such ideas be combined with the measure-and-conquer approach that, in itself, already needs a certain computer assistance?

References

1. Applegate, D.L., Bixby, R.E., Chvátal, V., Cook, W.J.: *The Traveling Salesman Problem: A Computational Study*. Princeton Univ. Press, Princeton (2006)
2. Bailey, M., Alden, J., Smith, R.L.: Approximate dynamic programming using epsilon-pruning (working paper). TR, University of Michigan, Ann Arbor (2002)
3. Björklund, A., Husfeldt, T., Kaski, P., Koivisto, M.: The Travelling Salesman Problem in bounded degree graphs. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfssdóttir, A., Walukiewicz, I. (eds.) *ICALP 2008, Part I*. LNCS, vol. 5125, pp. 198–209. Springer, Heidelberg (2008)
4. Chen, J., Kanj, I.A., Xia, G.: Labeled search trees and amortized analysis: improved upper bounds for NP-hard problems. In: Ibaraki, T., Katoh, N., Ono, H. (eds.) *ISAAC 2003*. LNCS, vol. 2906, pp. 148–157. Springer, Heidelberg (2003)
5. Dube, S.: Using fractal geometry for solving divide-and-conquer recurrences (extended abstract). In: Ng, K.W., Balasubramanian, N.V., Raghavan, P., Chin, F.Y.L. (eds.) *ISAAC 1993*. LNCS, vol. 762, pp. 191–200. Springer, Heidelberg (1993)
6. Eppstein, D.: Quasiconvex analysis of multivariate recurrence equations for back tracking algorithms. *ACM Trans. Algorithms* 2, 492–509 (2006)
7. Fedin, S.S., Kulikov, A.S.: Automated proofs of upper bounds on the running time of splitting algorithms. In: Downey, R.G., Fellows, M.R., Dehne, F. (eds.) *IWPEC 2004*. LNCS, vol. 3162, pp. 248–259. Springer, Heidelberg (2004)
8. Feigenbaum, J., Kannan, S., Vardi, M.Y., Viswanathan, M.: The complexity of problems on graphs represented as OBDDs. *Chicago J. Theor. Comput. Sci.* (1999)
9. Fernau, H.: Two-layer planarization: improving on parameterized algorithmics. *J. Graph Algorithms and Applications* 9, 205–238 (2005)
10. Fernau, H.: Parameterized algorithms for HITTING SET: the weighted case. In: Calamoneri, T., Finocchi, I., Italiano, G.F. (eds.) *CIAC 2006*. LNCS, vol. 3998, pp. 332–343. Springer, Heidelberg (2006)
11. Fernau, H.: A top-down approach to search trees: improved algorithmics for 3-HITTING SET. *Algorithmica* (to appear)
12. Fernau, H., Raible, D.: Exact algorithms for maximum acyclic subgraph on a superclass of cubic graphs. In: Nakano, S.-i., Rahman, M. S. (eds.) *WALCOM 2008*. LNCS, vol. 4921, pp. 144–156. Springer, Heidelberg (2008)
13. Flajolet, P., Sedgewick, R.: *Analytic Combinatorics*. Cambridge Univ. Press, Cambridge (2008)
14. Fomin, F.V., Gaspers, S., Pyatkin, A.V., Razgon, I.: On the Minimum Feedback Vertex Set Problem: Exact and Enumeration Algorithms. *Algorithmica* 52(2), 293–307 (2008)
15. Fomin, F., Golovach, P., Kratsch, D., Kratochvíl, J., Liedloff, M.: Branch & recharge: Exact algorithms for generalized domination. In: Dehne, F., Sack, J.-R., Zeh, N. (eds.) *WADS 2007*. LNCS, vol. 4619, pp. 507–518. Springer, Heidelberg (2007)
16. Fomin, F.V., Grandoni, F., Kratsch, D.: Measure and conquer: domination – a case study. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) *ICALP 2005*. LNCS, vol. 3580, pp. 191–203. Springer, Heidelberg (2005)
17. Fomin, F.V., Grandoni, F., Kratsch, D.: Solving connected dominating set faster than 2^n . In: Arun-Kumar, S., Garg, N. (eds.) *FSTTCS 2006*. LNCS, vol. 4337, pp. 152–163. Springer, Heidelberg (2006)

18. Fomin, F.V., Grandoni, F., Kratsch, D.: Measure and conquer: a simple $O(2^{0.288n})$ independent set algorithm. In: SODA, pp. 18–25 (2006)
19. Fomin, F.V., Grandoni, F., Pyatkin, A.V., Stepanov, A.A.: Combinatorial bounds via measure and conquer: Bounding minimal dominating sets and applications. *ACM Trans. Algorithms* 5, 1–17 (2008)
20. Gaspers, S.: Exponential Time Algorithms: Structures, Measures, and Bounds. Ph.D thesis, University of Bergen, Norway (2008)
21. Gaspers, S., Liedloff, M.: A Branch-and-Reduce Algorithm for Finding a Minimum Independent Dominating Set in Graphs. In: Fomin, F.V. (ed.) WG 2006. LNCS, vol. 4271, pp. 78–89. Springer, Heidelberg (2006)
22. Golomb, S.W., Baumert, L.D.: Backtrack programming. *J. ACM* 12, 516–524 (1965)
23. Graham, R., Knuth, D.E., Patashnik, O.: Concrete Mathematics. Addison-Wesley, Reading (1989)
24. Gramm, J.: Fixed-Parameter Algorithms for the Consensus Analysis of Genomic Data. Dissertation, Univ. Tübingen, Germany (2003)
25. Gramm, J., Guo, J., Hüffner, F., Niedermeier, R.: Automated generation of search tree algorithms for hard graph modification problems. *Algorithmica* 39, 321–347 (2004)
26. Hong, S.: A Linear Programming Approach for the Traveling Salesman Problem. Ph.D thesis, The Johns Hopkins University, Baltimore, Maryland, USA (1972)
27. Ibaraki, T.: Theoretical comparison of search strategies in branch-and-bound algorithms. *Intern. J. Computer and Information Sciences* 5, 315–344 (1976)
28. Ibaraki, T.: On the computational efficiency of branch-and-bound algorithms. *J. Operations Research Society of Japan* 26, 16–35 (1977)
29. Ibaraki, T.: The power of dominance relations in branch-and-bound algorithms. *J. ACM* 24, 264–279 (1977)
30. Ibaraki, T.: Branch-and-bound procedure and state-space representation of combinatorial optimization problems. *Information and Control* 36, 1–27 (1978)
31. Jeroslaw, R.G.: Trivial integer programs unsolvable by branch-and-bound. *Mathematical Programming* 6, 105–109 (1974)
32. Jünger, M., Thienel, S.: The ABACUS system for branch-and-cut-and-price algorithms in integer programming and combinatorial optimization. *Software: Practice and Experience* 30, 1325–1352 (2000)
33. Karp, R.M., Held, M.: Finite-state processes and dynamic programming. *SIAM J. Applied Mathematics* 15, 693–718 (1967)
34. Knuth, D.E.: Estimating the efficiency of backtrack programs. *Mathematics of Computation* 29, 121–136 (1975)
35. Kulikov, A.S.: Automated generation of simplification rules for SAT and MAXSAT. In: Bacchus, F., Walsh, T. (eds.) SAT 2005. LNCS, vol. 3569, pp. 430–436. Springer, Heidelberg (2005)
36. Kullmann, O.: New methods for 3-SAT decision and worst-case analysis. *Theoretical Computer Science* 223, 1–72 (1999)
37. Kullmann, O.: Fundamentals of Branching Heuristics. In: Handbook of Satisfiability, ch. 7, pp. 205–244. IOS Press, Amsterdam (2009)
38. McDiarmid, C.J.H., Provan, G.M.A.: An expected-cost analysis of backtracking and non-backtracking algorithms. In: IJCAI 1991, vol. 1, pp. 172–177 (1991)
39. Mehlhorn, K.: Graph Algorithms and NP-Completeness. Springer, Heidelberg (1984)
40. Mitten, L.G.: Branch-and-bound methods: general formulation and properties. *Operations Research* 18, 24–34 (1970)

41. Morin, T.L., Marsten, R.E.: Branch-and-bound strategies for dynamic programming. *Operations Research* 24, 611–627 (1976)
42. Niedermeier, R.: *Invitation to Fixed-Parameter Algorithms*. Oxford Univ. Press, Oxford (2006)
43. Raible, D., Fernau, H.: A new upper bound for MAX-2-SAT: A graph-theoretic approach. In: Ochmański, E., Tyszkiewicz, J. (eds.) *MFCS 2008*. LNCS, vol. 5162, pp. 551–562. Springer, Heidelberg (2008)
44. Raible, D., Fernau, H.: Power domination in $O^*(1.7548^n)$ using reference search trees. In: *ISAAC*. LNCS, vol. 5369, pp. 136–147. Springer, Heidelberg (2008)
45. Reiter, R.: A theory of diagnosis from first principles. *Artif. Intell.* 32, 57–95 (1987)
46. van Rooij, J.M.M., Bodlaender, H.L.: Design by measure and conquer, a faster exact algorithm for dominating set. In: *STACS*, pp. 657–668. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl (2008)
47. Wahlström, M.: *Algorithms, Measures and Upper Bounds for Satisfiability and Related Problems*. Ph.D thesis, Linköpings universitet, Sweden (2007)

Approximability and Fixed-Parameter Tractability for the Exemplar Genomic Distance Problems*

Binhai Zhu

Department of Computer Science
Montana State University
Bozeman, MT 59717-3880 USA
bhzh@cs.montana.edu

Abstract. In this paper, we present a survey of the approximability and fixed-parameter tractability results for some Exemplar Genomic Distance problems. We mainly focus on three problems: the exemplar breakpoint distance problem and its complement (i.e., the exemplar non-breaking similarity or the exemplar adjacency number problem), and the maximal strip recovery (MSR) problem. The following results hold for the simplest case between only two genomes (genomic maps) \mathcal{G} and \mathcal{H} , each containing only one sequence of genes (gene markers), possibly with repetitions.

1. For the general Exemplar Breakpoint Distance problem, it was shown that deciding if the optimal solution value of some given instance is zero is NP-hard. This implies that the problem does not admit any approximation, neither any FPT algorithm, unless $P=NP$. In fact, this result holds even when a gene appears in \mathcal{G} (\mathcal{H}) at most two times.
2. For the Exemplar Non-breaking Similarity problem, it was shown that the problem is linearly reducible from Independent Set. Hence, it does not admit any factor- $O(n^\epsilon)$ approximation unless $P=NP$ and it is $W[1]$ -complete (loosely speaking, there is no way to obtain an $O(n^{o(k)})$ time exact algorithm unless $FPT=W[1]$, here k is the optimal solution value of the problem).
3. For the MSR problem, after quite a lot of struggle, we recently showed that the problem is NP-complete. On the other hand, the problem was previously known to have a factor-4 approximation and we showed recently that it admits a simple FPT algorithm which runs in $O(2^{2.73k}n + n^2)$ time, where k is the optimal solution value of the problem.

1 Introduction

In bioinformatics and computational biology, we constantly need to process various biological data to extract meaningful biological relation, like building a

* This research is partially supported by NSF, NSERC, Louisiana Board of Regents under contract number LEQSF(2004-07)-RD-A-35, and MSU-Bozeman's Short-Term Professional Development Leave Program.

phylogenetic tree. However, such a process usually involves solving hard combinatorial optimization problems which are typically NP-complete.

In the area of bioinformatics and computational biology, one would typically apply three methods to handle these NP-complete problems. One is to find an approximation solution, with the requirement being that the approximation factor is small (better close to one). The other is to look for an exact solution (FPT algorithm) when the solution size of the problem is small. The vast majority of practical solutions for bioinformatics and computational biology are heuristic ones, which are possibly based on some formal methods like integer linear programming, branch-and-bound, etc.

In this paper, we review the approximability and fixed-parameter tractability results for three problems related to exemplar genomic distance computation. In these problems, we are given some genomes or genomic maps and we try to optimize some solutions values by deleting some genes or gene markers. So these problem fit naturally for approximation and/or FPT solutions. Unfortunately, as we will review a bit later, some of these problems are very hard in both aspects. In other words, it might be impossible to design good approximation and/or FPT algorithms for them, unless $P=NP$ or $FPT=W[1]$. On the other hand, many problems are still open along these lines.

The paper is organized as follows. In Section 2, we review the approximability and fixed-parameter tractability for the Exemplar Breakpoint Distance (EBD) problem. In Section 3, we review the approximability and fixed-parameter tractability for the Exemplar Non-breaking Similarity (ENbS) problem (which is the dual of EBD). In Section 4, we review the approximability and fixed-parameter tractability for the Maximal Strip Recovery (MSR) problem. In Section 5, we list a set of open problems to conclude this paper.

2 Approximability and Fixed-Parameter Tractability for EBD

In the genome comparison and rearrangement area, a standard problem is to compute the number (i.e., genetic distances) and the actual sequence of genetic operations needed to convert a source genome to a target genome. This problem is important in evolutionary molecular biology. Typical genetic distances include edit [23], signed reversal [26,24,6] and breakpoint [30], etc. (The idea of signed reversal and, implicitly, breakpoint, was initiated as early as in 1936 by Sturtevant and Dobzhansky [29].) In the past years, conserved interval distance was also proposed to measure the similarity of multiple sequences of genes [9]. Interested readers are referred to [21,22] for a summary of the research performed in this area.

However, in genome rearrangement research, it is almost always assumed that each gene appears in a genome exactly once. Under this assumption, the genome rearrangement problem is in essence the problem of comparing and sorting signed permutations [21,22]. However, this assumption is very restrictive and is only justified in several small virus genomes. For example, this assumption does not

hold on eukaryotic genomes where paralogous genes exist [25,27]. On the one hand, it is important in practice to compute genomic distances, e.g., Hannenhalli and Pevzner's method [21], when no gene duplications arise; on the other hand, one might have to handle this gene duplication problem as well.

Sankoff first considered the problem of computing genomic distance with duplicated genes. About ten years ago, Sankoff proposed a way to select, from the duplicated copies of genes, the common ancestor gene such that the distance between the reduced genomes (*exemplar genomes*) is minimized [27]. A general branch-and-bound algorithm was also implemented in [27]. In [25], Nguyen, Tay and Zhang proposed to use a divide-and-conquer method to compute the exemplar breakpoint distance empirically.

For the theoretical part of research, it was shown that both of the problems of computing the signed reversal and breakpoint distances between exemplar genomes are NP-complete [7]. A few years ago, Blin and Rizzi further proved that computing the conserved interval distance between exemplar genomes is NP-complete [8]; moreover, it is NP-complete to compute the minimum conserved interval matching (i.e., without deleting the duplicated copies of genes). Recently we showed much stronger inapproximability result for the exemplar conserved interval distance problem (even under a weaker model of approximation) [12]. While various exemplar genomic distances have been researched before, in this survey we will focus on the exemplar breakpoint distance. In fact, all the inapproximability result for exemplar breakpoint distance under the normal model of approximation holds for any other genomic distance $d(-, -)$ satisfying $d(G, H) = 0$ implies $G = H$ or $G = -H$.

2.1 Preliminaries

In the genome comparison and rearrangement problem, we are given a set of genomes, each of which is a signed sequence of genes. (In general a genome could contain a set of such sequences. The genomes we focus on are typically called *singletons*.) The order of the genes corresponds to the position of them on the linear chromosome and the signs correspond to which of the two DNA strands the genes are located. While most of the past research are under the assumption that each gene occurs in a genome once, this assumption is problematic in reality for eukaryotic genomes or the likes where duplications of genes exist [27]. Sankoff proposed a method to select an *exemplar genome*, by deleting redundant copies of a gene, such that in an exemplar genome any gene appears exactly once; moreover, the resulting exemplar genomes should have a property that certain genetic distance between them is minimized [27].

The following definitions are very much following those in [8]. Given n gene families (alphabet) \mathcal{F} , a genome \mathcal{G} is a sequence of elements of \mathcal{F} such that each element is with a sign (+ or -). In general, we allow the repetition of a gene family in any genome. Each occurrence of a gene family is called a *gene*, though we will not try to distinguish a gene and a gene family if the context is clear. Given a genome $G = g_1g_2\dots g_m$ with no repetition of any gene, we say that gene g_i *immediately precedes* g_j if $j = i + 1$. Given genomes G, H , if gene

a immediately precedes b in G and neither a immediately precedes b nor $-b$ immediately precedes $-a$ in H , then they constitute a *breakpoint* in G . The *breakpoint distance* is the number of breakpoints in G (symmetrically, it is the number of breakpoints in H).

The number of a gene g appearing in a genome \mathcal{G} is called the cardinality of g in \mathcal{G} , written as $\text{card}(g, \mathcal{G})$. A gene in \mathcal{G} is called *trivial* if g has cardinality exactly 1; otherwise, it is called *non-trivial*. A genome \mathcal{G} is called *r-repetitive*, if all the genes from the same gene family appear at most r times in \mathcal{G} . For example, $\mathcal{G} = c - adc - bdeb$ is 2-repetitive.

Given a genome \mathcal{G} over \mathcal{F} , an *exemplar genome* of \mathcal{G} is a genome \mathcal{G}' obtained from \mathcal{G} by deleting duplicating genes such that each gene family in \mathcal{G} appears exactly once in \mathcal{G}' . For example, let $\mathcal{G} = -bcaadag - e$, there are two exemplar genomes: $-bcadg - e$ and $-bcdag - e$.

The Exemplar Breakpoint Distance (EBD) problem is defined as follows:

Instance: Genomes \mathcal{G} and \mathcal{H} , each is of length $O(m)$ and each covers n identical gene families (i.e., at least one gene from each of the n gene families appears in both \mathcal{G} and \mathcal{H}); integer K .

Question: Are there two respective exemplar genomes of \mathcal{G} and \mathcal{H} , G and H , such that the breakpoint distance between them is at most K ?

In the next subsection, we present some hardness results on the approximability and fixed-parameter tractability for EBD, namely, the hardness to compute or approximate the minimum value K in the above formulation. Given a minimization (maximization) problem Π , let the optimal solution value of Π be OPT. We say that an approximation algorithm \mathcal{A} provides a *performance guarantee* of α for Π if for every instance I of Π , the solution value returned by \mathcal{A} is at most $\alpha \times \text{OPT}$ (at least OPT/α). Usually we say that \mathcal{A} is a factor- α approximation for Π . For the obvious reason, we are only interested in polynomial time approximation algorithms. Readers are referred to [16,19] for more details regarding the definitions related to approximation algorithms and NP-completeness.

As a well-known subject as well, an FPT algorithm for an optimization problem Π with optimal solution value $\text{OPT} = k$ is an algorithm which solves the problem in $O(f(k)n^c)$ time, where f is any function only on k and c is some fixed constant not related to k . More details on FPT algorithms can be found in [18].

2.2 Hardness Results

In [10], we presented the first set of inapproximability and approximation results for the Exemplar Breakpoint Distance problem, given two genomes each containing only one sequence of genes drawn from n identical gene families. We showed that even if a gene appears at most three times, deciding whether the optimal exemplar breakpoint distance is zero, i.e, whether $G = H$, is NP-complete. It was left as an open problem whether the result holds when each gene appears at most twice in each of the input genomes [10,1]. This year, this open

question was finally answered, i.e., it remains NP-complete even when each gene appears at most two times [4]. Combining these results, we have the following inapproximability result.

Theorem 1. *If both \mathcal{G} and \mathcal{H} are 2-repetitive genomes, then the Exemplar Breakpoint Distance problem does not admit any polynomial time approximation (regardless of its approximation factor), unless $P=NP$.*

Proof. If we view the Exemplar Breakpoint Distance problem as a minimization problem, then the result in [4] implies that deciding whether $\text{OPT} = 0$ is NP-complete (even if the input genomes are 2-repetitive). Let \mathcal{A} be any approximation algorithm for EBD with factor α . By definition, \mathcal{A} returns an approximation solution value APP, with

$$\text{APP} \leq \alpha \times \text{OPT}.$$

When $\text{OPT} = 0$, clearly APP must also satisfy $\text{APP} = 0$. In other words, \mathcal{A} would be able to solve the instance in [4] in polynomial time. This, however, contradicts with the corresponding NP-completeness result (unless $P=NP$). \square

Regarding the fixed-parameter tractability for EBD, we have the following theorem.

Theorem 2. *If both \mathcal{G} and \mathcal{H} are 2-repetitive genomes, then the Exemplar Breakpoint Distance problem does not admit any FPT algorithm, unless $P=NP$.*

Proof. Again, if we view the Exemplar Breakpoint Distance problem as a minimization problem, then the result in [4] implies that deciding whether $\text{OPT} = 0$ is NP-complete (even if the input genomes are 2-repetitive). Let \mathcal{B} be any FPT algorithm for EBD which runs in $O(f(k)n^c)$ time. When $\text{OPT} = k = 0$, \mathcal{B} solves EBD in $O(f(0)n^c) = O(n^c)$ time. In other words, \mathcal{B} would be able to solve the instance in [4] in polynomial time. This, again, contradicts with the corresponding NP-completeness result, unless $P=NP$. \square

3 Approximability and Fixed-Parameter Tractability for ENbS

We comment that the negative results in Section 2.2 hold for any genomic distance $d(-, -)$ satisfying that $d(G, H) = 0$ implies $G = H$ or $G = -H$. This, of course, implies that all the exemplar genomic distance problems (like exemplar reversal, exemplar transposition, and exemplar conserved interval distances) do not admit any polynomial time approximation algorithms or any FPT algorithm, unless $P=NP$.

There have been two ways to handle this problem. One is to use a weak model of approximation, which will be covered as related to open problems in Section 5. The other, on the other hand, is to use a different similarity measure. In this case, one would try to maximize certain similarity measure. The most notably such measures include non-breaking similarity (or number of adjacencies) [13] and

the number of common intervals [3]. (A common interval is a pair of substrings appearing in the two genomes with the same genes, but possibly different orders. Example. $G = abced, H = deacb$. (abc, acb) is a length-3 common interval.) We will focus the non-breaking similarity, which is really the complement of the breakpoint distance.

For two exemplar genomes G and H over the same alphabet of size n , a breakpoint in G is a two-gene substring $g_i g_{i+1}$ such that neither $g_i g_{i+1}$ nor $-g_{i+1} - g_i$ is a substring in H . A *non-breaking point* (or an *adjacency*) is a common two-gene substring $g_i g_{i+1}$ that appears either as $g_i g_{i+1}$ or as $-g_{i+1} - g_i$ in G and H . The number of non-breaking points between G and H is also called the *non-breaking similarity* between G and H , denoted as $\text{nbs}(G, H)$. Clearly, we have $\text{nbs}(G, H) = n - 1 - \text{bd}(G, H)$. For two genomes \mathcal{G} and \mathcal{H} , their *exemplar non-breaking similarity* $\text{enbs}(\mathcal{G}, \mathcal{H})$ is the maximum $\text{nbs}(G, H)$, where G and H are exemplar genomes derived from \mathcal{G} and \mathcal{H} . Again we have $\text{enbs}(\mathcal{G}, \mathcal{H}) = n - 1 - \text{ebd}(\mathcal{G}, \mathcal{H})$.

The Exemplar Non-breaking Similarity (ENbS) problem is formally defined as follows:

Instance: Genomes \mathcal{G} and \mathcal{H} , each is of length $O(m)$ and each covers n identical gene families (i.e., at least one gene from each of the n gene families appears in both \mathcal{G} and \mathcal{H}); integer K .

Question: Are there two respective exemplar genomes of \mathcal{G} and \mathcal{H} , G and H , such that the non-breaking similarity between them is at least K ?

We have the following negative results which have been proved in [13].

Theorem 3. *If one of \mathcal{G} and \mathcal{H} is exemplar and the other is 2-repetitive, then the Exemplar Non-breaking Similarity problem does not admit any factor- n^ϵ polynomial time approximation unless $P=NP$.*

Proof. We give a sketch of proof from [13]. In [13], it was shown that Independent Set can be linearly reduced to ENbS; i.e., the input graph has an independent set of size k iff the constructed ENbS instance has a non-breaking similarity (or number of adjacencies) equal to k . As Independent Set cannot be approximated within a factor of n^ϵ unless $P=NP$ [20], the theorem follows. \square

Theorem 4. *If one of \mathcal{G} and \mathcal{H} is exemplar and the other is 2-repetitive, the Exemplar Non-breaking Similarity problem does not admit an FPT algorithm unless $FPT=W[1]$.*

Proof. It is noted that the reduction from Independent Set to ENbS in [13] is in fact an FPT reduction. As Independent Set is $W[1]$ -complete [18], the theorem simply follows. \square

In fact, with the lower bound results proved in [15], Independent Set (hence ENbS) cannot be solved in $O(f(k)n^{o(k)})$ time even if k is bounded by an arbitrarily small function of n , unless ETH fails. (ETH — Exponential Time Hypothesis: 3SAT cannot be solved in subexponential time.)

4 Approximability and Fixed-Parameter Tractability for MSR

Given two genomic maps G and H represented by a sequence of n gene markers, a *strip* (syntenic block) is a sequence of distinct markers of length at least two which appear as subsequences in both of the input maps, either directly or in reversed and negated form. The problem *Maximal Strip Recovery* (MSR) is to find two subsequences G' and H' of G and H , respectively, such that the total length of disjoint strips in G' and H' is maximized. An example is as follows: $G = abcde$, $H = cbdae$ and the optimal solution is $G' = H' = cde$.

The MSR problem was proposed to handle the elimination of noise and ambiguities in genomic maps. This is related to the well-known problem in comparative genomics — to decompose two given genomes into syntenic blocks, i.e., segments of chromosomes which are deemed to be homologous in the two input genomes. Two years ago, a heuristic method was proposed to handle the MSR problem [17,32]. In [14], a factor-4 polynomial time approximation algorithm was proposed for the problem. This was done by applying the Maximum Weight Independent Set on 2-interval graphs, which admit a factor-4 approximation [5]. We also proved that several close variants of MSR, MSR- d (with $d > 2$ input maps), MSR-DU (with marker duplications), and MSR-WT (with markers weighted) are all NP-complete. It was left as an open problem whether the problem can be solved in polynomial time or is NP-complete [14].

Recently, in [31] we showed that MSR is in fact NP-complete, via a polynomial time reduction from One-in-Three 3SAT (which was shown to be NP-complete in [28,19]). We summarize the results in [14,31] as follows.

Theorem 5. *MSR is NP-complete, and it admits a factor-4 polynomial time approximation.*

As an effort to solve the MSR problem practically, we tried to solve MSR and its variants exactly with FPT algorithms, i.e., showing that MSR is fixed-parameter tractable [31]. Let k be the minimum number of markers deleted in various versions of MSR, the running time of our algorithms are $O(2^{2.73k}n + n^2)$ for MSR, $O(2^{2.73k}dn + dn^2)$ for MSR- d , and $O(2^{5.46k}n + n^2)$ for MSR-DU respectively. We summarize this result in [31] as follows.

Theorem 6. *Let k be the optimal number of gene markers deleted from the input genomic maps. MSR can be solved in $O(2^{2.73k}n + n^2)$ time; i.e., MSR is fixed-parameter tractable.*

Note that as k is typically greater than 50 in real datasets, our FPT algorithms are not yet practical.

5 Concluding Remarks and Open Problems

The negative results on EBD and ENbS do not mean that we have absolutely no way to tackle these problems. For instance, in [2], with integer linear programming, very nice empirical results are obtained. Here, we try to present a different way to handle these problems formally.

In many biological problems, the optimal solution value OPT could be zero. (Besides EBD, in some minimum recombination haplotype reconstruction problems the optimal solution value could be zero.) As implied by Theorem 1, if computing such an optimal solution with zero solution value is NP-complete then the problem does not admit *any* polynomial time approximation (unless $P=NP$). However, in reality one would be satisfied to obtain a solution with value one or two. Due to this reason, we can relax the traditional definition of approximation to a *weak approximation*. Given a minimization problem Π , let the optimal solution of Π be OPT. We say that a weak approximation algorithm \mathcal{W} provides a *performance guarantee* of α for Π if for every instance I of Π , the solution value returned by \mathcal{W} is at most $\alpha \times (\text{OPT} + 1)$.

In [10,11,12] we showed that EBD and the exemplar conserved interval distance problems are both hard to approximate even under the weak approximation model. But for the exemplar reversal distance problem, no such result is known yet.

For the exemplar common interval number problem [3], the only negative result is its NP-hardness. It would also be interesting to know whether it admits an efficient polynomial time approximation. We conclude this paper with a list of open problems.

1. For the exemplar reversal distance problem, does there exist a good weak approximation?
2. For the exemplar common interval number problem, does there exist a good approximation?
3. For the MSR problem, does there exist a polynomial time approximation with factor better than 4?
4. For the MSR problem, does there exist a more efficient FPT algorithm?

Acknowledgments

I would like to thank my collaborators for this series of research: Zhixiang Chen, Richard Fowler, Bin Fu, Minghui Jiang, Lusheng Wang, Jinhui Xu, Boting Yang, and Zhiyu Zhao. Special thanks to Jianer Chen for answering many questions regarding FPT.

References

1. Angibaud, S., Fertin, G., Rusu, I.: On the approximability of comparing genomes with duplicates. In: Nakano, S.-i., Rahman, M. S. (eds.) WALCOM 2008. LNCS, vol. 4921, pp. 34–45. Springer, Heidelberg (2008)
2. Angibaud, S., Fertin, G., Rusu, I., Thévenin, A., Vialette, S.: Efficient tools for computing the number of breakpoints and the number of adjacencies between two genomes with duplicate genes. *J. Computational Biology* 15, 1093–1115 (2008)
3. Blin, G., Chauve, C., Fertin, G., Rizzi, R., Vialette, S.: Comparing genomes with duplicates: a computational complexity point of view. *IEEE/ACM Trans. on Computational Biology and Bioinformatics* 4, 523–534 (2007)

4. Blin, G., Fertin, G., Sikora, F., Vialette, S.: The exemplar breakpoint distance for non-trivial genomes cannot be approximated. In: Proc. 3rd Workshop on Algorithm and Computation, WALCOM 2009 (to appear, 2009)
5. Bar-Yehuda, R., Halldórsson, M.M., Naor, J.(S.), Shachnai, H., Shapira, I.: Scheduling split intervals. *SIAM Journal on Computing* 36, 1–15 (2006)
6. Bafna, V., Pevzner, P.: Sorting by reversals: Genome rearrangements in plant organelles and evolutionary history of X chromosome. *Mol. Bio. Evol.* 12, 239–246 (1995)
7. Bryant, D.: The complexity of calculating exemplar distances. In: Sankoff, D., Nadeau, J. (eds.) *Comparative Genomics: Empirical and Analytical Approaches to Gene Order Dynamics, Map Alignment, and the Evolution of Gene Families*, pp. 207–212. Kluwer Acad. Pub., Dordrecht (2000)
8. Blin, G., Rizzi, R.: Conserved interval distance computation between non-trivial genomes. In: Wang, L. (ed.) *COCOON 2005*. LNCS, vol. 3595, pp. 22–31. Springer, Heidelberg (2005)
9. Bergeron, A., Stoye, J.: On the similarity of sets of permutations and its applications to genome comparison. In: Warnow, T.J., Zhu, B. (eds.) *COCOON 2003*. LNCS, vol. 2697, pp. 68–79. Springer, Heidelberg (2003)
10. Chen, Z., Fu, B., Zhu, B.: The approximability of the exemplar breakpoint distance problem. In: Cheng, S.-W., Poon, C.K. (eds.) *AAIM 2006*. LNCS, vol. 4041, pp. 291–302. Springer, Heidelberg (2006)
11. Chen, Z., Fu, B., Fowler, R., Zhu, B.: Lower bounds on the approximation of the exemplar conserved interval distance problem of genomes. In: Chen, D.Z., Lee, D.T. (eds.) *COCOON 2006*. LNCS, vol. 4112, pp. 245–254. Springer, Heidelberg (2006)
12. Chen, Z., Fu, B., Fowler, R., Zhu, B.: On the inapproximability of the exemplar conserved interval distance problem of genomes. *J. Combinatorial Optimization* 15(2), 201–221 (2008)
13. Chen, Z., Fu, B., Yang, B., Xu, J., Zhao, Z., Zhu, B.: Non-breaking similarity of genomes with gene repetitions. In: Ma, B., Zhang, K. (eds.) *CPM 2007*. LNCS, vol. 4580, pp. 119–130. Springer, Heidelberg (2007)
14. Chen, Z., Fu, B., Jiang, M., Zhu, B.: On recovering syntenic blocks from comparative maps. In: Yang, B., Du, D.-Z., Wang, C.A. (eds.) *COCOA 2008*. LNCS, vol. 5165, pp. 319–327. Springer, Heidelberg (2008)
15. Chen, J., Huang, X., Kanj, I., Xia, G.: Linear FPT reductions and computational lower bounds. In: *Proceedings of the 36th ACM Symposium on Theory of Computing (STOC 2004)*, pp. 212–221 (2004)
16. Cormen, T., Leiserson, C., Rivest, R., Stein, C.: *Introduction to Algorithms*, 2nd edn. MIT Press, Cambridge (2001)
17. Choi, V., Zheng, C., Zhu, Q., Sankoff, D.: Algorithms for the extraction of synteny blocks from comparative maps. In: Giancarlo, R., Hannenhalli, S. (eds.) *WABI 2007*. LNCS (LNBI), vol. 4645, pp. 277–288. Springer, Heidelberg (2007)
18. Downey, R., Fellows, M.: *Parameterized Complexity*. Springer, Heidelberg (1999)
19. Garey, M., Johnson, D.: *Computers and Intractability: A Guide to the Theory of NP-completeness*. Freeman, San Francisco (1979)
20. Hästad, J.: Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Mathematica* 182, 105–142 (1999)
21. Hannenhalli, S., Pevzner, P.: Transforming cabbage into turnip: polynomial algorithm for sorting signed permutations by reversals. *J. ACM* 46(1), 1–27 (1999)
22. Gascuel, O. (ed.): *Mathematics of Evolution and Phylogeny*. Oxford University Press, Oxford (2004)

23. Marron, M., Swenson, K., Moret, B.: Genomic distances under deletions and insertions. *Theoretical Computer Science* 325(3), 347–360 (2004)
24. Makaroff, C., Palmer, J.: Mitochondrial DNA rearrangements and transcriptional alternatives in the male sterile cytoplasm of *Ogura* radish. *Mol. Cell. Biol.* 8, 1474–1480 (1988)
25. Nguyen, C.T., Tay, Y.C., Zhang, L.: Divide-and-conquer approach for the exemplar breakpoint distance. *Bioinformatics* 21(10), 2171–2176 (2005)
26. Palmer, J., Herbon, L.: Plant mitochondrial DNA evolves rapidly in structure, but slowly in sequence. *J. Mol. Evolut.* 27, 87–97 (1988)
27. Sankoff, D.: Genome rearrangement with gene families. *Bioinformatics* 16(11), 909–917 (1999)
28. Schaefer, T.: The complexity of satisfiability problem. In: *Proceedings of the 10th ACM Symposium on Theory of Computing (STOC 1978)*, pp. 216–226 (1978)
29. Sturtevant, A., Dobzhansky, T.: Inversions in the third chromosome of wild races of *drosophila pseudoobscura*, and their use in the study of the history of the species. *Proc. Nat. Acad. Sci. USA* 22, 448–450 (1936)
30. Watterson, G., Ewens, W., Hall, T., Morgan, A.: The chromosome inversion problem. *J. Theoretical Biology* 99, 1–7 (1982)
31. Wang, L., Zhu, B.: On the tractability of maximal strip recovery. In: Chen, J., Cooper, S.B. (eds.) *TAMC 2009. LNCS*, vol. 5532 (2009)
32. Zheng, C., Zhu, Q., Sankoff, D.: Removing noise and ambiguities from comparative maps in rearrangement analysis. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 4, 515–522 (2007)

A Quadratic Kernel for 3-Set Packing^{*}

Faisal N. Abu-Khzam

Department of Computer Science and Mathematics
Lebanese American University
Beirut, Lebanon
faisal.abukhzam@lau.edu.lb
<http://www.csm.lau.edu.lb/fabukhzam>

Abstract. We present a reduction procedure that takes an arbitrary instance of the 3-Set Packing problem and produces an equivalent instance whose number of elements is bounded by a quadratic function of the input parameter. Such parameterized reductions are known as kernelization algorithms, and each reduced instance is called a problem kernel. Our result improves on previously known kernelizations and can be generalized to produce improved kernels for the r -Set Packing problem whenever r is a fixed constant. Improved kernelization for r -Dimensional-Matching can also be inferred.

Keywords: Fixed-parameter algorithms, kernelization, crown decomposition, Set Packing.

1 Introduction

Let π be a problem that is parameterized by some positive integer k . An $f(k)$ kernelization algorithm for π is a polynomial-time pre-processing procedure that takes an instance (I, k) of π as input and produces an equivalent instance (I', k') whose size is bounded by $f(k')$, where $k' \leq k$.

When such an algorithm exists, we say that the problem has an $f(k)$ kernel and each reduced instance is a problem kernel. Such reductions are desired when f is a low-order polynomial function of k .

Let C be a collection of subsets of a finite universe S . A packing of C is a sub-collection of pair-wise disjoint elements of C . Finding a packing of maximum cardinality has applications in many areas including scheduling, computational biology and code optimization [8]. The corresponding decision problem, dubbed Set Packing, is defined formally as follows:

Given: A collection C of sets, and a positive integer k .

Question: Does C have a packing of cardinality k ?

Set Packing is NP-Complete [5], unless every set in the given collection is a pair, in which case it coincides with the Maximum Matching problem in simple

^{*} This research has been supported in part by the research council of the Lebanese American University.

undirected graphs. When the maximum cardinality of every element of C is bounded by some integer r , the problem is known as r -Set Packing. If r is a small constant, r -Set Packing is fixed-parameter tractable [3]. In short, this means that it can be solved by an algorithm that runs in time $O(f(k)n^c)$, where n is the total input size, c is a constant and f is an arbitrary function of the input parameter only.

In this paper, we consider the parameterized search version of 3-Set Packing, henceforth $3SP$, which received a great deal of attention lately. Following the above mentioned work of Downey and Fellows in [3], there was a sequence of improved fixed-parameter algorithms for $3SP$ (see [4,6,7]). In particular, Fellows et al. showed that $3SP$ has a problem kernel whose size is in $O(k^3)$ [4]. In this paper, we present an algorithm that produces kernels whose number of elements (i.e., $|S|$) is in $O(k^2)$ [1].

2 Background

Throughout this paper, we describe a sequence of reduction procedures that apply to a given input instance (S, C, k) of $3SP$. We shall assume that elements of C are size-three subsets of S . Dealing with the more general case is a simple modification of our algorithm and would only improve the size of the resulting kernel. We start with a few definitions, some of which were used in [4] and [6].

For a collection $C' \subset C$, we denote by $val(C')$ the union of all elements of C' . For $S' \subset S$, $C(S') = \{e \in C : e \cap S' \neq \emptyset\}$, and for $x \in S$, we denote by $C(x)$ the set of all elements of C that contain x . Moreover, we denote by $G(x)$ the simple graph whose vertex set, $V(x)$, is $val(C(x))$ and whose edge set is $E(x) = \{yz : \{x, y, z\} \in C\}$.

When (S, C, k) is a yes instance, the target solution is denoted by P . An element of the target packing P will be called a packing-set, while the elements of $val(P)$ will be called packed-elements. We say that a yes instance (S, C, k) is an *extreme yes instance* if $(S, C, k + 1)$ is a no instance.

We introduce the notion of a pair cover, which plays a key role in our reduction procedure. A *pair cover* T of (S, C) is a collection of size-two subsets of S such that every element of C contains (as a subset) an element of T .

The notion of a crown decomposition also plays a major role in our algorithm. A crown decomposition of a simple connected undirected graph G is a triple (H, I, M) such that I is an independent set of G , $H = N_G(I)$, and M is a matching in G that satisfies:

- (i) Every edge of M joins a vertex from H to a vertex of I .
- (ii) Every vertex of H is matched, under M , to a vertex of I .

Not all graphs have crowns. An extreme example of a crown-free graph is any complete graph on three or more vertices. A necessary and sufficient condition for

¹ The cardinality of S is in $O(k^3)$ in the previously known kernel.

a graph to have a crown is to have an independent set I such that $|N_G(I)| \leq |I|$. This can be easily deduced from [2], in which a construction procedure was described.

The size of a crown (H, I, M) is the number of edges in M . A crown of maximum size can be constructed in $O(|V(G)|^{2.5})$, as described in [1]. The maximum crown construction procedure, which we call *Construct_Crown*, can be applied whenever an independent set I that satisfies the above condition ($|N_G(I)| \leq |I|$) is found. See [1] for more details.

A cubic-size kernelization algorithm for 3SP was described by Fellows et al. in [4]. In this paper, the main focus is on how to obtain a quadratic upper bound on the number of elements of a 3SP kernel. We shall omit the complete details of efficiency analysis whenever it is clear that the run time is polynomial in the input size, which is in $\theta(|S| + |C|)$.

3 A Reduction Procedure

Our reduction process consists of three steps. The first two are based on simple counting arguments. We shall assume (S, C, k) is a given instance of 3SP and that P is a potential solution, if any.

3.1 The High-Degree Rule

During the search for a large packing, we may determine that an element of S can be packed in at least one solution. Such is the case of elements that belong to a sufficiently large number of sets.

The high-degree rule: If $k \geq 0$ and $\{x\} \subset S$ is the intersection of $3k - 2$ elements of C , then delete x and all the sets of $C(x)$, and decrement k by one.

The soundness of this reduction rule is due to a simple observation: if $C(x)$ contains $3k - 2$ sets whose pair-wise intersection is $\{x\}$, then any solution (or packing) of $(S \setminus \{x\}, C \setminus C(x), k - 1)$ has an empty intersection with at least one element of $C(x)$. Since $C(x)$ can contribute at most one element to any packing, it follows easily that: (S, C, k) has a solution if and only if $(S \setminus \{x\}, C \setminus C(x), k - 1)$ has a solution.

From this point on, we say that an instance (S, C, k) is preprocessed if the high-degree rule cannot be applied to reduce it further.

3.2 Using Pair Covers

Given a preprocessed instance (S, C, k) , we construct a pair cover using the following procedure. (Note that a maximal packing of (S, C) can be constructed in $O(|C|)$ time using a simple greedy approach.)

Procedure Construct_Pair_Cover**Input:** Preprocessed instance (S, C, k) of 3SP**Output:** Either a packing of size $\geq k$ of (S, C) , or a pair cover T **Begin**

Construct a maximal packing P of (S, C)
 If $|P| \geq k$
 Return P
 $H = \text{val}(P)$
 For each $x \in H$ do
 Construct a maximal matching M_x in $G(x)$
 For each edge $\{y, z\}$ of M_x do
 $T \leftarrow T \cup \{\{x, y\}, \{x, z\}\}$
 Return T .

End

The correctness of *construct_Pair_Cover* is obvious. If a greedily constructed packing P has less than k sets, then the output is a set T of size-two subsets of S . Each triple $\{u, v, w\}$ in C contains at least one element, say u , from $H = \text{val}(P)$. At least one of v and w , say v , is in M_u . Thus $\{u, v\}$ is an element of T . This proves that T is a pair cover. We now prove that its size is quadratic in k .

Lemma 1. *There is a polynomial time algorithm that takes an arbitrary instance (S, C, k) of 3SP as input and produces any of the following:*

- a packing of size k or more;
- a No answer if it detects that no solution exists;
- a pair cover whose size is at most $2(3k - 3)^2$.

Proof. Given an arbitrary instance (S, C, k) of 3SP, we apply the high-degree rule followed by the *construct_Pair_Cover* procedure. If we fail to find a solution and we do not detect that (S, C, k) is a no instance, then the output of *construct_Pair_Cover* is a collection T of size-two subsets of S . In this case, the set H found in the *construct_Pair_Cover* procedure has size $3|P| \leq 3k - 3$. Every element x of H can be the unique intersection of at most $3k - 3$ elements of $C(x)$. Therefore, M_x has at most $3k - 3$ elements. It follows that at most $2(3k - 3)$ pairs of T contain x . This completes the proof.

3.3 Using Crown Decomposition

Assume that (S, C, k) is a preprocessed yes instance of 3SP. Let T be a pair cover and let $I = \{x \in S : x \notin \text{val}(T)\}$. Observe that no two elements of I belong to the same element (or triple) of C . This is the case because any triple of C that contains two elements of I is not covered by T .

If $|I| \leq |T|$ then we already have a quadratic number of elements in S . Assume, in the sequel, that $|I| > |T|$. In this case, our algorithm proceeds by constructing a simple bipartite graph, $G_{I,T}$, whose vertex set is $I \cup T$ and whose edge set is $\{(x, \{y, z\}) : x \in I, \{y, z\} \in T \text{ and } \{x, y, z\} \in C\}$. Then a crown decomposition, (T', M_G, I') , of $G_{I,T}$ is used.

Our *3SP* kernelization algorithm, shown below, uses the three steps discussed in this section. A crown reduction is used by identifying the set $I'' \subset I' \subset I$ that can be deleted. The correctness of this step is due to Lemma 2 below. Note that we used $val(M_G)$ to denote the union of all pairs of T' together with the elements of I that are matched under M_G .

Algorithm 3SP-Kernel

Input: Instance (S, C, k) of *3SP*, such that every element of C contains exactly 3 elements of S

Output: Either No if (S, C) has no set packing of size $\geq k$, or a solution, or an instance (S', C', k') such that $|S'| \leq 4(3k - 3)^2$

Begin

Apply the high-degree rule

If $k > 0$ and $C = \phi$

Return No

$T \leftarrow \text{Construct_Pair_Cover}(S, C, k)$

If T is a matching of size $\geq k$

Return T

Construct the simple bipartite graph $G_{I,T}$

If $|I| > |T|$

$(T', M_G, I') \leftarrow \text{Construct_Crown}(G_{I,T})$

$I'' \leftarrow I' \setminus val(M_G)$

Remove I'' and $C(I'')$

Return the (possibly) new instance

End

Lemma 2. Let I and $G(I, T)$ be as defined above and assume $|I| > |T|$. Let (T', M_G, I') be a maximum crown in $G_{I,T}$ and let I'' be the set of vertices of I' that are not matched under M_G . Then (S, C, k) has a solution if and only if the instance $(S \setminus I'', C \setminus C(I''), k)$ has a solution.

Proof. Again, let P be a maximum packing in (S, C) , and recall that T is a pair cover constructed by *Construct_Pair_Cover*. Let $P' = \{e \in P : e \cap I'' \neq \phi\}$. In other words, P' is the set of packing-sets that contain elements of I' that are not matched under M_G . Let $T'' = \{p \in T' : p \text{ is contained in an element of } P'\}$. It is easy to observe that $|T''| = |P'|$.

Now let A denote the set of elements of I' that are matched in the crown (under M_G) with elements of T'' . Since $A \cap \text{val}(T'') = \emptyset$, we can have a packing $P'' = \{\{x, y, z\} : \{x, y\} \in T'' \text{ and } z \in A\}$. It follows that $|P''| = |T''| = |P'|$. Therefore P' can be replaced by P'' and the packed elements (under P) of I'' can be deleted. This completes the proof.

Note that, in the above discussion, $I \setminus I'$ is smaller than its neighborhood in T . Otherwise, a larger crown can be constructed, contrary to our assumption. We now state our kernelization theorem.

Theorem 1. *There is a polynomial-time algorithm that, for an arbitrary input instance (S, C, k) of 3SP, either determines that (S, C, k) is a no instance, or finds a solution, or computes a kernel instance (S', C', k') such that $|S'|$ is bounded above by $4(3k - 3)^2$.*

Proof. Based on the above reduction procedures and lemmas, the number of elements that remain in I cannot exceed the number of pairs in T . The theorem follows from the fact that T has at most $2(3k - 3)^2$ pairs.

4 Conclusion

We presented work in progress on the r -Set packing problem. We showed how to obtain an improved kernelization algorithm for 3-Set Packing that achieves a quadratic bound (in the parameter) on the number of elements in the reduced kernel instance. The use of the notion of a pair cover played a key role in our reduction procedure. We believe that pair covers can be used for an improved fixed-parameter algorithm.

Our method can be generalized to a kernelization algorithm for r -Set Packing, where r is a small constant. To do this, the number of sets in the condition of the high-degree rule becomes $r(k - 1)$, and the pair cover is replaced by an $(r - 1)$ -tuple cover. A similar approach can be applied to r -Dimensional Matching.

References

1. Abu-Khizam, F.N., Fellows, M.R., Langston, M.A., Suters, W.H.: Crown Structures for Vertex Cover Kernelization. *Theory of Computing Systems (TOCS)* 41(3), 411–430 (2007)
2. Chor, B., Fellows, M.R., Juedes, D.: Linear Kernels in Linear Time, or How to Save k Colors in $O(n^2)$ Steps. In: Hromkovič, J., Nagl, M., Westfechtel, B. (eds.) *WG 2004*. LNCS, vol. 3353, pp. 257–269. Springer, Heidelberg (2004)
3. Downey, R.G., Fellows, M.R.: *Parameterized Complexity*. Springer, New York (1999)
4. Fellows, M.R., Knauer, C., Nishimura, N., Ragde, P., Rosamond, F., Stege, U., Thilikos, D.M., Whitesides, S.: Faster fixed-parameter tractable algorithms for matching and packing problems. *Algorithmica* 52(2), 167–176 (2008)
5. Garey, M.R., Johnson, D.S.: *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York (1979)

6. Jia, W., Zhang, C., Chen, J.: An efficient parameterized algorithm for m -set packing. *J. Algorithms* 50, 106–117 (2004)
7. Koutis, I.: A faster parameterized algorithm for set packing. *Information Processing Letters* 94, 7–9 (2005)
8. Liu, Y., Chen, J., Wang, J.: Parameterized Algorithms for Weighted Matching and Packing Problems. In: Cai, J.-Y., Cooper, S.B., Zhu, H. (eds.) *TAMC 2007*. LNCS, vol. 4484, pp. 692–702. Springer, Heidelberg (2007)

Quantitative Aspects of Speed-Up and Gap Phenomena

Klaus Ambos-Spies and Thorsten Kräling

Institut für Informatik, University of Heidelberg, D-69120 Heidelberg, Germany
ambos@math.uni-heidelberg.de, kraeling@informatik.uni-heidelberg.de

Abstract. We show that, for any abstract complexity measure in the sense of Blum and for any computable function f (or computable operator F), the class of problems which are f -speedable (or F -speedable) does not have effective measure 0. On the other hand, for sufficiently fast growing f (or F), the class of the nonspeedable problems does not have effective measure 0 too. These results answer some questions raised by Calude and Zimand in [CZ96] and [Zim06]. We also give a short quantitative analysis of Borodin and Trakhtenbrot's Gap Theorem which corrects a claim in [CZ96] and [Zim06].

1 Introduction

In the early 1990s Lutz [Lu92] started a program for a quantitative analysis of phenomena in structural complexity theory. This program, which attracted quite a number of researchers, built on resource-bounded variants of Schnorr's effective measure [Sch73] and Mehlhorn's effective Baire category [Me73] which allowed to apply these classical classification tools to the (countable) computable universe. While most of Lutz's program was devoted to complexity classes in the lower part of the intractable world, like the exponential time classes, Calude and Zimand [CZ96] exploited these tools for the quantitative analysis of more general phenomena encountered in all of the common complexity measures, thereby taking up some earlier lines of research by Mehlhorn and others.

In particular, Calude and Zimand applied effective measure and category to a quantitative analysis of the fundamental theorems in Blum's abstract complexity theory [Blu67] like the Speed-Up Theorem and the Gap Theorem. For instance, they have shown that, in the sense of effective category, speedable sets are not rare, namely, for any complexity measure and for any effective operator F , the class of the computable sets which are F -speedable is not effectively meager in the sense of Mehlhorn. Calude and Zimand [CZ96] raised the question whether this observation is true in the sense of effective measure too.

Here we answer this question affirmatively by showing that (in any Blum space) the class of F -speedable problems does not have effective measure 0 in the sense of Schnorr (Section 3). We also show, however, that for sufficiently fast growing f (or F) the class of the nonspeedable problems does not have effective measure 0 too, and we obtain the corresponding result for Mehlhorn's effective

category (Section 4). Our results may be interpreted so that the effective measure and category concepts of Schnorr and Mehlhorn, respectively, are too coarse in order to give a complete quantitative analysis of the speed-up phenomena.

In case of another important result of abstract complexity theory, however, namely the Gap Theorem, we give a complete quantitative analysis. For sufficiently fast growing effective operators F , F -gaps are rare, i.e., the class of the computable complexity bounds t at which F -gaps occur is effectively meager and has effective measure 0 (in the Baire space). The former corrects a claim by Calude and Zimand in [CZ96] while the latter answers a question raised there (Section 5).

Before we present our results, in the next section we shortly review some basic concepts and results of Blum's abstract complexity theory ([Blu67]) and of effective measure which we will need. For a more complete account of this material see e.g. the recent monograph of Zimand [Zim06].

2 Preliminaries

Abstract Complexity Measures. An *abstract complexity measure* (or *Blum space* for short) is a pair (φ, Φ) of binary partially computable functions where φ is a Gödel numbering of the unary partially computable functions and Φ satisfies the following two conditions, called *Blum axioms*:

- (B1) $\text{dom}(\varphi) = \text{dom}(\Phi)$
- (B2) $\text{graph}(\Phi) = \{(e, x, y) : \Phi_e(x) = y\}$ is computable.

In the following we will focus on the complexity of computable sets. So we will assume that the functions φ_e are 0-1-valued and identify a set with its characteristic function.

Among the most interesting phenomena which hold for all Blum spaces are the Speed-Up Theorem and the Gap Theorem. Blum's Speed-Up Theorem says that there are computable problems without optimal solutions even if we measure the costs only up to a very large (e.g. exponential) factor. Formally, given a computable function $f : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$, a computable set A is *f-speedable* with respect to Blum space (φ, Φ) if

$$(\forall e)(\varphi_e = A \Rightarrow (\exists e')(\varphi_{e'} = A \text{ and } f(x, \Phi_{e'}(x)) \leq_{a.e.} \Phi_e(x))) \quad (1)$$

(where $g \leq_{a.e.} h$ denotes that $g(x) \leq h(x)$ for almost all x).

Theorem 1. (Speed-Up Theorem – Function Version; [Blu67]) *Let f be a computable function of type $\mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ and let (φ, Φ) be a Blum space. There is a computable f -speedable set A , i.e.,*

$$\text{SPEED}((\varphi, \Phi), f) = \{A \in \text{REC} : A \text{ } f\text{-speedable}\} \neq \emptyset.$$

As Meyer and Fischer have shown, the Speed-Up Theorem can be strengthened by replacing the computable function f by a total effective operator F . A computable set A is *F-speedable* with respect to (φ, Φ) if

$$(\forall e)(\varphi_e = A \Rightarrow (\exists e')(\varphi_{e'} = A \text{ and } F(\Phi_{e'})(x) \leq_{a.e.} \Phi_e(x))). \quad (2)$$

Theorem 2. (Speed-Up Theorem – Operator Version; [MF72]) *Let F be a total effective operator and let (φ, Φ) be a Blum space. There is a computable F -speedable set A , i.e.,*

$$\text{SPEED}((\varphi, \Phi), F) = \{A \in \text{REC} : A \text{ } F\text{-speedable}\} \neq \emptyset.$$

The second fundamental observation on arbitrary complexity measures which we will discuss is Trakhtenbrot and Borodin’s Gap Theorem which states that for any total effective operator F there is a computable cost function t such that an increase of the complexity bound t by the factor F will not allow the solution of any additional problems. For the formal statement let

$$C_t^{(\varphi, \Phi)} = \{A \in \text{REC} : (\exists e)(\varphi_e = A \text{ and } \Phi_e(x) <_{a.e.} t(x))\} \tag{3}$$

denote the (φ, Φ) -complexity class with bound (name) t .

Theorem 3. (Gap Theorem; [Tra67], [Bor72]) *Let (φ, Φ) be a Blum space and let F be a total effective operator such that, for every partial computable function ψ , $F(\psi)(n) \geq_{a.e.} \psi(n)$. There is a computable function t such that $C_t^{(\varphi, \Phi)} = C_{F(t)}^{(\varphi, \Phi)}$.*

For our proofs we will need Blum’s observation that all Blum spaces are computably related to each other as follows.

Theorem 4. (Recursive-Relatedness Theorem; [Blu67]) *Let (φ, Φ) and (ψ, Ψ) be Blum spaces and let h be a computable translation function from φ to ψ . There is a strictly increasing 2-ary computable function g such that*

$$(\forall e)(\Psi_{h(e)}(x) \leq_{a.e.} g(x, \Phi_e(x)) \text{ and } \Phi_e(x) \leq_{a.e.} g(x, \Psi_{h(e)}(x))). \tag{4}$$

Effective Measure. We identify a set A of natural numbers with its characteristic sequence whence A may be viewed as an element of the Cantor space 2^ω . The classical Lebesgue measure on 2^ω can be described in terms of martingales (betting games). By considering only computable martingales this yields an *effective* measure on 2^ω (Schnorr [Sch73]). For developing this theory it suffices to consider rational valued martingales d of norm 1 (see e.g. [ASM97]).

Definition 1. (Schnorr [Sch73], Lutz [Lu92]).

(a) *A (rational valued, normed) martingale d is a function $d : 2^{<\omega} \rightarrow \mathbb{Q}_{\geq 0}$ such that $d(\lambda) = 1$ and d satisfies the fairness condition*

$$d(w) = \frac{d(w0) + d(w1)}{2}. \tag{5}$$

A martingale d succeeds on a set $A \in 2^\omega$ if $\limsup_{n \rightarrow \infty} d(A \upharpoonright n) = \infty$ (where $A \upharpoonright n = A(0) \dots A(n-1)$), and d succeeds on a class $C \subseteq 2^\omega$ if d succeeds on all sets $A \in C$.

(b) A class $C \subseteq 2^\omega$ has effective measure 0 if there is a computable martingale which succeeds on C , and C has effective measure 1 if the complement of C has effective measure 0.

(c) A class C has measure 0 in REC if $C \cap \text{REC}$ has effective measure 0, and C has measure 1 in REC if the complement of C has measure 0 in REC.

Intuitively, a class C of computable sets is *small* if it has effective measure 0 (or, equivalently, measure 0 in REC), *non-small* if it does not have effective measure 0 (or, equivalently, does not have measure 0 in REC), and *large* if it has measure 1 in REC.

3 A Measure-Theoretic Version of the Speed-Up Theorem

Our first result is the measure theoretic analog of Calude and Zimand’s quantitative analysis of the Speed-Up Theorem in terms of category. It answers the central open question of [CZ96].

Theorem 5. (Operator Speed-Up – Measure-Theoretic Version) *Let F be a total effective operator and let (φ, Φ) be a Blum space. Then the class of the F -speedable sets, $\text{SPEED}((\varphi, \Phi), F)$, does not have effective measure 0.*

Since the basic ideas of the proof can already be found in the less involved proof of the weaker function version, here we will only sketch a proof of the latter.

Theorem 6. (Function Speed-Up – Measure-Theoretic Version) *Let f be a computable function of type $\mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ and let (φ, Φ) be a Blum space. Then the class $\text{SPEED}((\varphi, \Phi), f)$ does not have effective measure 0.*

Proof. (SKETCH) By the Recursive-Relatedness Theorem, for any Blum spaces (φ, Φ) and (ψ, Ψ) and any computable function f there is a computable function f' such that $\text{SPEED}((\psi, \Psi), f') \subseteq \text{SPEED}((\varphi, \Phi), f)$. Hence it suffices to prove the theorem for an appropriately chosen Blum space (φ, Φ) . Here we let $(\varphi_e)_{e \in \mathbb{N}}$ be the Gödel numbering induced by a standard enumeration of the Turing machines $(M_e)_{e \in \mathbb{N}}$, and let $\Phi_e = \text{time}_{M_e}$ be the run time of M_e .

So, given a computable martingale d , it suffices to construct a set A in $\text{SPEED}((\varphi, \Phi), f)$ for the given Blum space (φ, Φ) such that d does not succeed on A . In fact, since $\text{SPEED}((\varphi, \Phi), g) \subseteq \text{SPEED}((\varphi, \Phi), f)$ for any computable function g dominating f , we may replace f by a computable function $g > f$ such that g and its diagonal $h(n) = g(n, n)$ have the following properties:

- (i) $g(m, n)$ is strictly increasing in m and n ;
- (ii) for some Turing machine M computing d , $h(n) > \text{time}_M(w)$ for all strings w of length $n + 1$;
- (iii) $h(0) > 0$ and for all n , $h(n + 1) > 2^{h(n)} \geq 2^{n+1}$;
- (iv) h is time constructible, i.e., there is a Turing machine M' such that $\text{time}_{M'}(n) = h(n)$ for all $n \geq 0$.

Note that a function $g > f$ with the above properties can be easily constructed by the standard technique for constructing a time constructible function above a given computable function.

Now, a g -speedable set A on which the given computable martingale d does not succeed is constructed by a *slow diagonalization* (or *wait-and-see argument*) where $A(s)$ is determined at stage s of the construction.

For making A g -speedable we use a sequence of functions $(u_e)_{e \in \mathbb{N}}$ defined by

$$u_e(x) = h^{x \dot{-} e}(x).$$

Note that, by choice of h , the functions u_e are time constructible and

$$\forall x > e (h(u_{e+1}(x)) = u_e(x) \geq x) \tag{6}$$

($e \geq 0$). I.e., the e th function u_e exceeds the $(e+1)$ th function u_{e+1} by the factor h on all inputs $x > e$.

Then, in order to guarantee that A is g -speedable, it suffices to ensure

$$(\forall e)(\exists e')(A = \varphi_{e'} \text{ and } \text{time}_{M_{e'}}(x) \leq_{a.e.} u_e(x)) \tag{7}$$

and to meet the requirements

$$R_e : A = \varphi_e \Rightarrow \text{time}_{M_e}(x) >_{a.e.} u_e(x) \tag{8}$$

for all $e \geq 0$. Namely, given a machine M_e with $\varphi_e = A$, we obtain a machine $M_{e'}$ which computes A and satisfies

$$g(x, \text{time}_{M_{e'}}(x)) <_{a.e.} \text{time}_{M_e}(x) \tag{9}$$

as follows. By (7), there is an index e' such that $\varphi_{e'} = A$ and $\text{time}_{M_{e'}}(x) \leq_{a.e.} u_{e+1}(x)$. Since g is increasing in both arguments and since $x \leq u_{e+1}(x)$, it follows that

$$g(x, \text{time}_{M_{e'}}(x)) \leq_{a.e.} g(u_{e+1}(x), u_{e+1}(x)) = h(u_{e+1}(x)).$$

This implies (9) since, by (8) and (6), $\text{time}_{M_e}(x) >_{a.e.} u_e(x) =_{a.e.} h(u_{e+1}(x))$.

The basic strategy for satisfying the requirements R_e , $e \geq 0$, is as follows. If there are infinitely many numbers x such that $\text{time}_{M_e}(x) \leq u_e(x)$ (note that R_e is trivially met otherwise) then requirement R_e is met by diagonalization, i.e., by ensuring $A(x) \neq \varphi_e(x)$ for some such x . To be more precise, at any stage $s > e$ such that R_e has not been satisfied at any previous stage and such that $\text{time}_{M_e}(s) \leq u_e(s)$ we say that requirement R_e *requires attention* at stage s . If R_e is the highest priority requirement which requires attention at stage s (i.e., e is the least number e' such that $R_{e'}$ requires attention at stage s) then let $A(s) = 1 \dot{-} \varphi_e(s)$ and declare R_e to be *active* and *satisfied* at stage s . (If no requirement requires attention, let $A(s) = 0$.)

Note that, by the Blum axioms, we can decide whether R_e requires attention at stage s (in fact, by time constructibility of u_e , this can be done in $u_e(s)$ steps), and – if so – $\varphi_e(s)$ is defined (and can be computed in $\text{time}_{M_e}(s) \leq u_e(s)$

steps), whence $A(s) \neq \varphi_e(s)$ if R_e becomes satisfied at stage s . Moreover, any requirement will become active at most once.

So, by defining A according to the above strategy, A will be computable, and, by a straightforward induction, any requirement which is not trivially met will eventually become satisfied.

Moreover, the above strategy for meeting the requirements R_e is compatible with ensuring condition (7). Given e , it suffices to define a Turing machine M computing $A(s)$ in $\leq u_e(s)$ steps for almost all s . Such a machine is obtained as follows. Since any requirement acts at most once, we may fix a stage $s_e > e$ such that no requirement $R_{e'}$ with $e' \leq e$ acts after stage s_e . So if we let SAT be the finite list of requirements $R_{e'}$ with $e' > e$ satisfied by the end of stage s_e then, for $s > s_e$, in the above described procedure for computing $A(s)$ we may ignore all requirements $R_{e'}$ such that $e' \leq e$ or $e' \in SAT$ and we will still get the correct value of $A(s)$. Since, for any e' such that $R_{e'}$ has not been satisfied prior to stage s , in a total of $O(u_{e'}(s))$ steps we can decide whether $R_{e'}$ requires attention at stage s and if so compute the value of $\varphi_{e'}(s)$, it follows (by a straightforward induction) that a Turing machine M formalizing the thus simplified procedure for computing $A(s)$ will run in time $time_M(s) \leq s \cdot O(u_{e+1}(s) + \dots + u_s(s))$ whence, by (6) and by property (iii) of h , $time_M(s) \leq u_e(s)$.

Having explained the basic strategy for making A g -speedable, we next will show how this strategy can be modified so that the martingale d does not succeed on A . In order to guarantee that d does not succeed on A , it suffices to meet the requirements

$$Q_s : d(A \upharpoonright s + 1) \leq 1 \tag{10}$$

($s \geq 0$). Note that there is a trivial strategy for meeting these requirements: If we let $A(s) = 0$ if $d((A \upharpoonright s)0) \leq d((A \upharpoonright s)1)$ and $A(s) = 1$ otherwise then, by the fairness property (5) of d , $d(A \upharpoonright s + 1) \leq d(A \upharpoonright s)$. (If $A(s)$ will be defined this way then we say that $A(s)$ is *defined according to the basic Q -strategy*.) So using this strategy the value of $d(A \upharpoonright s)$ is nonincreasing in s whence, by $d(\lambda) = 1$, all Q -requirements are met. Of course this basic Q -strategy is too strict and it has to be relaxed in order to become compatible with our strategy for meeting the requirements R_e .

Recall that, for a requirement R_e which is not trivially met, there are infinitely many stages at which R_e may become satisfied by becoming active. (In the following call such a requirement *nontrivial*.) So it suffices to ensure that at at least one of these stages s requirement R_e can become satisfied without hurting the corresponding Q -requirement Q_s . One might be tempted to allow the highest priority requirement R_e which requires attention at stage s to act if letting $A(s) = 1 - \varphi_e(s)$ will not injure Q_s . But this procedure is not fair and may result in some of the nontrivial requirements never becoming satisfied. So we have to design a slightly more involved strategy which will treat the individual R -requirements fairly and will guarantee that any nontrivial requirement will eventually become satisfied.

The idea of this refined strategy is roughly as follows. If requirement R_e is entitled to become satisfied at stage s but instead of satisfying requirement R_e

(by letting $A(s) = 1 - \varphi_e(s)$) we define $A(s)$ according to the basic Q -strategy then (by (5)) the value of d will *strictly decrease* at stage s , i.e., $d(A \upharpoonright s + 1) < d(A \upharpoonright s)$ (since otherwise $d((A \upharpoonright s)0) = d((A \upharpoonright s)1)$ whence there were no reason to prevent R_e from becoming satisfied at stage s). Now, whenever R_e is entitled to act but not allowed to do so, then the resulting decrease $d(A \upharpoonright s) - d(A \upharpoonright s + 1)$ in the value of d is paid into an “account” of R_e , and R_e will be allowed to become satisfied at a stage at which it is entitled to act, if the increase of d caused by this action is bounded by the current balance of R_e ’s account. This will work since a nontrivial requirement R_e which were never satisfied were entitled to act infinitely often and the balance of its account were unbounded. So, eventually, the balance will cover the growth of d caused by satisfying R_e (which is bounded by 1) whence R_e will become satisfied.

More formally, the construction of A is as follows (using the concepts introduced above when describing the basic R - and Q -strategies). If R_e is the highest priority requirement which requires attention at stage s then we say that R_e is *entitled to act* at stage s and we call s an *e-stage* (and we call s a *-1-stage* if no requirement requires attention at stage s).

If s is an *e-stage* but R_e does not become active at stage s or if s is a *-1-stage* then we define $A(s)$ according to the basic Q -strategy, i.e., let $A(s) = 0$ if $d((A \upharpoonright s)0) \leq d((A \upharpoonright s)1)$ and $A(s) = 1$ otherwise.

In order to decide whether the requirement which is entitled to act at stage s (if any) actually becomes active, we attach to any requirement R_e and any stage s a rational $b_e(s)$ (denoting the balance of R_e ’s account at the end of stage s) where

$$b_e(s) = \sum_{s' < s, s' \text{ e-stage}} d(A \upharpoonright s') - d(A \upharpoonright s' + 1).$$

if $s > 0$ and R_e has not been satisfied by stage s , and $b_s(e) = 0$ otherwise.

Now, if R_e is entitled to act at stage s and

$$d((A \upharpoonright s)(1 - \varphi_e(s))) \leq d(A \upharpoonright s) + b_e(s - 1), \tag{11}$$

then let $A(s) = 1 - \varphi_e(s)$ and say that requirement R_e becomes *active* and *satisfied* at stage s .

This completes the actual construction of A . The correctness of the construction of A is established along the following lines.

By a straightforward induction on s , $b_e(s) \geq 0$ and

$$d(A \upharpoonright s + 1) + \sum_{e=0}^{\infty} b_e(s) \leq 1. \tag{12}$$

So, in particular, the requirements Q_s are met whence d does not succeed on A .

Moreover, every requirement R_e is met and will be entitled to act at most finitely often. Namely, for a contradiction, fix e minimal such that R_e is not met or is entitled to act infinitely often. Then R_e will be nontrivial, never satisfied, and, by minimality of e , entitled to act infinitely often. It follows that $b_e(s)$ is nondecreasing in s ; for the first *e-stage* s_0 , $b_e(s_0) > 0$; and, for any of the

following e -stages s' , $b_e(s') > 2b_e(s' - 1)$ (by (5) and by failure of (11) for $s = s'$). So $b_e(s)$ is unbounded contrary to (12).

Finally, to show that A satisfies (7), we can argue as in the case of the preliminary construction above. It suffices to note that, by property (ii) of h , the added parts involving computations of the martingale d will not lead to an essential increase of the complexity of the construction.

This completes the proof.

4 A Refined Quantitative View of the Speed-Up Theorem

Our measure theoretic version of the Speed-Up Theorem shows that in any Blum space the class of F -speedable (hence the class of f -speedable) sets does not have effective measure 0, hence *is not small*. But is this class *large*? By Definition 1 (c) we may formalize this question, by asking whether $\text{SPEED}((\varphi, \Phi), F)$ has measure 1 in REC, i.e., whether the class of the computable non- F -speedable sets has effective measure 0. Our next theorem shows that for sufficiently fast growing functions f (hence for sufficiently fast growing operators F) the answer is negative.

Theorem 7. *Let (φ, Φ) be a Blum space. There is a computable function f such that for $C = \text{SPEED}((\varphi, \Phi), f)$ the class $C^c \cap \text{REC}$ does not have effective measure 0.*

Proof. (IDEA) By the Recursive-Relatedness Theorem it suffices to prove the theorem for the Blum space (φ, Φ) of the Turing computable functions together with time complexity.

We exploit some facts on time-bounded random sets (compare e.g. with the corresponding results for resource-bounded random sets of strings in [ASM97]). For a computable function $t(n)$, a set A is $t(n)$ -random if there is no martingale d which can be computed in time $t(n)$ and which succeeds on A . Randomness is related to measure as follows. A class does not have effective measure 0 iff for any computable function t there is a $t(n)$ -random set in the class.

Now, as one can easily show, there is a number $k \geq 1$ such that, for any nondecreasing time constructible function $t(n) \geq n$, there is a $t(n)^2$ -random set A_t which can be computed in time $t(n)^k$ while, on the other hand, no $t(n)^2$ -random set can be computed in time $t(n)$. So, A_t is not f -speedable for the exponential function $f(m, n) = 2^n$. We deduce that for any computable t there is a computable non- f -speedable $t(n)$ -random set A whence $C^c \cap \text{REC}$ does not have effective measure 0.

By a similar argument, we obtain the corresponding result in terms of effective Baire category.

Theorem 8. *Let (φ, Φ) be a Blum space. There is a computable function f such that for $C = \text{SPEED}((\varphi, \Phi), f)$ the class $C^c \cap \text{REC}$ is not effectively meager.*

So it seems that both, effective measure and effective category, are too weak in order to decide whether a *typical* computable set is speedable or not. For a weaker property, namely almost-everywhere complexity, there is a similar situation in case of category. Both, the class of the computable a.e.- $t(n)$ -complex sets and the class of the computable sets which are not a.e.- $t(n)$ -complex, are not effectively meager (see e.g. [CZ96] and Mayordomo [May94], respectively). For a refinement of Mehlhorn’s effective Baire category concept based on *partial* extension functions (see [ASR97]), however, the question of which case is the typical one can be resolved. Namely, for this stronger concept, the class of the computable sets which are not a.e.- $t(n)$ -complex is effectively meager (see [AS96]). As we can show, however, this refined effective category concept does not answer the corresponding question for speedability.

5 The Gap Theorem – The Quantitative View

We close our quantitative analysis of the fundamental results of abstract complexity theory by a short look at the Gap Theorem. Calude and Zimand give a quantitative analysis of this theorem in terms of Mehlhorn’s effective Baire category [Me73] on the Baire space. They claim that, for any Blum space (φ, Φ) and for any total effective operator F satisfying $F(\psi) \geq_{a.e.} \psi$ for all partial computable functions ψ , the class

$$\text{GAP}_F^{(\varphi, \Phi)} = \{t \in \text{FREC} : C_t^{(\varphi, \Phi)} = C_{F(t)}^{(\varphi, \Phi)}\}$$

is not effectively meager (see [CZ96], Theorem 4.1 and [Zim06], Theorem 2.4.1). So, in a topological sense, F -gaps are not uncommon. The proof of this result, however, is erroneous. In fact, for any Blum space and for any sufficiently large effective operator F , the gap class $\text{GAP}_F^{(\varphi, \Phi)}$ is effectively meager, i.e., F -gaps are rare in the sense of effective Baire category.

Theorem 9. *Let (φ, Φ) be a Blum space. There is a total effective operator F , where $F(\psi)(n) \geq \psi(n)$ for all partial computable functions ψ and almost all numbers n , such that $\text{GAP}_F^{(\varphi, \Phi)}$ is effectively meager.*

Proof. (IDEA) Given a computable function $g : \mathbb{N} \rightarrow \mathbb{N}$, the class

$$C_{a.e. \neq g} = \{t \in \text{FREC} : t(n) \neq_{a.e.} g(n)\} \text{ is effectively meager} \quad (13)$$

since, for any $m \geq 0$, the class $\{t \in \text{FREC} : (\forall n \geq m)(t(n) \neq g(n))\}$ is effectively nowhere dense via the computable extension function $f_m(\alpha) = \alpha 0^m g(|\alpha| + m)$ whence $C_{a.e. \neq g}$ is the union of uniformly effectively nowhere dense classes (for the basic definitions of effective Baire category, see e.g. [Zim06]). Since, for any time bound t such that $C_t^{(\varphi, \Phi)} \neq \emptyset$, $t(n) >_{a.e.} 0$, i.e., $t(n) \neq_{a.e.} g(n)$ for the constant null function g , we may conclude that

$$\text{NE}^{(\varphi, \Phi)} = \{t \in \text{FREC} : C_t^{(\varphi, \Phi)} \neq \emptyset\} \text{ is effectively meager.} \quad (14)$$

Now take any computable set A , let e be an index of A in (φ, Φ) , and define the total effective monotone operator F by letting $F(\psi)(n) = \max(\psi(n), \Phi_e(n) + 1)$. Then, for any computable time bound t , $A \in C_{F(t)}^{(\varphi, \Phi)}$, whence $C_t^{(\varphi, \Phi)} = C_{F(t)}^{(\varphi, \Phi)}$ implies that $C_t^{(\varphi, \Phi)} \neq \emptyset$. So $\text{GAP}_F^{(\varphi, \Phi)}$ is contained in the class $\text{NE}^{(\varphi, \Phi)}$, hence effectively meager by (14).

The proof of Theorem 9 shows that a quantitative analysis of computable time bounds in terms of effective Baire category is rather useless since the class of the computable time bounds of the nontrivial complexity classes is effectively meager, hence small. The analog of Theorem 9 for effective measure holds too. If we let f be a computable function $f : \mathbb{N} \rightarrow \mathbb{Q}$ defining a nonvanishing measure on \mathbb{N} (i.e. satisfying $f(n) > 0$ for all n and $\sum_{n \in \mathbb{N}} f(n) = 1$) then, for the corresponding effective product measure μ_f on the Baire space, $\mu_f(\text{GAP}_F^{(\varphi, \Phi)}) = 0$ for all sufficiently large effective functionals F .

References

- [AS96] Ambos-Spies, K.: Resource-bounded genericity. In: Cooper, S., Slaman, T., Wainer, S. (eds.) *Computability, enumerability, unsolvability. Directions in recursion theory*, pp. 1–60. Cambridge University Press, Cambridge (1996)
- [ASM97] Ambos-Spies, K., Mayordomo, E.: Resource-bounded measure and randomness. In: Sorbi, A. (ed.) *Complexity, Logic and Recursion Theory*, pp. 1–47. Dekker, New York (1997)
- [ASR97] Ambos-Spies, K., Reimann, J.: Effective Baire category concepts. In: *Proc. Sixth Asian Logic Conference 1996*, pp. 13–29. Singapore University Press (1997)
- [Blu67] Blum, M.: A machine-independent theory of the complexity of recursive functions. *Journal of the ACM* 14(2), 322–336 (1967)
- [Bor72] Borodin, A.: Computational complexity and the existence of complexity gaps. *Journal of the ACM* 19(1), 158–174 (1972)
- [CZ96] Calude, C., Zimand, M.: Effective category and measure in abstract complexity theory. *Theoretical Computer Science* 154(2), 307–327 (1996)
- [Lu92] Lutz, J.: Almost everywhere high nonuniform complexity. *Journal of Computer and System Sciences* 44, 220–258 (1992)
- [May94] Mayordomo, E.: Almost every set in exponential time is P-bi-immune. *Theoretical Computer Science* 136, 487–506 (1994)
- [Me73] Mehlhorn, K.: On the size of sets of computable functions. In: *Proceedings of the 14th IEEE Symposium on Switching and Automata Theory*, pp. 190–196 (1973)
- [MF72] Meyer, A.R., Fischer, P.C.: Computational speed-up by effective operators. *Journal of Symbolic Logic* 37(1), 55–68 (1972)
- [Sch73] Schnorr, C.: Process complexity and effective random tests. *Journal of Computer and System Sciences* 7, 376–388 (1973)
- [Tra67] Trakhtenbrot, B.A.: *Complexity of algorithms and computations*. Course Notes, Novosibirsk (in Russian) (1967)
- [Zim06] Zimand, M.: *Computational Complexity: A Quantitative Perspective*. Elsevier, Amsterdam (2006)

Computing the Exact Distribution Function of the Stochastic Longest Path Length in a DAG

Ei Ando¹, Hirotaka Ono^{1,2}, Kunihiro Sadakane¹, and Masafumi Yamashita^{1,2}

¹ Department of Computer Science and Communication Engineering,
Graduate School of Information Science and Electrical Engineering,
Kyushu University

² Institute of Systems, Information Technologies and Nanotechnologies

Abstract. Consider the longest path problem for directed acyclic graphs (DAGs), where a mutually independent random variable is associated with each of the edges as its edge length. Given a DAG G and any distributions that the random variables obey, let $F_{\text{MAX}}(x)$ be the distribution function of the longest path length. We first represent $F_{\text{MAX}}(x)$ by a repeated integral that involves $n - 1$ integrals, where n is the order of G . We next present an algorithm to symbolically execute the repeated integral, provided that the random variables obey the standard exponential distribution. Although there can be $\Omega(2^n)$ paths in G , its running time is bounded by a polynomial in n , provided that k , the cardinality of the maximum anti-chain of the incidence graph of G , is bounded by a constant. We finally propose an algorithm that takes x and $\epsilon > 0$ as inputs and approximates the value of repeated integral of x , assuming that the edge length distributions satisfy the following three natural conditions: (1) The length of each edge $(v_i, v_j) \in E$ is non-negative, (2) the Taylor series of its distribution function $F_{ij}(x)$ converges to $F_{ij}(x)$, and (3) there is a constant σ that satisfies $\sigma^p \leq |(\frac{d}{dx})^p F_{ij}(x)|$ for any non-negative integer p . It runs in polynomial time in n , and its error is bounded by ϵ , when x , ϵ , σ and k can be regarded as constants.

1 Introduction

Let $G = (V, E)$ be a directed acyclic graph (DAG), where V and E are the sets of n vertices and m edges, respectively. Each edge (v_i, v_j) is associated with a random variable X_{ij} representing its length. Although the longest path problem for DAGs is solvable in linear time when edge lengths are constant values, the same problem with stochastic edge lengths is formidable. Actually, there are at least two different problem formulations; to find a path that has the highest probability of being the longest [12], or to compute the distribution function $F_{\text{MAX}}(x)$ of the longest path length [1, 2, 3, 4, 5, 7, 8, 10, 11]. In this paper, we adopt the second formulation.

The longest path problem in G with uncertain edge lengths is known as the classic problems such as Program Evaluation and Review Technique (PERT) [6] or Critical Path Planning (CPP) [9]. In these problems, the lower and the upper

bounds of the edge lengths (the activity duration) are given as static values, and their goal is to obtain the lower and upper bounds on the longest path length in G , the duration of the whole project. However, we assume, in this paper, that edge lengths are random variables; we are not to determine the edge lengths but to cope with the resulting edge lengths that realize with some probability.

Delay analysis of logical circuits is a killer application of this problem, and besides Monte Carlo simulations, many heuristic approximation algorithms have been proposed so far (see e.g., [3,5,7]). They run fast but their general drawback is that they do not have a theoretical approximation guarantee. To theoretically guarantee an approximation ratio, some authors of this paper proposed an algorithm to construct a primitive function that approximates $F_{\text{MAX}}(x)$ [12].

Computing the exact distribution function has also a long research history. Martin [11] proposed a series-parallel reduction based method, assuming that each edge length obeys a polynomial distribution. Kulkarni and Adlakhia [10] proposed an algorithm that is based on the analysis of continuous time Markov chain. Both algorithms unfortunately take an exponential time with respect to the graph size. Indeed, when edge lengths obey discrete distributions, the problem is #P-complete [8], and is NP-hard even for the series-parallel graphs [4].

We first show that $F_{\text{MAX}}(x)$ is represented by a repeated integral that involves $n - 1$ integrals, for any instance of the problem. The problem of computing $F_{\text{MAX}}(x)$ for any x is thus reducible to the problem of evaluating the repeated integral for x . The evaluation of the repeated integral is possible by making use of standard numerical methods at the expense of accuracy and time.

In this paper, we pursue the possibility of exact computation using the repeated integral. That only $n - 1$ integrals are involved might give us a chance to symbolically compute it in polynomial in n , although there can be $\Omega(2^n)$ paths in G (and the above NP-hardness results essentially suggest that any algorithm would need to evaluate each of the $\Omega(2^n)$ paths).

Assuming that the random variables obey the standard exponential distribution, we show that there is an algorithm to transform the repeated integral into a product of primitive functions. It runs in polynomial time in n , provided that k , the cardinality of the maximum anti-chain of the incidence graph of G , is bounded by a constant.

We (of course) cannot present a polynomial time algorithm that works for any distribution of edge length. Naive numerical methods to approximate the repeated integral, on the other hand, need sufficiently long computation time and do not guarantee approximation performance. We thus assume that the distribution function F_{ij} associated with any edge (v_i, v_j) satisfies the following three natural conditions: (1) The length of each edge is positive (i.e., $F_{ij}(x) = 0$ for $x \leq 0$), (2) there is a constant σ that satisfies $\left| \left(\frac{d}{dx} \right)^p F_{ij}(x) \right| \leq \sigma^p$ for any non-negative integer p , and (3) the Taylor series of $F_{ij}(x)$ converges to $F_{ij}(x)$, and then present, for any $\epsilon > 0$, an approximation algorithm that evaluates $F_{\text{MAX}}(x)$ (i.e., the repeated integral) with an error less than ϵ . It runs in polynomial time in n , when x , ϵ , σ and k can be regarded as constants.

This paper is organized as follows: After giving basic definitions and formulas in Section 2, we derive the repeated integral form of $F_{\text{MAX}}(x)$ in Section 3. Section 4 is devoted to the first case in which an exact formula is derived assuming the standard exponential distribution, and Section 5 proposes an approximation algorithm for the second case. Section 6 concludes this paper.

2 Preliminaries

Let $G = (V, E)$ be a directed acyclic graph with vertex set $V = \{v_1, v_2, \dots, v_n\}$ and directed edge set $E \subseteq V \times V$ of m edges. We assume that each edge $(v_i, v_j) \in E$ is associated with its length X_{ij} that is a random variable. A *source* (resp. *terminal*) of G is a vertex in V such that its in-degree (resp. out-degree) is 0. We define the (*directed*) *incidence graph* of $G = (V, E)$ as a directed graph G' with vertex set $V' = V \cup E$ and edge set $E' = \{(v_i, e), (e, v_j) | e = (v_i, v_j) \in E\} \subseteq (V \times E) \cup (E \times V)$. We denote the incidence graph of G by $L(G)$. A subset A of V is called an *antichain* of G if each $v_a \in A$ is not reachable from any other vertex $v_b \in A$. If $(v_i, v_j) \in E$, two vertices v_i and v_j are *neighbors* to each other, v_i is a *parent* of v_j , and v_j is a *child* of v_i . By $N(W)$ we denote the set of all neighbors of vertices in W . Let \mathcal{P} be the set of all source-terminal paths. The longest path length X_{MAX} of G is given as $X_{\text{MAX}} = \max_{\pi \in \mathcal{P}} \left\{ \sum_{(v_i, v_j) \in \pi} X_{ij} \right\}$.

Let X be a random variable. The probability $P(X \leq x)$ is called the (*cumulative*) *distribution function* of X . The *density function* of X is the derivative of $P(X \leq x)$ with respect to x . We say X *obeys the standard exponential distribution* if the distribution function $P(X \leq x)$ is given by $P(X \leq x) = 1 - \exp(-x)$ if $x \geq 0$ and $P(X \leq x) = 0$ if $x < 0$.

Let X_1 and X_2 be two mutually independent random variables. Let $f_1(x)$ and $f_2(x)$ be the density functions of X_1 and X_2 , respectively. The sum $X_1 + X_2$ is also a random variable whose distribution function is given as

$$P(X_1 + X_2 \leq x) = \int_{\mathbf{R}} P(X_1 + t \leq x | X_2 = t) f_2(t) dt = \int_{\mathbf{R}} F_1(x - t) f_2(t) dt, \quad (1)$$

where $F_1(x)$ and $F_2(x)$ are the distribution functions of X_1 and X_2 , respectively. The distribution function of $\max\{X_1, X_2\}$ is given as

$$P(\max\{X_1, X_2\} \leq x) = P(X_1 \leq x \wedge X_2 \leq x) = F_1(x)F_2(x).$$

3 Repeated Integral Representation of $F_{\text{MAX}}(x)$

In this section, we show that the distribution function $F_{\text{MAX}}(x)$ of the longest path length is represented by a repeated integral that involves $n - 1$ integrals. By definition,

$$F_{\text{MAX}}(x) = P(X_{\text{MAX}} \leq x) = P\left(\bigwedge_{\pi \in \mathcal{P}} \left(\sum_{e \in \pi} X_e \leq x\right)\right). \quad (2)$$

Although this formula is compact, this fact does not directly implies an efficient computability, since it would take into account all source-terminal paths in G , which can be as many as $\Omega(2^n)$. Next theorem shows that $F_{\text{MAX}}(x)$ is represented by a repeated integral that involves $n - 1$ integrals. Thus $F_{\text{MAX}}(x)$ can be computed by executing only $n - 1$ integrals, which may be dramatically more efficient than the calculation of Eq. (2). Let $H(x)$ be a function that satisfies $H(x) = 1$ if $x \geq 0$ and $H(x) = 0$ if $x < 0$. Let $\mathbf{1}(x)$ be a constant function that maps every x to 1. Note that if $P(X \leq x) = H(x)$ (resp. $P(X \leq x) = \mathbf{1}(x)$) for any x , X is always equal to 0 (resp. $-\infty$).

Theorem 1. *Let $G = (V, E)$ is a DAG. Without loss of generality, we assume that $V = \{v_1, v_2, \dots, v_n\}$ is topologically ordered. For any edge $(v_i, v_j) \in E$, let $F_{ij}(x)$ be the distribution function that X_{ij} obeys. We associate a function $F_{ij}(x)$ with each edge $(v_i, v_j) \notin E$ as follows: If (v_i, v_j) connects two sources or two terminals, then $F_{ij}(x) = H(x)$; otherwise, $F_{ij}(x) = \mathbf{1}(x)$. Then the distribution function $F_{\text{MAX}}(x)$ is given as*

$$P(X_{\text{MAX}} \leq x) = \int_{\mathbf{R}^{n-1}} H(x - z_1) \prod_{1 \leq i \leq n-1} \left(\frac{d}{dz_i} \prod_{i+1 \leq j \leq n} F_{ij}(z_i - z_j) \right) dz_i. \quad (3)$$

Proof. Given a DAG $G = (V, E)$, we first add edges as many as possible in such a way that the added edges do not change the topological order. This yields a complete graph with acyclic orientations, which is denoted by $\vec{K}_n = (V, E_K)$, where $E_K = \{(v_i, v_j) \mid 1 \leq i < j \leq n\}$. Notice that \vec{K}_n has a unique source v_0 and a unique terminal v_n . For each of the edge $(v_i, v_j) \in E_K$, we associate a random variable X_{ij} whose distribution function is shown in the statement of this theorem. Since the lengths of the edges in $E_K \setminus E$ do not have any contribution to the longest path length, $F_{\text{MAX}}(x)$ is exactly the same for G and \vec{K}_n . In what follows, we assume $G = \vec{K}_n$.

Define several notations: $\mathcal{P}(i, j)$ is the set of all paths from v_i to v_j , $\mathcal{P}_k(i, j)$ is the set of all v_i - v_j paths that *do not* pass a vertex in $U_k = \{v_k, v_{k+1}, \dots, v_{n-1}\}$, and $Z_{n-1} = X_{n-1, n}$ is the longest path length from v_{n-1} to the unique terminal v_n of \vec{K}_n . Since $Z_{n-1} = X_{n-1, n}$ and X_{ij} 's are mutually independent, we have

$$P(X_{\text{MAX}} \leq x) = P \left(\bigwedge_{\pi \in \mathcal{P}_{n-1}(1, n)} \left(\sum_{(v_i, v_j) \in \pi} X_{ij} \leq x \right) \wedge \bigwedge_{\pi \in \mathcal{P}(1, n-1)} \left(\sum_{(v_i, v_j) \in \pi} X_{ij} + Z_{n-1} \leq x \right) \right). \quad (4)$$

Let $G_{n-1}(x) = F_{n-1, n}(x)$ and $g_{n-1}(x)$ be the distribution function of Z_{n-1} and its density function, respectively. Since $dG_{n-1}(z_{n-1}) = g_{n-1}(z_{n-1})dz_{n-1}$, like the derivation of Eq. (1), by introducing an integral, the right-hand side of Eq. (4) is represented as

$$\int_{\mathbf{R}} P \left(\underbrace{\bigwedge_{\pi \in \mathcal{P}_{n-1}(1,n)} \left(\sum_{(v_i, v_j) \in \pi} X_{ij} \leq x \right)}_{(A)} \wedge \underbrace{\bigwedge_{\pi \in \mathcal{P}(1,n-1)} \left(\sum_{(v_i, v_j) \in \pi} X_{ij} + z_{n-1} \leq x \right)}_{(B)} \right) dG_{n-1}(z_{n-1}). \quad (5)$$

We calculate the contribution of each edge by repeating the transformation of representing (and replacing) the contribution by an integral. For $Z_k = \max_{k+1 \leq l \leq n} \{X_{kl} + z_l\}$, we divide the paths from v_1 to v_n into two groups according to whether they pass v_k . We then introduce one more integral to aggregate the probability that Z_k takes a constant value z_k , which is the dummy variable of an integral. Note that $z_n = 0$ by definition. Now for each of $Z_{n-1}, Z_{n-2}, \dots, Z_2$, an integral has been introduced with respect to z_i , and (4) is transformed into

$$\int_{\mathbf{R}^{n-2}} P \left(\bigwedge_{2 \leq l \leq n} \bigwedge_{\pi \in \mathcal{P}_2(1,l)} \left(\sum_{(v_i, v_j) \in \pi} X_{ij} + z_l \leq x \right) \wedge \bigwedge_{\pi \in \mathcal{P}(1,2)} \left(\sum_{(v_i, v_j) \in \pi} X_{ij} + z_2 \leq x \right) \right) \prod_{1 \leq i \leq n-1} G_i(z_i, \dots, z_{n-1}) dz_i, \quad (6)$$

where $G_i(z_i, \dots, z_{n-1}) = \frac{d}{dz_i} \prod_{i+1 \leq j \leq n} P(X_{ij} + z_j \leq z_i) = \frac{d}{dz_i} \prod_{i+1 \leq j \leq n} F_{ij}(z_i - z_j)$.

By definition, $\mathcal{P}(1,2) = \{(v_1, v_2)\}$ and $\mathcal{P}_2(1,l) = \{(v_1, v_l)\}$. Hence (6) is equal to

$$\int_{\mathbf{R}^{n-2}} \prod_{2 \leq l \leq n} F_{1l}(x - z_l) \prod_{2 \leq i \leq n-1} \left(\frac{d}{dz_i} \prod_{i+1 \leq j \leq n} F_{ij}(z_i - z_j) \right) dz_i, \quad (7)$$

which implies the theorem. □

It is worth noting that Theorem 1 is applicable, even if the length c_{ij} of each edge (v_i, v_j) is a constant value. The step function $H(x - c_{ij})$ is the distribution function of X_{ij} . Let d_i be the (definite) longest path length from v_i to v_n . Then the step function $F_{\text{MAX}}(x) = H(x - d_1)$ is obtained by Theorem 1.

Let $Q_1(z_1, z_2, \dots, z_{n-1}; x) = H(x - z_1)$ and

$$Q_{l+1}(z_{l+1}, \dots, z_{n-1}; x) = \int_{\mathbf{R}} Q_l(z_l, z_{l+1}, \dots, z_{n-1}; x) G_l(z_l, z_{l+1}, \dots, z_{n-1}) dz_l.$$

Theorem 1 states that we can calculate $Q_n(x) = F_{\text{MAX}}(x)$ by repeating integrals. In the following, we call dummy variable z_i the corresponding variable of v_i .

¹ Notice that we define Z_k after $z_{k+1}, z_{k+2}, \dots, z_{n-1}$; if we define Y_i as the length of the longest path from v_k to v_n at a time, then Y_i 's are dependent on each other, which implies that the above proof cannot be applied to Y_i 's.

4 Exact Computation of the Repeated Integral

This section considers the case in which the edge lengths are given by mutually independent random variables that obey the standard exponential distribution function. We present an algorithm to compute each of Q_1, Q_2, \dots, Q_n symbolically in this order by expanding the integrand into a sum of products before calculating each integral. Let k be the cardinality of the maximum anti-chain of $L(G)$. By bounding the number of different terms that can appear during the symbolic calculation, we show that its running time is a polynomial in the size of G , if k is bounded by a constant.

Proposition 1. *Let $W_i = \{v_j \mid 1 \leq j \leq i\}$. If $v_l \in W_i \setminus \{v_i\}$ or $(u, v_l) \notin E$ for any $u \in W_i$, then $Q_i(z_i, \dots, z_{n-1}; x)$ does not depend on z_l .*

Proof. Since z_l is a dummy variable of an integral if $l < i$, it is obvious that z_l never shows up in $Q_i(z_i, \dots, z_{n-1}; x)$ after the integrals are computed.

Suppose otherwise that $l > i$. Then $v_l \notin N(W_i) \setminus W_i$. By Theorem [11](#), $G_i(z_i, \dots, z_{n-1})$ does not depend on z_l . □

Let $m = |E|$, $n = |V|$ and V_i be the set of children of v_i .

Theorem 2. *Let $G = (V, E)$ be a DAG such that the cardinality of the maximum anti-chain of $L(G)$ is at most k . Assume that each random variable X_{ij} , which represents the length of edge (v_i, v_j) , obeys the standard exponential distribution. Then the distributed function $F_{\text{MAX}}(x)$ of the longest path length in G is computable in $O((k + 1)!n^{k+2}(2m + 1)^{k+1})$ time.*

Proof. We first show how we calculate $Q_{i+1}(z_{i+1}, \dots, z_{n-1}; x)$ from $Q_i(z_i, \dots, z_{n-1}; x)$ by symbolically executing the integral with respect to z_i . For example, we have

$$Q_3(z_3, \dots, z_{n-1}; x) = \int_{\mathbf{R}} Q_2(z_2, \dots, z_{n-1}; x) G_2(z_2, \dots, z_{n-1}) dz_2, \quad (8)$$

where $Q_2(z_2, \dots, z_{n-1}; x) = \prod_{v_j \in V_1} F_{1j}(x - z_j)$. Since $H(x) = 0$ for $x < 0$, $Q_3(z_3, \dots, z_{n-1}; x) = 0$ if $x < 0$. Since $H(x) = 1$ if $x \geq 0$,

$$Q_3(z_3, \dots, z_{n-1}; x) = \int_a^b \prod_{v_j \in V_1} (1 - \exp(-(x - z_j))) \frac{d}{dz_2} \prod_{v_l \in V_2} (1 - \exp(-(z_2 - z_l))) dz_2, \quad (9)$$

where $a = \max_{v_l \in V_2} z_l$ and $b = x$, since otherwise the contribution to the integral becomes 0 because of the effect of H . Since each of z_l 's can take the maximum, at most $|V_2|$ different formulas appear, corresponding to different $a = z_l$, as possible results of $Q_3(z_3, \dots, z_{n-1}; x)$. Once a is fixed to a z_l , executing symbolic integration of the right-hand side of Eq. [\(9\)](#) is easy, since possible terms appearing in the integrand have a form of $c_1 \exp(-c_2 z_2)$ for some constant c_1 and c_2 . In general, we obtain $Q_{i+1}(z_{i+1}, \dots, z_{n-1}; x)$ from $Q_i(z_i, \dots, z_{n-1}; x)$ in this way.

To estimate the time complexity of the algorithm, let us estimate the number of terms that are possible to appear in the execution. By Proposition [1](#), the number of variables appeared in $Q_i(z_i, \dots, z_{n-1}; x)$ is at most $k + 1$ for any i . As explained, to obtain $Q_3(z_3, \dots, z_{n-1}; x)$, we need to consider at most $k + 1$ different cases corresponding to different $a = z_l$. It is easy to see that to obtain $Q_4(z_4, \dots, z_{n-1}; x)$, for each of the cases for Q_3 , we need to consider at most k different cases. Although this leads to that there may be $O(k^i)$ cases for Q_i in general, the number of variables on which Q_i depends is at most $k + 1$ by Proposition [1](#), which implies that, in general in Q_i , there can be no more than $(k + 1)!$ distinct cases. To complete the proof, we show that at most $n^{k+1}(2m + 1)^{k+1}$ terms are possible to appear, for each of at most $(k + 1)!$ cases.

Let us consider the number of the terms in the integrand in each case. Since it is easy to see that each term is a product of x^{α_0} , $z_j^{\alpha_j}$, $\exp(\beta_0 x)$ and $\exp(\beta_j z_j)$, where α_j 's and β_j 's are integers, we bound the number of terms by the number of possible terms. By the form of Theorem [1](#) we can see that the maximum degrees of z_j 's and x that appear in the terms in $Q_i(z_i, \dots, z_{n-1}; x)G_i(z_i, \dots, z_{n-1})$ of each case can only increase by one in one integral and hence α_j 's are non-negative integer and less than n . Similarly, we can also see that the degrees β_j 's of $\exp(z_j)$'s and $\exp(x)$ can only increase or decrease by one in a multiplication of two distribution functions and hence β_j 's are integers between $-m$ and m . Therefore, the integrand in each cases consists of at most $n^{k+1}(2m + 1)^{k+1}$ terms, which amounts to that the calculation of each $Q_{i+1}(z_{i+1}, \dots, z_{n-1}; x)$ from $Q_i(z_i, \dots, z_{n-1}; x)$ takes $O((k + 1)!n^{k+1}(2m + 1)^{k+1})$ time. \square

Corollary 1. *A closed form of $F_{\text{MAX}}(x)$ consisting of primitive functions is obtained in polynomial time if k is bounded by a constant, provided that the edge lengths obey the standard exponential distribution.*

5 Approximation of the Repeated Integral

In this section, we assume that the cardinality of the incidence graph $L(G)$ of a given DAG G is bounded by a constant k . We show that $F_{\text{MAX}}(x)$ can be approximately calculated in polynomial time in n , if the length of each edge $(v_i, v_j) \in E$ is non-negative and the Taylor series of its distribution function $F_{ij}(x)$ converges to $F_{ij}(x)$. Here by ‘‘Taylor polynomial of a function f ’’, we mean the Taylor polynomial generated by f at the origin.

We must be careful for the order of computing the Taylor polynomial of the repeated integral that is shown in Theorem [1](#). Let p be the order of the Taylor polynomial. The most intuitive idea is computing the Taylor polynomial of the whole integrand of [\(3\)](#) in Theorem [1](#) treating it as a function of n variables (i.e., $x, z_1, z_2, \dots, z_{n-1}$). However, this intuitive way of computing the Taylor polynomial is not efficient for obtaining the value of $F_{\text{MAX}}(x)$ with an error less than ϵ ; the running time may be more than exponential with respect to the size of G even if k is a constant.

In order to lower the running time, we approximate $Q_i(z_i, \dots, z_{n-1}; x)$ by $A_i^p(z_i, \dots, z_{n-1}; x)$ that is computed by the following: (1) $A_2^p(z_2, \dots, z_{n-1}; x)$ is

the Taylor polynomial of order p generated by $Q_2(z_2, \dots, z_{n-1}; x) = \prod_{v_j \in V_1} F_{1j}(x - z_j)$, and (2) $A_i^p(z_i, \dots, z_{n-1}; x)$ is the Taylor polynomial of order p generated by

$$\int_{\mathbf{R}} A_{i-1}^p(z_{i-1}, \dots, z_{n-1}; x) G_{i-1}(z_{i-1}, \dots, z_{n-1}) dz_{i-1}. \tag{10}$$

This integral can be calculated using integration by parts, which yields a sum of products of polynomials and some anti-derivatives of $G_{i-1}(z_{i-1}, \dots, z_{n-1}) = \prod_{i < j \leq n} F_{i-1,j}(z_{i-1} - z_j)$. The procedure (2) can be repeated for $i = 3, 4, \dots, n$.

Since all edge lengths are non-negative by assumption, the anti-derivative of $G_{i-1}(z_{i-1}, \dots, z_{n-1})$ of positive order is equal to 0 at the origin $x = z_i = z_{i+1} = \dots = z_{n-1} = 0$, which allows us to compute $A_i(z_i, \dots, z_{n-1}; x)$ without knowing the form of the anti-derivatives of $G_{i-1}(z_{i-1}, \dots, z_n)$.

In the next theorem, we show that the time to compute $A_n^p(x)$ where p is large enough to keep the error less than ϵ is polynomial of the size of G , assuming that x , ϵ and the maximum size k of an antichain in $L(G)$ is a constant. We also assume the existence of a constant σ , that satisfies $\sigma^p \geq \left| \left(\frac{d}{dx} \right)^p F_{ij}(x) \right|$ for any non-negative integer p and any edge $(v_i, v_j) \in E$.

Notice that σ must be bounded by a constant for the assumption that x is bounded by a constant. If there is an algorithm A that gives the value of $F_{\text{MAX}}(x)$ in the same time regardless of σ , we can consider the ‘‘compressed edge length’’ $X'_e = X_e/s$, where X_e is the length of e and $s \geq 1$. Then we can define the ‘‘compressed’’ distribution function $F'_{\text{MAX}}(x) = P(\bigwedge_{\pi \in \mathcal{P}} (\sum_{e \in \pi} X'_e \leq x))$ of the longest path length. Since A gives the value of $F'_{\text{MAX}}(x) = F_{\text{MAX}}(sx)$ for any s in the same running time, A can be used for obtaining the value of $F_{\text{MAX}}(x)$ for arbitrary x . Therefore, it is essential to bound σ by a constant as well as x .

Theorem 3. *Let $G = (V, E)$ be a DAG and assume that the cardinality of the anti-chain of its incidence graph $L(G)$ is at most k . Let $F_{ij}(x)$ be the distribution function of the length of an edge (v_i, v_j) that is defined in Theorem 2. Let σ be a value such that $\sigma^p \geq \left| \left(\frac{d}{dx} \right)^p (F_{ij}(x)) \right|$ for any non-negative integer p and any edge $(i, j) \in E$. We further assume that the Taylor series of $F_{ij}(x)$ converges to $F_{ij}(x)$ itself and that the time complexity of computing the p -th derivative of $F_{ij}(x)$ is $O(\exp(p))$. Then $A_n^p(x)$ such that $|A_n^p(x) - F_{\text{MAX}}(x)| \leq \epsilon$ holds is calculated in time $O((k + 1)!(p + 1)^k k^{p+1} \exp(p))$, where $p = O(k^2 x \sigma + \ln n + \ln 1/\epsilon)$.*

Proof. By the similar argument in the previous section, it can be shown that the time to compute $A_n^p(x)$ is $O((k + 1)!n(p + 1)^k k^{p+1} \exp(p))$.

Now we concentrate on proving that $p = O(k^2 x \sigma + \ln n + \ln 1/\epsilon)$ is sufficient for $|A_n^p(x) - F_{\text{MAX}}(x)| \leq \epsilon$. For each edge (v_i, v_j) , we consider a random variable $X'_{ij} = X_{ij}/(k\sigma)$. Consider that the length of each edge (v_i, v_j) is X'_{ij} instead of X_{ij} . Let $F'_{\text{MAX}}(x)$ be the corresponding distribution function. Since $F_{\text{MAX}}(x) = F'_{\text{MAX}}(kx\sigma)$, we consider the normalized edge length X'_{ij} and the normalized distribution function $F'_{\text{MAX}}(x)$ instead of X_{ij} and $F_{\text{MAX}}(x)$ in the following. For the simplicity, we give the proof for the case $\sigma = 1/k$. The proof for the general σ can be given by replacing x in the following by $kx\sigma$.

Let ϵ_i be the difference between $A_i^p(z_i, \dots, z_n; x)$ and $Q_i(z_i, \dots, z_{n-1}; x)$. We first bound the error of $A_2^p(z_2, \dots, z_{n-1}; x)$. Since the distribution functions of edge lengths are normalized by the above manner, it can be shown that

$$|\epsilon_2| \leq \frac{(xk)^{p+1}}{(p+1)!}, \tag{11}$$

by the evaluation of the Taylor polynomials in [13] and Proposition 11

Let us bound the error that is created when $A_{i+1}^p(z_{i+1}, \dots, z_{n-1}; x)$ is computed with respect to z_i . By definition, we have

$$\epsilon_{i+1} = A_{i+1}^p(z_{i+1}, \dots, z_{n-1}; x) - \int_{\mathbf{R}} (A_i^p(z_i, \dots, z_{n-1}; x) - \epsilon_i) G_i(z_i, \dots, z_{n-1}) dz_i. \tag{12}$$

By the similar argument as in obtaining (11), we have

$$|\epsilon_{i+1}| \leq \frac{(kx)^{p+1}}{(p+1)!} + \int_{\mathbf{R}} |\epsilon_i| \frac{d}{dz_i} \prod_{i+1 \leq j \leq n-1} F_{ij}(z_i - z_j) dz_i = \frac{(kx)^{p+1}}{(p+1)!} + |\epsilon_i|. \tag{13}$$

This leads to $|\epsilon_n| \leq (n-1) \frac{(kx)^{p+1}}{(p+1)!}$ by (11) and (13) for $i = 2, 3, \dots, n$.

If $kx \leq 1$, the error ϵ_n converges to 0 very quickly. If $kx > 1$, $p = O(\ln(n-1) + kx + \ln 1/\epsilon)$ is sufficient to have $|\epsilon_n| = |A_n^p(x) - Q_n(x)|$ less than ϵ . \square

We immediately obtain the following corollary.

Corollary 2. *If x, ϵ, k and σ are constants, the proposed algorithm computes the value of $F_{\text{MAX}}(x)$ within error ϵ in a polynomial time of the size of G .*

6 Conclusion

In this paper, we have investigated the longest path problem for DAGs, where the edge lengths are mutually independent random variables. We have shown that the distribution function $F_{\text{MAX}}(x)$ of the longest path length is given as a form of repeated integral that involves $n-1$ integrals, where n is the order of DAG G . We can thus approximately evaluate $F_{\text{MAX}}(x)$ for any fixed x by applying numerical methods to the form, at the expense of accuracy and time.

We however suggest that an important application of the repeated integral is in symbolic computation of $F_{\text{MAX}}(x)$. Because only $n-1$ integrals are involved, it may give us a chance to symbolically compute it in polynomial in n , although there are $\Omega(2^n)$ paths in G . In fact, we have shown that a representation of $F_{\text{MAX}}(x)$ by a combination of primitive functions is obtained in $O((k+1)!n^{k+2}(2m+1)^{k+1})$ time, provided that the edge lengths obey the standard exponential distribution, where k is the maximum anti-chain cardinality of the incidence graph $L(G)$. Recall that the problem is NP-hard even for series-parallel graphs when the edge lengths obey discrete distributions. A natural open

question is thus to find another class of distribution functions for which there is a polynomial algorithm to symbolically execute the repeated integral.

We have proposed an approximation algorithm to compute $F_{\text{MAX}}(x)$ with error smaller than ϵ for any given x and ϵ , assuming that the distribution function $F_e(x)$ of each $e \in E$ satisfy the following three natural conditions; 1) $F_e(x) = 0$ for $x \leq 0$, 2) the Taylor series of $F_e(x)$ converges to $F_e(x)$, and 3) for any non-negative integer p , there is a constant σ satisfying $\sigma^p \geq \left| \left(\frac{d}{dx} \right)^p F_e(x) \right|$. The running time is a polynomial in n , when k , x , ϵ and σ can be regarded as constants. In other words, we showed the existence of PTAS for computing $F_{\text{MAX}}(x)$ for fixed x , DAGs with constant k , and the edge lengths that satisfy the above three conditions. A natural open question is to propose an approximation algorithm that works for any distribution of edge length.

References

1. Ando, E., Nakata, T., Yamashita, M.: Approximating the longest path length of a stochastic DAG by a normal distribution in linear time. *Journal of Discrete Algorithms* (2009), doi:10.1016/j.jda.2009.01.001
2. Ando, E., Ono, H., Sadakane, K., Yamashita, M.: A Generic Algorithm for Approximately Solving Stochastic Graph Optimization Problems (submitted for publication)
3. Ando, E., Yamashita, M., Nakata, T., Matsunaga, Y.: The Statistical Longest Path Problem and Its Application to Delay Analysis of Logical Circuits. In: *Proc. TAU*, pp. 134–139 (2002)
4. Ball, M.O., Colbourn, C.J., Proban, J.S.: Network Reliability. In: Ball, M.O., Magranti, T.L., Monma, C.L., Nemhauser, G.L. (eds.) *Handbooks in Operations Research and Management Science. Network Models*, vol. 7, pp. 673–762. Elsevier Science B. V., Amsterdam (1995)
5. Berkelaar, M.: Statistical delay calculation, a linear time method. In: *Proceedings of the International Workshop on Timing Analysis (TAU 1997)*, pp. 15–24 (1997)
6. Clark, C.E.: The PERT model for the distribution of an activity time. *Operations Research* 10, 405–406 (1962)
7. Hashimoto, M., Onodera, H.: A performance optimization method by gate sizing using statistical static timing analysis. *IEICE Trans. Fundamentals* E83-A(12), 2558–2568 (2000)
8. Hagstrom, J.N.: Computational Complexity of PERT Problems. *Networks* 18, 139–147 (1988)
9. Kelley Jr., J.E.: Critical-path planning and scheduling: Mathematical basis. *Operations Research* 10, 912–915 (1962)
10. Kulkarni, V.G., Adlakha, V.G.: Markov and Markov-Regenerative PERT Networks. *Operations Research* 34, 769–781 (1986)
11. Martin, J.J.: Distribution of the time through a directed, acyclic network. *Operations Research* 13, 46–66 (1965)
12. Nikolova, E.: Stochastic Shortest Paths Via Quasi-convex Maximization. In: Azar, Y., Erlebach, T. (eds.) *ESA 2006. LNCS*, vol. 4168, pp. 552–563. Springer, Heidelberg (2006)
13. Thomas Jr., G.B.: *Thomas' Calculus International Edition*, pp. 965–1066. Pearson Education, London (2005)

On the Connection between Interval Size Functions and Path Counting^{*}

Evangelos Bampas¹, Andreas-Nikolas Göbel¹, Aris Pagourtzis¹,
and Aris Tentes²

¹ School of Elec. & Comp. Eng., National Technical University of Athens
Polytechnioupoli Zografou, 157 80 Athens, Greece

ebamp@cs.ntua.gr, agob@corelab.ntua.gr, pagour@cs.ntua.gr

² New York University, USA

tentes@cs.nyu.edu

Abstract. We investigate the complexity of hard counting problems that belong to the class #P but have easy decision version; several well-known problems such as #PERFECT MATCHINGS, #DNFSAT share this property. We focus on classes of such problems which emerged through two disparate approaches: one taken by Hemaspaandra *et al.* [1] who defined classes of functions that count the size of intervals of ordered strings, and one followed by Kiayias *et al.* [2] who defined the class TotP, consisting of functions that count the total number of paths of NP computations. We provide inclusion and separation relations between TotP and interval size counting classes, by means of new classes that we define in this work. Our results imply that many known #P-complete problems with easy decision are contained in the classes defined in [1]—but are unlikely to be complete for these classes under certain types of reductions. We also define a new class of interval size functions which strictly contains FP and is strictly contained in TotP under reasonable complexity-theoretic assumptions. We show that this new class contains some hard counting problems.

1 Introduction

Valiant's pioneering work on counting problems associated with NP computations [3] revealed the existence of functions that are quite hard to compute exactly (#P-complete), despite the fact that deciding whether the function value is nonzero is easy (in P). This category contains the problem of evaluating the permanent of a 0-1 matrix (PERMANENT), which is equivalent to counting perfect matchings in bipartite graphs (#PM), the problem of counting satisfying assignments to monotone Boolean formulae in 2-CNF form (#MON2SAT), and many more [4]. A common feature of all these problems is that their #P-completeness property is based on the Cook (poly-time Turing) reduction which blurs structural differences between complexity classes; for example, PERMANENT is also

^{*} Research supported in part by a Basic Research Support Grant (IIEBE 2007) of the National Technical University of Athens.

complete in the Cook sense for the whole counting version of the Polynomial Hierarchy [5,6], but also for subclasses of #P [7]. Hence, #P is not considered to be the most appropriate class to describe the complexity of these problems.

During the last twenty years there has been constant interest for identifying subclasses of #P that contain hard counting problems with easy decision version [1,2,8,9,10,11,12] and may therefore be more adequate to describe their complexity. In this paper we investigate the relation among subclasses of #P defined and studied through two independent lines of research: (a) classes $IF_p^<$ and $IF_t^<$ [1] that consist of functions that count the size of intervals of strings under poly-time decidable partial or total (resp.) orders equipped with efficient adjacency checks, and (b) the class TotP [2] that consists of functions that count the total number of paths of NPTMs, and the class #PE [11] that contains all functions of #P for which telling whether the function value is nonzero is easy (in P). Since it is clear from properties of $IF_p^<$ shown in [1] that $IF_p^< = \#PE$ we turn our focus to the relation between $IF_t^<$ and TotP, which are subclasses of $IF_p^<$. To this end we define new interval size function classes by replacing efficient adjacency checks with other suitable feasibility constraints. Our results can be summarized as follows (see also Figure 1):

- TotP is equal to IF_t^{LN} , that is, to the class of interval size functions defined on total p-orders with efficiently computable lexicographically nearest function.
- IF_t^{LN} , hence also TotP, is contained in $IF_t^<$. The inclusion is strict unless $P = UP \cap coUP$. This, among others, implies that several problems that lie in TotP are unlikely to be $IF_t^<$ -complete via reductions under which TotP is closed downwards (for example, under Karp reductions); in particular, the class of problems that reduce to #MONSAT by such reductions is strictly contained in $IF_t^<$ unless $P = UP \cap coUP$. This partially answers an open question posed in [1].
- One of our new classes, namely IF_t^{rmed} , lies between FP and $IF_t^{LN} = TotP$. We show that IF_t^{rmed} contains hard counting problems: we define $\#SAT_{+2^n}$, which is #P-complete under Cook reductions, and prove that it lies in IF_t^{rmed} . We also show that any #P function can be obtained by subtracting a function in FP from a function in IF_t^{rmed} . Therefore IF_t^{rmed} is Cook-interreducible with TotP, $IF_t^<$, $IF_p^< = \#PE$, and #P but not Karp-interreducible with any of these classes under reasonable assumptions.

2 Definitions–Preliminaries

In the following we assume a fixed alphabet Σ , conventionally $\Sigma = \{0, 1\}$. The symbol Σ^* denotes the set of all finite strings over the alphabet Σ . The length of a string $x \in \Sigma^*$ is denoted by $|x|$. If S is a set, $\|S\|$ denotes the cardinality of S .

A binary relation over Σ^* is a *partial order* if it is reflexive, antisymmetric, and transitive. A partial order A is a *total order* if for any $x, y \in \Sigma^*$, it holds that $(x, y) \in A$ or $(y, x) \in A$. An order A is called a *p-order* if there exists a bounding polynomial p such that for all $(x, y) \in A$ it holds that $|x| \leq p(|y|)$.

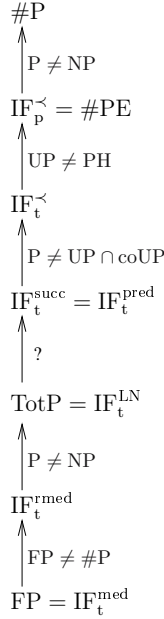


Fig. 1. Inclusions among interval size function classes. Next to each arrow appear the assumptions under which the inclusions are proper; it is open whether $TotP = IF_t^{succ}$ implies an unlikely collapse.

Definition 1 (Notation for orders, cf. [1]). For any order A we will use the following notation:

1. $x \leq_A y$ is equivalent to $(x, y) \in A$,
2. $x <_A y$ is equivalent to $(x \leq_A y \wedge x \not\equiv y)$,
3. $x \prec_A y$ is equivalent to $(x <_A y \wedge \neg \exists z \in \Sigma^*(x <_A z <_A y))$ (we say that x is the predecessor of y , or y is the successor of x),
4. $A_{\prec} \stackrel{\text{def}}{=} \{(x, y) : x \prec_A y\}$, and
5. $(x, y)_A \stackrel{\text{def}}{=} \{z \in \Sigma^* : x <_A z <_A y\}$ ($(x, y)_A$ will be called an interval, even if A is a partial order). We will also use $[x, y]_A$, $[x, y)_A$, and $(x, y]_A$ for the closed, right-open, and left-open intervals respectively.

We will use lex to denote the standard lexicographic order of the strings in Σ^* .

Remark 1. For any p-order A with bounding polynomial p and any $y \in \Sigma^*$, $\|\{x : x \leq_A y\}\| \leq 2^{p(|y|)+1} - 1$. As a corollary, every p-order has a minimal element.

Definition 2 (Notation for total orders). For any total order A we will use the following notation:

1. $\text{succ}_A : \Sigma^* \rightarrow \Sigma^*$ is the successor function for A ,
2. $\text{pred}_A : \Sigma^* \rightarrow \Sigma^*$ is the predecessor function for A (if A contains a bottom element, pred_A is undefined for that element),

3. $\text{med}_A : \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$ is the median function for A , defined recursively as follows:
 - if $y <_A x$ then $\text{med}_A(x, y)$ is undefined,
 - otherwise if $x = y$ or $x <_A y$ then $\text{med}_A(x, y) = y$,
 - otherwise $\text{med}_A(x, y) = \text{med}_A(\text{succ}_A(x), \text{pred}_A(y))$.
4. $\text{LN}_A : \Sigma^* \times \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$ is the lexicographically nearest function for A : $\text{LN}_A(x, y, z)$ is the string $w \in [x, y]_A$ such that w is as close to z as possible in the lexicographic order (breaking ties arbitrarily).
5. $\text{rmed}_A^c : \Sigma^* \times \Sigma^* \rightarrow \Sigma^*$, $c \in (0, \frac{1}{2}]$, is some relaxed median function for A , that satisfies the following properties:
 - if $y <_A x$ then $\text{rmed}_A^c(x, y)$ is undefined,
 - otherwise if $x = y$ or $x <_A y$ then $\text{rmed}_A^c(x, y) = y$,
 - otherwise $\text{rmed}_A^c(x, y)$ is a string $z \in (x, y)_A$ such that $\|[x, z]_A\| \geq \lfloor c \cdot \|[x, y]_A\| \rfloor$ and $\|[z, y]_A\| \geq \lfloor c \cdot \|[x, y]_A\| \rfloor$.

Remark 2. For a total order A , we will say that $\text{rmed}_A \in \text{FP}$ if there is some $c \in (0, \frac{1}{2}]$ such that some relaxed median function $\text{rmed}_A^c \in \text{FP}$. Observe that med_A is a function that satisfies the properties of $\text{rmed}_A^{\frac{1}{2}}$, therefore if $\text{med}_A \in \text{FP}$ then also $\text{rmed}_A \in \text{FP}$.

We say that an order A is *P-decidable* if $A \in \text{P}$, and we say that it has *efficient adjacency checks* if $A_{<} \in \text{P}$. We also say that a function $f \in \text{FP}$ is *FP-computable*.

We say that a function $f : \Sigma^* \rightarrow \mathbb{N}$ is an *interval size function defined on an order A* if there exist *boundary functions* $b, t : \Sigma^* \rightarrow \Sigma^*$ such that for all $x \in \Sigma^*$, $f(x) = \|(b(x), t(x))_A\|$. In the following, we will primarily be concerned with interval size functions defined on P-decidable p-orders via polynomial-time computable boundary functions.

Definition 3 (Hemaspaandra et al. [1])

$\text{IF}_p^<$ ($\text{IF}_t^<$) is the class of interval size functions defined on P-decidable partial (total) p-orders with efficient adjacency checks via polynomial-time computable boundary functions.

Remark 3. Note that in [1], $\text{IF}_p^<$ and $\text{IF}_t^<$ were called IF_p and IF_t , respectively. We will use superscript in order to distinguish these classes from other classes that we will define below.

Furthermore, we will be interested in interval size functions defined on P-decidable p-orders with various other feasibility constraints, apart from $A_{<} \in \text{P}$. We define the following classes:

Definition 4. $\text{IF}_t^{\text{succ}}$ (resp. $\text{IF}_t^{\text{pred}}$, IF_t^{LN} , $\text{IF}_t^{\text{rmed}}$, IF_t^{med}) is the class of interval size functions each of which is defined on some P-decidable total p-order A via polynomial-time computable boundary functions, where in addition $\text{succ}_A \in \text{FP}$ (resp. pred_A , LN_A , rmed_A , $\text{med}_A \in \text{FP}$).

The computational model we are going to use is the Non-deterministic Polynomial-time Turing Machine (NPTM). For an NPTM M we denote with $M(x)$ the computation of M on input x . We say that M is in normal form if we can represent the computation $M(x)$ with a full, complete binary tree of depth exactly $p(|x|)$, where p is the polynomial that bounds the running time of M .

Valiant in [4] defines as $\#P$ the class of all total functions f for which there exists an NPTM M such that for all x , $f(x)$ is the number of accepting paths of $M(x)$.

In [11] the class $\#PE$ is defined as the class of $\#P$ functions with their underlying language in P , where for a function f , its underlying language is defined to be the language $L_f = \{x \mid f(x) > 0\}$. In [2] the class TotP is defined as the class that contains the functions f for which there exists an NPTM M such that for all x , $f(x)$ is the number of the computation paths of the computation of $M(x)$ minus one. The functions of TotP are usually denoted with $\text{tot}_M(x)$, where M is the associated NPTM, and x the input. In [12] TotP is proven to be exactly the closure under Karp (parsimonious) reduction of the set of self-reducible functions of $\#PE$. The results can be summarized by the following chain of inclusions:

$$FP \subseteq \text{TotP} \subseteq \#PE \subseteq \#P ,$$

where all the inclusions are proper unless $P = NP$.

In [1] it is (implicitly) proven that $\#PE = \text{IF}_p^<$, and furthermore that:

$$FP \subseteq \text{IF}_t^< \subseteq \text{IF}_p^< \subseteq \#P .$$

Again the inclusions are proper unless unlikely complexity class collapses occur.

Definition 5. *Polynomial-time reductions between functions:*

- Cook (poly-time Turing): $f \leq_T^p g : f \in \text{FP}^g$.
- Karp (parsimonious): $f \leq_m^p g : \exists h \in \text{FP}, \forall x f(x) = g(h(x))$.

Proposition 1. *Every interval size function class \mathcal{F} that contains functions defined via polynomial-time boundary functions is downward closed under Karp reductions.*

Proof. Consider $f \in \mathcal{F}$ via an arbitrary order A and boundary functions $b, t \in \text{FP}$. That is, for every x , $f(x) = \|(b(x), t(x))_A\|$. Assume also that $g \leq_m^p f$, that is $\exists h \in \text{FP}$ such that $\forall x, g(x) = f(h(x))$. This implies that $g(x) = f(h(x)) = \|(b(h(x)), t(h(x)))_A\|$, therefore $g \in \mathcal{F}$ via the same order A and boundary functions $b' = b \circ h \in \text{FP}$ and $t' = t \circ h \in \text{FP}$. □

3 The *status quo* between TotP and $\text{IF}_t^<$

As we have seen in the previous section, both TotP and $\text{IF}_t^<$ are contained in $\#PE = \text{IF}_p^<$. In this section we will investigate the relationship between these two classes. Namely we will show that $\text{TotP} \subseteq \text{IF}_t^<$, and that the inclusion is proper unless $P = \text{UP} \cap \text{coUP}$.

Theorem 1. $\text{TotP} \subseteq \text{IF}_t^{\text{succ}} \subseteq \text{IF}_t^<$.

Proof (sketch). Intuitively, given a path encoding of a TotP computation tree, it is easy to find the next one. The idea is to map computation path encodings to appropriately ordered strings. The detailed proof will appear in the full version. \square

We now proceed to show that $\text{IF}_t^{\text{succ}}$, and therefore also TotP, is strictly contained in $\text{IF}_t^<$, under the assumption that $\text{P} \neq \text{UP} \cap \text{coUP}$. We need a new definition and a couple of lemmata.

Definition 6. For any constant $k \geq 0$, we define the operator $\mathcal{C}_{>k}$. If \mathcal{F} is any function class, then $\mathcal{C}_{>k} \cdot \mathcal{F}$ defines the following class of languages:

$$\mathcal{C}_{>k} \cdot \mathcal{F} = \{L \mid \exists f \in \mathcal{F} \ \forall x (x \in L \iff f(x) > k)\} .$$

Remark 4. Observe that $\mathcal{C}_{>0}$ coincides with the \exists -operator used by Hemaspaandra et al. in [11], which in turn coincides with the Sig- operator defined by Hempel and Wechsung in [13].

Lemma 1. $\text{UP} \cap \text{coUP} \subseteq \mathcal{C}_{>1} \cdot \text{IF}_t^<$.

Proof. Let $L \in \text{UP} \cap \text{coUP}$, so there is an NPTM M that decides L with the property that, for any input x , M has exactly one decisive path (either accepting or rejecting) and all the other paths output “?”. We assume that M is normalized so that its computation for any input x is a full complete binary tree in which all computation paths have length exactly $p(|x|)$, where p is the polynomial that bounds the running time of M .

We construct an order A that coincides with the lexicographic order of Σ^* , except that for every $x \in \Sigma^*$ the interval between $x0^{p(|x|)+2}$ and $x1^{p(|x|)+2}$ (inclusive) is ordered in the following way:

- First comes $x0^{p(|x|)+2}$,
- if $x \notin L$ next comes $x01z$, where z encodes the unique rejecting path of M on input x , while if $x \in L$ next come $x01z$ and $x10z$, where z encodes the unique accepting path of M on input x ,
- next comes $x110^{p(|x|)}$,
- and last come the rest of the strings of the form xw , where $|w| = p(|x|) + 2$, in the lexicographic order.

It is easy to see that A is a p-order with efficient adjacency checks. We define the boundary functions $b, t \in \text{FP}$: for any $x \in \Sigma^*$, $b(x) = x0^{p(|x|)+2}$ and $t(x) = x110^{p(|x|)}$. It holds that, for any $x \in \Sigma^*$, $\|(b(x), t(x))_A\| > 1$ if and only if $x \in L$. Therefore, $L \in \mathcal{C}_{>1} \cdot \text{IF}_t^<$. \square

Lemma 2. $\mathcal{C}_{>1} \cdot \text{IF}_t^{\text{succ}} = \text{P}$.

Proof. For any $f \in \text{IF}_t^{\text{succ}}$, we can decide in polynomial time whether for a given x , $f(x) > 1$ or not. Just compute $\text{succ}_A(b(x))$ and $\text{succ}_A(\text{succ}_A(b(x)))$,

where A is the underlying total p -order with $\text{succ}_A \in \text{FP}$ and $b, t \in \text{FP}$ the boundary functions for f . If any of the computed strings is equal to $t(x)$ then reject, else accept. Therefore, $\mathcal{C}_{>_I} \cdot \text{IF}_t^{\text{succ}} \subseteq \text{P}$. For the other direction, note that $\text{P} = \mathcal{C}_{>_I} \cdot \text{FP} \subseteq \mathcal{C}_{>_I} \cdot \text{IF}_t^{\text{succ}}$. \square

Theorem 2. *If $\text{IF}_t^{\prec} = \text{IF}_t^{\text{succ}}$, then $\text{P} = \text{UP} \cap \text{coUP}$.*

Proof. Assuming that $\text{IF}_t^{\prec} = \text{IF}_t^{\text{succ}}$, from Lemma 1 and Lemma 2 we get that $\text{UP} \cap \text{coUP} \subseteq \mathcal{C}_{>_I} \cdot \text{IF}_t^{\prec} = \mathcal{C}_{>_I} \cdot \text{IF}_t^{\text{succ}} = \text{P}$. \square

4 TotP as an Interval Size Function Class

In this section we prove that TotP coincides with the class of interval size functions defined on orders with polynomial-time computable *lexicographically nearest* functions. To this end, we will employ two variations of the LN function and show a useful property of them.

Definition 7. *For a p -order A we define the following partial functions:*

1. $\text{LN}_A^+(u, v, x)$ is the lexicographically smallest $y \in [u, v]_A$ such that $x \leq_{\text{lex}} y$.
2. $\text{LN}_A^-(u, v, x)$ is the lexicographically largest $y \in [u, v]_A$ such that $x \geq_{\text{lex}} y$.

Lemma 3. *For a total p -order A , if $\text{LN}_A \in \text{FP}$ then also $\text{LN}_A^+ \in \text{FP}$ and $\text{LN}_A^- \in \text{FP}$.*

Proof. We will prove the claim for LN_A^+ only; the proof for LN_A^- is symmetric. Let p be the bounding polynomial of A . We will compute $\text{LN}_A^+(u, v, x)$. Let $y = \text{LN}_A(u, v, x)$. If $x \leq_{\text{lex}} y$, then $\text{LN}_A^+(u, v, x) = y$. For the rest of the proof, assume that $y <_{\text{lex}} x$ and let $\delta = \|[y, x]_{\text{lex}}\|$.

We compute a sequence of strings $x = x_0 <_{\text{lex}} x_1 <_{\text{lex}} \dots <_{\text{lex}} x_k$ where for all i , $\|[y, x_i]_{\text{lex}}\| = 2^i \cdot \delta$, and k is the smallest index such that $\text{LN}_A(u, v, x_k) \neq y$. It is clear that for all i , $[u, v]_A \cap (y, x_i)_{\text{lex}} = \emptyset$, therefore $\text{LN}_A^+(u, v, x) = \text{LN}_A^+(u, v, x_k) = \text{LN}_A(u, v, x_k)$. If, during this process, we reach some x_j such that $|x_j| > p(|v|)$, then for all $w \geq_{\text{lex}} x_j$ we have $|w| \geq |x_j| > p(|v|)$, which implies that $w >_A v$. So we can safely conclude that $[u, v]_A$ contains no string lexicographically larger than x and halt the computation leaving $\text{LN}_A^+(u, v, x)$ undefined. Note that the size of $[y, x_i]_{\text{lex}}$ is doubled after each iteration, therefore the length of x_i will exceed $p(|v|)$ after at most $\mathcal{O}(p(|v|))$ iterations. \square

Theorem 3. $\text{TotP} = \text{IF}_t^{\text{LN}}$.

Proof. We first prove that $\text{TotP} \subseteq \text{IF}_t^{\text{LN}}$. The intuition behind it is that given a TotP computation $M(x)$ and a string z , we can efficiently find a computation path, the encoding of which is lexicographically closest to z .

Let f be a TotP function, i.e. there exists an NPTM M such that on all $x \in \Sigma^*$, $f(x) = \text{tot}_M(x)$. We assume that all paths of $M(x)$ are of length exactly $p(|x|)$.

We define a total order A on Σ^* as follows: A coincides with the lexicographic order except that, for every $x \in \Sigma^*$, the interval between $x00^{p(|x|)+1}$ and $x10^{p(|x|)+1}$ (inclusive) is ordered in the following way: first comes $x00^{p(|x|)}0$, next come the elements of $\{x0y0 \mid |y| = p(|x|) \wedge y \text{ encodes a path of } M(x)\}$, in lexicographic order, next comes $x10^{p(|x|)}0$, and last come the elements of $\{x0y0 : |y| = p(|x|) \wedge y \text{ does not encode a path of } M(x)\} \cup \{x0y1 : |y| = p(|x|)\}$, in lexicographic order.

We will show that $\text{LN}_A(u, v, z)$ can be computed in polynomial time. If $z \in [u, v]_A$ then $\text{LN}_A(u, v, z) = z$. If $z \notin [u, v]_A$ we distinguish among three cases:

Case 1. Let $u = x0y_u0$ and let $v = x0y_v0$, where both y_u, y_v encode paths in $M(x)$, and let $z = x0y0$, for some $y, |y| = p(|x|)$, where y does not encode a path in $M(x)$. Let also $y_u <_{\text{lex}} y <_{\text{lex}} y_v$ (if not, the output is u or v). We simulate $M(x)$ following the non-deterministic choices according to the bits of y , until we encounter a choice that is not available. Assume without loss of generality that this choice is ‘1’. Then we follow the available choice, ‘0’, and we continue the simulation by choosing ‘1’ whenever this is available. This way we obtain the “rightmost” computation path of $M(x)$ which is lexicographically smaller than y , call it y' . Then by following a standard procedure we obtain the “leftmost” path of $M(x)$ which is lexicographically larger than y , call it y'' . Return the lexicographically closest to y between y' and y'' .

Case 2. Let $u = x0y_u a_u$ and $v = x0y_v a_v$, where y_u (y_v) encodes a path in $M(x)$ and $a_u = 1$ ($a_v = 1$), or y_u (y_v) is of length $p(|x|)$ and $a_u \in \Sigma$ ($a_v \in \Sigma$). And, furthermore let $z = z0y0$, where y encodes a computation path of $M(x)$, and $y_u <_{\text{lex}} y <_{\text{lex}} y_v$ (if not, the output is u or v). Return $x0y1$.

Case 3. The remaining cases are either trivial or can be dealt with by combining techniques used for the above two cases. Details are left for the full version.

We now give a sketch of the proof for the inclusion $\text{IF}_t^{\text{LN}} \subseteq \text{TotP}$. Let f be an IF_t^{LN} function, via a total p-order $A \in \text{P}$ with bounding polynomial p and boundary functions $b, t \in \text{FP}$. By definition, $\text{LN}_A \in \text{FP}$, therefore by Lemma [3](#) we have that $\text{LN}_A^+ \in \text{FP}$ and $\text{LN}_A^- \in \text{FP}$.

We outline the operation of an NPTM N that, on input x , performs a computation with exactly $\|(b(x), t(x))_A\| + 1$ computation paths. It first computes $b(x)$ and $t(x)$ and halts if $b(x) \prec_A t(x)$, otherwise it branches into two paths: one of them is a dummy path that halts immediately, and the other one runs a recursive procedure that accepts as input two strings u, v which satisfy the conditions $u <_{\text{lex}} v$ and $u, v \in [b(x), t(x)]_A$. This procedure first computes $z = \text{med}_{\text{lex}}(u, v)$, $z^+ = \text{LN}_A^+(b(x), t(x), z)$, and $z^- = \text{LN}_A^-(b(x), t(x), z)$. It then branches into either one or two paths that halt immediately, depending on whether $z^- = z^+$ or not. Furthermore, it branches into two recursive calls of this procedure with inputs (u, z^-) and (z^+, v) , respectively. The effect of this procedure, when initially called with inputs $u = \text{LN}_A(b(x), t(x), \varepsilon)$ and $v = \text{LN}_A(b(x), t(x), 0^{p(|t(x)|)+1})$ (that is, the lexicographically smallest and largest string in $[b(x), t(x)]_A$, respectively) is to output exactly $\|(b(x), t(x))_A\|$ computation paths. We omit the details of how to avoid branching into a recursive call that would have to count

the strings of an empty interval, but it should be clear that it is possible to check if the upcoming procedure call will have to count zero strings or more *before* the machine actually branches into it. \square

The above result, combined with the fact that TotP contains all problems in #PE which possess a natural self-reducibility property [12], implies that a number of known problems are contained in IF_t^{LN} . Actually, Theorem 3 from [12] can be restated as follows:

Corollary 1. *The problems #DNFSAT, #MONSAT, NONNEGATIVE PERMANENT, #PERFECT MATCHINGS, RANKING are IF_t^{LN} -complete under Cook-1 reductions.*

Remark 5. Note that in [12] it was shown that #MON2SAT is in TotP but the proof can be easily adapted to show that #MONSAT is in TotP as well. In fact, by slightly extending a property shown in [1], namely that it is easy to find the least satisfying assignment that is lexicographically greater than a given assignment, it is possible to show directly that #MONSAT is in IF_t^{LN} .

5 Inside TotP

In this section we give a characterization of FP as an interval size function class, and show that $\text{IF}_t^{\text{rmed}}$ is a class that contains FP and is contained in TotP.

Theorem 4. $\text{FP} = \text{IF}_t^{\text{rmed}} \subseteq \text{IF}_t^{\text{med}} \subseteq \text{TotP}$. *The first inclusion is proper unless #P = FP and the second inclusion is proper unless P = NP.*

Proof. The detailed proof is left for the full version. We only sketch some key ideas. For showing that $\text{FP} = \text{IF}_t^{\text{rmed}}$ implies #P = FP, we consider any function $f \in \#P$ and derive a function $g(x) = f(x) + 2^{p(|x|)}$, where p is a polynomial bounding the computation length of the NPTM that corresponds to f . We next show that $g \in \text{IF}_t^{\text{rmed}}$; it then suffices to notice that if $g \in \text{FP}$, then so does f . For the proof of the assumptions under which the second inclusion is proper, we introduce the exponential gap operator, \mathcal{C}_{eg} , defined as follows: if \mathcal{F} is a function class, then $\mathcal{C}_{\text{eg}} \cdot \mathcal{F}$ contains exactly the languages L for which there exist some $f \in \mathcal{F}$, $q \in \text{poly}$, and $q' \in \omega(1)$ such that for all x : if $x \notin L$ then $f(x) \leq 2^{q(|x|)}$, while if $x \in L$ then $f(x) \geq 2^{q(|x|) \cdot q'(|x|)}$. We then prove that $\text{NP} \subseteq \mathcal{C}_{\text{eg}} \cdot \text{TotP}$ and $\mathcal{C}_{\text{eg}} \cdot \text{IF}_t^{\text{rmed}} \subseteq \text{P}$. \square

Let us now define a problem that lies in $\text{IF}_t^{\text{rmed}}$:

#SAT $_{+2^n}$: given a Boolean formula φ with n variables, count the number of satisfying assignments of the formula $\varphi \vee x_{n+1}$, where x_{n+1} is a fresh variable not appearing in φ .

Proposition 2. *#SAT $_{+2^n}$ is $\text{IF}_t^{\text{rmed}}$ -complete under Cook-1 reductions.*

Proof. Membership can be shown by similar techniques to those used for proving the first part of Theorem 4 (omitted due to lack of space). For completeness it suffices to observe that #SAT can be immediately reduced to #SAT $_{+2^n}$ by subtracting 2^n . \square

From the argument used in the above proof, the following is immediate (for function classes \mathcal{F} , \mathcal{G} , let $\mathcal{F}-\mathcal{G} = \{f - g \mid f \in \mathcal{F}, g \in \mathcal{G}\}$):

Corollary 2. $\#P \subseteq \text{IF}_t^{\text{rmed}} - \text{FP}$.

Acknowledgements. We would like to thank Taso Viglas and Stathis Zachos for stimulating discussions, and the anonymous referees for their useful comments and suggestions.

References

1. Hemaspaandra, L.A., Homan, C.M., Kosub, S., Wagner, K.W.: The complexity of computing the size of an interval. *SIAM J. Comput.* 36(5), 1264–1300 (2007)
2. Kiayias, A., Pagourtzis, A., Sharma, K., Zachos, S.: The complexity of determining the order of solutions. In: *Proceedings of the First Southern Symposium on Computing*, Hattiesburg, Mississippi, December 4-5 (1998); Extended and revised version: Acceptor-definable complexity classes. LNCS 2563, pp. 453–463. Springer, Heidelberg (2003)
3. Valiant, L.G.: The complexity of computing the permanent. *Theor. Comput. Sci.* 8, 189–201 (1979)
4. Valiant, L.G.: The complexity of enumeration and reliability problems. *SIAM J. Comput.* 8(3), 410–421 (1979)
5. Toda, S.: PP is as hard as the polynomial-time hierarchy. *SIAM J. Comput.* 20(5), 865–877 (1991)
6. Toda, S., Watanabe, O.: Polynomial time 1-Turing reductions from $\#PH$ to $\#P$. *Theor. Comput. Sci.* 100(1), 205–221 (1992)
7. Kiayias, A., Pagourtzis, A., Zachos, S.: Cook reductions blur structural differences between functional complexity classes. In: *Panhellenic Logic Symposium*, pp. 132–137 (1999)
8. Dyer, M.E., Goldberg, L.A., Greenhill, C.S., Jerrum, M.: The relative complexity of approximate counting problems. *Algorithmica* 38(3), 471–500 (2003)
9. Álvarez, C., Jenner, B.: A very hard log space counting class. In: *Structure in Complexity Theory Conference*, pp. 154–168 (1990)
10. Saluja, S., Subrahmanyam, K.V., Thakur, M.N.: Descriptive complexity of $\#P$ functions. *J. Comput. Syst. Sci.* 50(3), 493–505 (1995)
11. Pagourtzis, A.: On the complexity of hard counting problems with easy decision version. In: *Proceedings of 3rd Panhellenic Logic Symposium*, Anogia, Crete, July 17-21 (2001)
12. Pagourtzis, A., Zachos, S.: The complexity of counting functions with easy decision version. In: Kráľovič, R., Urzyczyn, P. (eds.) *MFCS 2006*. LNCS, vol. 4162, pp. 741–752. Springer, Heidelberg (2006)
13. Hempel, H., Wechsung, G.: The operators min and max on the polynomial hierarchy. *Int. J. Found. Comput. Sci.* 11(2), 315–342 (2000)

On the Red/Blue Spanning Tree Problem

Sergey Bereg¹, Minghui Jiang², Boting Yang³, and Binhai Zhu⁴

¹ Department of Computer Science, University of Texas at Dallas
besp@utdallas.edu

² Department of Computer Science, Utah State University
mjiang@cc.usu.edu

³ Department of Computer Science, University of Regina
boting@cs.uregina.ca

⁴ Department of Computer Science, Montana State University
bhz@cs.montana.edu

Abstract. A geometric spanning tree of a point set S is a tree whose vertex set is S and whose edge set is a set of non-crossing straight line segments with endpoints in S . Given a set of red points and a set of blue points in the plane, the red/blue spanning tree problem is to find a geometric spanning tree for red points and a geometric spanning tree for blue points such that the number of crossing points of the two trees is minimum. If no three points are collinear, we show that the minimum number of crossing points is completely determined by the number of maximal red chains on the convex hull of all red points and blue points. We design an optimal algorithm for constructing a geometric spanning tree of all the red points and a geometric spanning tree of all the blue points with the minimum number of crossing points. If collinear points are allowed, we prove that the problem of deciding whether there exists a geometric spanning path of all the red points and a geometric spanning path of all the blue points without crossing is NP-complete.

1 Introduction

Let R be a set of red points and B be a set of blue points in the plane. In this paper, we investigate the problem of finding a geometric spanning tree for R and a geometric spanning tree for B such that the number of crossing points of the two trees is minimum. This problem has many applications. For instance, a hydro company may wish to connect their water pipelines to a set of sites R , and an energy company may want to connect their gas pipelines to a set of sites B in the same area. They want to design a network with fewest intersections between water pipelines and gas pipelines. Similar problems come up in other applications such as telecommunications, road network design, VLSI, and medical imaging.

For a pair of points a and b in the plane, we use ab to denote the straight line segment (edge) with endpoints a and b . A *polygon* is defined by a finite set of edges such that every endpoint of edges is shared by exactly two edges and no subset of edges has the same property. A polygon is *simple* if there is no pair of nonconsecutive edges sharing a point. A simple polygon partitions the plane

into two regions, the interior (bounded) and the exterior (unbounded) that are separated by the simple polygon. The union of a simple polygon and its interior is called a *polygonal region*. For a sequence of ordered points p_1, p_2, \dots, p_n in the plane, we use $p_1 p_2 \dots p_n$ ($n \geq 3$) to denote a polygon with vertices p_1, p_2, \dots, p_n , and use (p_1, p_2, \dots, p_n) ($n \geq 1$) to denote a chain with vertices p_1, p_2, \dots, p_n , where p_1 and p_n are endpoints of the chain. For a pair of straight line segments, if their intersection is an interior point of at least one of the segments, then this point is called a *crossing point*; if their intersection is a segment, then they have an infinite number of crossing points. Let S be a set of points in the plane. We use $|S|$ to denote the number of points in S . The *convex hull* of S , denoted by $\text{CH}(S)$, is the boundary of the smallest convex domain in the plane containing S . A *geometric spanning tree* of S , denoted by $\text{ST}(S)$, is a tree whose vertex set is S and whose edge set is a set of non-crossing straight line segments with endpoints in S . For a polygonal region A , the boundary of A is a simple polygon, denoted by $\text{bd}(A)$.

Throughout this paper, let R be a set of red points and B be a set of blue points in the plane satisfying $R \cap B = \emptyset$. The *red/blue spanning tree problem* is to find $\text{ST}(R)$ and $\text{ST}(B)$ such that the number of crossing points of two trees is minimum. We use $\text{ST}^*(R)$ and $\text{ST}^*(B)$ to denote an optimal solution for the red/blue spanning tree problem, and use $\text{cr}(R, B)$ to denote the number of crossing points between $\text{ST}^*(R)$ and $\text{ST}^*(B)$.

Suppose that $\text{CH}(R \cup B)$ contains at least one red point and at least one blue point. A red (resp. blue) chain (u_1, u_2, \dots, u_m) ($m \geq 1$) on $\text{CH}(R \cup B)$ is *maximal* if any longer chain on $\text{CH}(R \cup B)$ which contains (u_1, u_2, \dots, u_m) must contain at least one blue (resp. red) point. If $\text{CH}(R \cup B)$ contains both red and blue points, then the number of the maximal red chains is equal to that of the maximal blue chains.

The main results of this paper can be stated as follows:

Theorem 1. *Let R be a set of red points and B be a set of blue points such that $R \cap B = \emptyset$ and no three points in $R \cup B$ are collinear. If $\text{CH}(R \cup B)$ is monochromatic, then $\text{cr}(R, B) = 0$; if $\text{CH}(R \cup B)$ consists of k maximal red chains and k maximal blue chains, then $\text{cr}(R, B) = k - 1$. Moreover, the red/blue spanning tree problem can be solved in $O((|R| + |B|) \log(|R| + |B|))$ time.*

Note that $O((|R| + |B|) \log(|R| + |B|))$ running time is optimal because even constructing a geometric spanning tree for R takes $\Omega(|R| \log |R|)$ time when collinearity of points is allowed.

The red/blue spanning tree problem can be considered as a special geometric version of the crossing number problem [4,5].

2 Lower Bound for $\text{cr}(R, B)$

Theorem 2. *Let R be a set of red points and B be a set of blue points such that $R \cap B = \emptyset$. If no three points in $R \cup B$ are collinear and $\text{CH}(R \cup B)$ consists of k ($k \geq 2$) maximal red chains and k maximal blue chains, then $\text{cr}(R, B) \geq k - 1$.*

Proof. Let $ST^*(R)$ and $ST^*(B)$ be an optimal solution for the red/blue spanning tree problem. From each maximal red (resp. blue) chain on $CH(R \cup B)$, we arbitrarily select one endpoint and let R' (resp. B') be the set of all such endpoints. Note that $|R'| = |B'| = k$. For each leaf of $ST^*(R)$, if it is not in R' , then delete it. Recursively running this process we obtain a tree, denoted by $T(R')$, which is a subtree of $ST^*(R)$. Notice that $T(R')$ contains all the points of R' and each leaf of $T(R')$ is a point in R' . Similarly, Let $T(B')$ be a subtree of $ST^*(B)$ such that $T(B')$ contains all the points of B' and each leaf of $T(B')$ is a point in B' . Since $T(B')$ is a tree and each leaf is on $CH(R \cup B)$, we know that $CH(R \cup B) \cup T(B')$ decomposes the plane into a set \mathcal{S} of polygonal regions and the boundary of each bounded polygonal region must contain at least an edge on $CH(R \cup B)$. Consider a bounded polygonal region $A \in \mathcal{S}$. Let $L = \text{bd}(A) \cap CH(R \cup B)$ be a chain. Since $T(B')$ is a subtree of $ST^*(B)$, we know that the two endpoints of L must be blue.

We first show that L contains at most one point of R' . Suppose that L contains at least two points r_1 and r_2 of R' . Since there exists a maximal blue chain between r_1 and r_2 , there must exist a blue point b on L which is in B' and is between r_1 and r_2 . Note that there exists a path on $T(B')$ from b to an endpoint b' of L , and this path does not contain any red point. Thus, this path and the subchain of L between b and b' must form a region which is enclosed in A . This is a contradiction. Thus, L contains at most one point of R' .

We now show that L contains at least one point of R' . Suppose that L does not contain any point of R' . Since the two endpoints of L are blue, we know that L is a subchain of a maximal blue chain on $CH(R \cup B)$. Consider the two leaves of $T(B')$ which is on L . Since $\text{bd}(A)$ contains only blue points and $T(B')$ is a tree, we know that at least one leaf of $T(B')$ on L is not in B' . This contradicts the fact that every leaf of $T(B')$ belongs to B' . Therefore, L contains exactly one point of R' .

Recall that every leaf of $T(R')$ belongs to R' and each tree has at least two leaves. Arbitrarily select a red point in R' as the root of $T(R')$, denoted by r_0 . For a leaf $r_1 (\neq r_0)$ of $T(R')$, there is a unique bounded region $A_{r_1} \in \mathcal{S}$ such that the chain $L_{r_1} = \text{bd}(A_{r_1}) \cap CH(R \cup B)$ contains r_1 . Since no three points in $R \cup B$ are collinear, there must exist an edge e_{r_1} in the path from r_0 to r_1 on $T(R')$ such that e_{r_1} intersects an edge in $\text{bd}(A_{r_1}) \cap T(B')$ at c_{r_1} and there is no other crossing point between $T(B')$ and the path from c_{r_1} to r_1 on $T(R')$. Let $T(R' \setminus \{r_1\})$ be a subtree of $ST^*(R)$ such that $T(R' \setminus \{r_1\})$ contains all the points of $R' \setminus \{r_1\}$ and each leaf of $T(R' \setminus \{r_1\})$ is a point in $R' \setminus \{r_1\}$. Since A_{r_1} contains only r_1 and there is no crossing point between $T(B')$ and the path from c_{r_1} to r_1 on $T(R')$, we know that e_{r_1} is not on $T(R' \setminus \{r_1\})$. Similarly, for a leaf $r_2 (\neq r_0)$ of $T(R' \setminus \{r_1\})$, there is a unique bounded region $A_{r_2} \in \mathcal{S}$ such that the chain $L_{r_2} = \text{bd}(A_{r_2}) \cap CH(R \cup B)$ contains r_2 . Thus, there exists an edge e_{r_2} in the path from r_0 to r_2 on $T(R' \setminus \{r_1\})$ such that e_{r_2} intersects an edge in $\text{bd}(A_{r_2}) \cap T(B')$ at c_{r_2} and there is no other crossing point between $T(B')$ and the path from c_{r_2} to r_2 on $T(R' \setminus \{r_1\})$. Repeat the above process $k - 1$ times until $R' \setminus \{r_1, r_2, \dots, r_{k-1}\} = \{r_0\}$. Thus, we have found $k - 1$ different

crossing points $c_{r_1}, c_{r_2}, \dots, c_{r_{k-1}}$. Hence the number of crossing points between $T(R')$ and $T(B')$ is at least $k - 1$. Since $T(R')$ and $T(B')$ are subtrees of $ST^*(R)$ and $ST^*(B)$ respectively, we have $\text{cr}(R, B) \geq k - 1$.

3 Optimal Solutions

First, we prove a key lemma providing an useful tool for constructing spanning trees with minimum crossings.

Let $P = p_1p_2 \dots p_mq_1q_2 \dots q_n$ be a polygon of red/blue points such that (1) both polygons $p_1p_2 \dots p_mq_1$ and $q_1q_2 \dots q_np_1$ are convex, (2) vertices q_2, \dots, q_n are inside the polygon $p_1p_2 \dots p_mq_1$, and (3) each p_i has the same color that is different from the color of any q_i . Such a polygon is called a *crescent*, in which (p_1, \dots, p_m) is called the *outer chain* and (q_1, \dots, q_n) is called the *inner chain*. Let S be a set of points inside P . We now consider how to construct a $ST(S \cup \{p_1, \dots, p_m\})$ whose edges are on or inside P . It is easy to see that the triangulation of P , denoted by $T(P)$, can be constructed in linear time. For each triangle in $T(P)$ containing a subset of points $S' \subseteq S$, since at least one vertex, say p , of the triangle belongs to $\{p_1, \dots, p_m\}$, we can link p with every points in S' . In this way, every point of S is linked to a point in $\{p_1, \dots, p_m\}$. Thus, all the added edges and the chain (p_1, p_2, \dots, p_m) form a $ST(S \cup \{p_1, \dots, p_m\})$ whose edges are on the outer chain or inside P . Therefore, we have the following lemma.

Lemma 1. *If $P = p_1p_2 \dots p_mq_1q_2 \dots q_n$ is a crescent with outer chain (p_1, p_2, \dots, p_m) and the inner chain (q_1, q_2, \dots, q_n) , and S is a set of points inside P , then there exists a $ST(S \cup \{p_1, \dots, p_m\})$ whose edges are on the outer chain or inside P .*

3.1 One Maximal Red/Blue Chain on $CH(R \cup B)$

Let $CH(R \cup B) = u_0u_1 \dots u_mu_nv_{n+1} \dots v_1v_0$ with the maximal red chain (u_0, u_1, \dots, u_m) ($m \geq 0$) and the maximal blue chain (v_0, v_1, \dots, v_n) ($n \geq 0$). If $m = n = 0$, then $CH(R \cup B)$ is an edge. Since no three points in $R \cup B$ are collinear, we have $R = \{u_0\}$ and $B = \{v_0\}$. Thus, $ST^*(R)$ is an empty graph (only vertex u_0) and $ST^*(B)$ is also an empty graph. In the remaining of this subsection, we suppose that at least one of the maximal chains has at least two vertices. We will decompose the interior of $CH(R \cup B)$ into a sequence of crescent regions.

Let $CH(B \cup \{u_0\}) = v_0v_1 \dots v_nv_{n+1} \dots v_{n'}u_0$. Note that the polygon $P = u_0u_1 \dots u_mu_nv_{n+1} \dots v_{n'}$ is a crescent with outer chain (u_0, u_1, \dots, u_m) and inner chain $(v_n, v_{n+1}, \dots, v_{n'})$, and all the points inside this polygon are red points. From Lemma [1](#), we can construct a geometric spanning tree for $\{u_0, u_1, \dots, u_m\}$ and all the points inside P . Let R_1 be the set of all the red points inside $CH(\{u_0\} \cup B)$. Let P_1 be the polygon between $CH(\{u_0\} \cup B)$ and $CH(R_1 \cup \{u_0, v_0\})$ which is a crescent with a blue outer chain and a red inner chain. All the points inside P_1 are blue. From Lemma [1](#), we can construct a geometric spanning tree for all the points on the outer chain and inside P_1 . Let B_1 be the set of all the blue points inside $CH(R_1 \cup \{u_0, v_0\})$. Let P_2 be the polygon

between $\text{CH}(R_1 \cup \{u_0, v_0\})$ and $\text{CH}(B_1 \cup \{u_0, v_0\})$ which is a crescent with a red outer chain and a blue inner chain. We can construct a geometric spanning tree for all the red points on the outer chain and inside P_2 . Repeat this process until every red point is linked to a red tree and every blue point is linked to a blue tree. Since every red tree of crescents contains u_0 and every blue tree of crescents contains v_0 , we know that all the red trees form a $\text{ST}(R)$, all the blue trees form a $\text{ST}(B)$, and there is no crossing point between them. Thus, we have the following theorem.

Theorem 3. *Let R be a set of red points and B be a set of blue points such that $R \cap B = \emptyset$. Let $\text{CH}(R \cup B) = u_0 u_1 \dots u_m v_n \dots v_1 v_0$ with the maximal red chain (u_0, u_1, \dots, u_m) ($m \geq 0$) and the maximal blue chain (v_0, v_1, \dots, v_n) ($n \geq 0$). If no three points in $R \cup B$ are collinear, then $\text{cr}(R, B) = 0$.*

3.2 Only Red or Blue Points on $\text{CH}(R \cup B)$

In this subsection, we consider the case that all the vertices on $\text{CH}(R \cup B)$ have the same color, say red. Let $\text{CH}(R \cup B) = u_0 u_1 \dots u_m$ and $\text{CH}(\{u_0\} \cup B) = u_0 v_0 v_1 \dots v_n$. Similar to the crescent polygon, we can triangulate the polygonal region P between $\text{CH}(R \cup B)$ and $\text{CH}(\{u_0\} \cup B)$. Since each triangle in the triangulation must have a red vertex that is different from u_0 , we link this red vertex to every red vertex inside this triangle. All the added edges and the chain (u_0, u_1, \dots, u_m) form a geometric spanning tree of all the points in $\{u_0, u_1, \dots, u_m\}$ and inside P . Since $\text{CH}(\{u_0\} \cup B)$ consists of one maximal red chain and one maximal blue chain, it follows from Theorem 3 that we can construct a geometric spanning tree for all the red points on and inside $\text{CH}(\{u_0\} \cup B)$ and a geometric spanning tree for all the blue points on and inside $\text{CH}(\{u_0\} \cup B)$. Thus, we have the following result.

Theorem 4. *Let R be a set of red points and B be a set of blue points such that $R \cap B = \emptyset$. If no three points in $R \cup B$ are collinear and all the vertices of $\text{CH}(R \cup B)$ have the same color, then $\text{cr}(R, B) = 0$.*

3.3 Two Red and Two Blue Vertices on $\text{CH}(R \cup B)$

In this subsection, we consider the case that $\text{CH}(R \cup B) = u_0 v_0 u_1 v'_0$ with two maximal red chains, (u_0) and (u_1) , and two maximal blue chains, (v_0) and (v'_0) . By linking u_0 with u_1 , we can decompose the problem into two subproblems. Let B_1 and B_2 be sets of blue points in triangles $u_0 u_1 v_0$ and $u_0 u_1 v'_0$ respectively. Let P_1 be the crescent between triangle $u_0 u_1 v_0$ and $\text{CH}(B_1 \cup \{u_0, v_0\})$, and v_n be the vertex of P_1 which is adjacent to u_0 . Let P_2 be the crescent between triangle $u_0 u_1 v'_0$ and $\text{CH}(B_2 \cup \{u_0, v'_0\})$, and v'_n be the vertex of P_2 which is adjacent to u_0 . Note that all the points inside P_1 and P_2 are red. For each red point inside P_1 and P_2 , if it is inside triangle $u_0 v_n v'_n$, then link it with u_0 , otherwise, link it to u_1 . Thus, all the added edges form a geometric spanning tree of $\{u_0, u_1\}$ and all the points inside P_1 and P_2 .

Let R' and R'' be sets of red points in triangles $u_0u_1v_0$ and $u_0u_1v'_0$ respectively. Since $\text{CH}(B_1 \cup \{u_0, v_0\})$ consists of one maximal red chain and one maximal blue chain, it follows from Theorem 3 that we can construct $\text{ST}^*(R' \cup \{u_0, u_1\})$ and $\text{ST}^*(B_1 \cup \{v_0\})$ without any crossing point. Similarly, we can also construct $\text{ST}^*(R'' \cup \{u_0, u_1\})$ and $\text{ST}^*(B_2 \cup \{v'_0\})$ without any crossing point. The union of $\text{ST}^*(R' \cup \{u_0, u_1\})$ and $\text{ST}^*(R'' \cup \{u_0, u_1\})$ forms $\text{ST}(R)$, and the union of $\text{ST}(B_1 \cup \{v_0\})$, $\text{ST}(B_2 \cup \{v'_0\})$ and the edge $v_nv'_n$ forms $\text{ST}(B)$. From the above construction, we know that there is only one crossing point between $\text{ST}(R)$ and $\text{ST}(B)$ which is the intersection point of u_0u_1 and $v_nv'_n$. It follows from Theorem 2 that $\text{ST}(R)$ and $\text{ST}(B)$ is an optimal solution. Thus, we have the following result.

Lemma 2. *Let R be a set of red points and B be a set of blue points such that $R \cap B = \emptyset$ and no three points in $R \cup B$ are collinear. If $\text{CH}(R \cup B) = u_0v_0u_1v'_0$, where $u_0, u_1 \in R$ and $u_0, v'_0 \in B$, then $\text{cr}(R, B) = 1$.*

3.4 More Than One Maximal Red/Blue Chains on $\text{CH}(R \cup B)$

In this subsection, we consider the case that $\text{CH}(R \cup B)$ consists of k ($k \geq 2$) maximal red chains and k maximal blue chains. Let $\text{CH}(R \cup B) = u_1^1 \dots u_{m_1}^1 u_1^2 \dots u_{m_2}^2 \dots u_1^k \dots u_{m_k}^k v_{n_k}^k \dots v_1^k \dots v_{n_2}^2 \dots v_1^2 v_{n_1}^1 \dots v_1^1$, where $k \geq 2$, $m_i \geq 1$, $n_i \geq 1$, $1 \leq i \leq k$ and each chain $(u_1^i, \dots, u_{m_i}^i)$ is a maximal red or blue chain and each chain $(v_1^i, \dots, v_{n_i}^i)$ is also a maximal red or blue chain. We first add the following edges $u_1^1v_1^1, u_{m_1}^1v_{n_1}^1, u_1^2v_1^2, u_{m_2}^2v_{n_2}^2, \dots, u_1^kv_1^k, u_{m_k}^kv_{n_k}^k$. These edges decompose the problem into at most $2k - 1$ subproblems with the following convex hulls: $u_1^1 \dots u_{m_1}^1 v_{n_1}^1 \dots v_1^1, u_{m_1}^1 u_1^2 v_1^2 v_{n_1}^1, u_1^2 \dots u_{m_2}^2 v_{n_2}^2 \dots v_1^2, u_{m_2}^2 u_1^3 v_1^3 v_{n_2}^2, \dots, u_1^k \dots u_{m_k}^k v_{n_k}^k \dots v_1^k$.

For every convex polygon $u_1^i \dots u_{m_i}^i v_{n_i}^i \dots v_1^i$, $1 \leq i \leq k$, from Theorem 3, we can construct a geometric spanning tree for red points and a geometric spanning tree for blue points on and inside the polygon without any crossing point. For every quadrilateral $u_{m_i}^i u_1^{i+1} v_1^{i+1} v_{n_i}^i$, $1 \leq i \leq k - 1$, from Lemma 2, we can construct a geometric spanning tree for red points and a geometric spanning tree for blue points in the quadrilateral with one crossing point. The union of all the red geometric spanning trees forms $\text{ST}(R)$ and the union of all the blue geometric spanning trees form $\text{ST}(B)$, and there are $k - 1$ crossing points between $\text{ST}(R)$ and $\text{ST}(B)$. From Theorem 2, we know $\text{ST}(R)$ and $\text{ST}(B)$ are the optimal solution. Therefore, we have the main result of this paper.

Theorem 5. *Let R be a set of red points and B be a set of blue points such that $R \cap B = \emptyset$ and no three points in $R \cup B$ are collinear. If $\text{CH}(R \cup B)$ consists of k maximal red chains and k maximal blue chains, then $\text{cr}(R, B) = k - 1$.*

4 Algorithms

We first describe the algorithms as follows.

Algorithm **RBT**(R, B)

/* Given a red point set R and a blue point set B such that $R \cap B = \emptyset$ and no three points are collinear, find $\text{ST}(R)$ and $\text{ST}(B)$ with minimum crossings. */

1. Compute $\text{CH}(R \cup B)$.
2. If $\text{CH}(R \cup B)$ is monochromatic then call **RBT-mono**(R, B).
3. Compute k , the number of red chains in $\text{CH}(R \cup B)$.
4. If $k = 1$ then call **RBT-one**(R, B).
5. If $k \geq 2$ then
 - (a) partition both R and B into $2k - 1$ sets $R = R_1 \cup \dots \cup R_{2k-1}$ and $B = B_1 \cup \dots \cup B_{2k-1}$ as in section 3.4
 - (b) For each $i = 1, 3, \dots, 2k - 1$ call **RBT-one**(R_i, B_i).
 - (c) For each $i = 2, 4, \dots, 2k - 2$ call **RBT-quad**(R_i, B_i).

Algorithm **RBT-mono**(R, B)

/* Find $\text{ST}(R)$ and $\text{ST}(B)$ without crossing if $\text{CH}(R \cup B)$ is monochromatic. */

1. If $\text{CH}(R \cup B) = u_0 \dots u_m$ is red, then compute a crescent P between $\text{CH}(R \cup B)$ and $\text{CH}(\{u_0\} \cup B)$. Let S be the set of red points inside P .
Call **Crescent**(P, S, u_0, u_m).
Call **RBT-one**(R_1, B_1), where R_1 and B_1 is the sets of red and blue points on and inside $\text{CH}(\{u_0\} \cup B)$ respectively.
2. If $\text{CH}(R \cup B)$ is blue, then compute a crescent P between $\text{CH}(R \cup B)$ and $\text{CH}(R \cup \{u_0\})$. Let S be the set of blue points inside P .
Call **Crescent**(P, S, u_0, u_m).
Call **RBT-one**(R_1, B_1), where R_1 and B_1 is the sets of red and blue points on and inside $\text{CH}(R \cup \{u_0\})$ respectively.

Algorithm **RBT-one**(R, B)

/* Find $\text{ST}(R)$ and $\text{ST}(B)$ without crossing if $\text{CH}(R \cup B)$ consists of one maximal red chain and one maximal blue chain. */

1. Let $\text{CH}(R \cup B) = u_0 u_1 \dots u_m v_n \dots v_1 v_0$ with the maximal red chain (u_0, u_1, \dots, u_m) and the maximal blue chain (v_0, v_1, \dots, v_n) . If $m = n = 0$, then return.
2. If $n \geq 1$, compute $\text{CH}(\{u_0\} \cup B)$ and the crescent P between $\text{CH}(R \cup B)$ and $\text{CH}(\{u_0\} \cup B)$. Let S be the set of red points in P . Let R_1 and B_1 be the sets of red and blue points on and inside $\text{CH}(\{u_0\} \cup B)$ respectively.
Call **Crescent**(P, S, u_0, u_m).
Call **RBT-one**(R_1, B_1).
3. If $n = 0$, compute $\text{CH}(R \cup \{v_0\})$ and the crescent P between $\text{CH}(R \cup B)$ and $\text{CH}(R \cup \{v_0\})$. Let S be the set of blue points in P . Let R_1 and B_1 be the sets of red and blue points on and inside $\text{CH}(R \cup \{v_0\})$ respectively.
Call **Crescent**(P, S, v_0, v_n).
Call **RBT-one**(R_1, B_1).

Algorithm **RBT-quad**(R, B)

/* Find $\text{ST}(R)$ and $\text{ST}(B)$ with one crossing point if $\text{CH}(R \cup B) = u_0 v_0 u_1 v'_0$, where $u_0, u_1 \in R$ and $v_0, v'_0 \in B$. */

1. Let $\text{CH}(R \cup B) = u_0 v_0 u_1 v'_0$, where $u_0, u_1 \in R$ and $v_0, v'_0 \in B$. Let B_1 and B_2 be sets of blue points inside triangles $u_0 u_1 v_0$ and $u_0 u_1 v'_0$ respectively.
2. Compute the crescent P_1 between triangle $u_0 u_1 v_0$ and $\text{CH}(B_1 \cup \{u_0, v_0\})$. Compute the crescent P_2 between triangle $u_0 u_1 v'_0$ and $\text{CH}(B_2 \cup \{u_0, v'_0\})$.
3. Find the vertex v_n on P_1 which is adjacent to u_0 , and find the vertex v'_n on P_2 which is adjacent to u_0 . Connect v_n with v'_n .
4. For each red point inside P_1 and P_2 , if it is inside triangle $u_0 v_n v'_n$, then link it with u_0 , otherwise, link it to u_1 .
5. Let R_1 and R_2 be sets of red points in $\text{CH}(B_1 \cup \{u_0, v_0\})$ and $\text{CH}(B_2 \cup \{u_0, v'_0\})$ respectively.
 Call $\text{RBT-one}(R_1 \cup \{u_0\}, B_1 \cup \{v_0\})$.
 Call $\text{RBT-one}(R_2 \cup \{u_0\}, B_2 \cup \{v'_0\})$.

Algorithm **Crescent**(P, S, p_1, p_m)

/* Given a crescent P with outer chain (p_1, \dots, p_m) and a set of points S inside P , find a $\text{ST}(S \cup \{p_1, \dots, p_m\})$ whose edges are on the outer chain or inside P .
 */

1. For $i = 1, \dots, m - 1$, link p_i with p_{i+1} , where (p_1, \dots, p_m) is the outer chain of the crescent P .
2. Compute the triangulation \mathcal{T} of P .
3. For each point $v \in S$, find the triangle in \mathcal{T} which contains v , and link v with a vertex of the triangle which is on the outer chain of P .

Theorem 6. *Algorithm $\text{RBT}(R, B)$ can be implemented with $O((|R| + |B|) \log(|R| + |B|))$ running time.*

Proof. In **Crescent**(P, S, p_1, p_m), it takes $O(|P|)$ time in Steps 1 and 2, and it takes $O(|S| \log |P|)$ in Step 3. Thus, the running time of **Crescent** is $O(|P| + |S| \log |P|)$.

For **RBT-one**(R, B), we first use the following procedure to compute the crescent P between $\text{CH}(R \cup B)$ and $\text{CH}(\{u_0\} \cup B)$ in Step 2.

- i. Set $u = u_0$ initially.
- ii. Find the vertex u' that is next to u in counterclockwise order on the convex hull. If $u' = v_n$, then stop.
- iii. If u' is red, then delete u' and update the convex hull; otherwise set $u = u'$ and go to Step ii.

Since the data structure described in [1] can support insertions and deletions in $O(\log n)$ amortized time, we know that Steps ii and iii take $O(k \log(|R| + |B|))$ time, where k is the number of red points deleted in Step iii. Similarly, it takes $O(k' \log(|R| + |B|))$ time to compute the crescent P between $\text{CH}(R \cup B)$ and $\text{CH}(R \cup \{v_0\})$ in Step 3, where k' is the number of deleted blue points. (The difference is that we delete blue points in clockwise order starting from v_0). Hence, the running time of **RBT-one** is $O((|R| + |B|) \log(|R| + |B|))$.

We now consider **RBT-quad**(R, B). Step 2 takes $O(k' + k \log(|R| + |B|))$ time, where k and k' are the number of red and blue points in P_1 and P_2 respectively. Step 3 takes $O(\log(|R| + |B|))$ time. Step 4 takes $O(k)$ time. Step 5

takes $O((|R| + |B|) \log(|R| + |B|))$ time. Thus, the running time of **RBT-quad** is $O((|R| + |B|) \log(|R| + |B|))$.

It is easy to see that the running time of **RBT-mono** is $O((|R| + |B|) \log(|R| + |B|))$. Therefore, the total running time of **RBT**(R, B) is $O((|R| + |B|) \log(|R| + |B|))$.

5 Red/Blue Paths

In this section, we consider the red/blue path problem, a variation of the red/blue tree problem. Given a red point set R and a blue point set B in the plane, the *red/blue path problem* is to find a geometric spanning path of R and a geometric spanning path of B such that the number of crossing points between two paths is minimum.

Theorem 7. *The RB-PATH problem is NP-complete.*

Proof. It is easy to see that the RB-PATH problem belongs to NP. We will show it is NP-hard by a reduction from the planar Hamilton path problem that is NP-complete. Given a planar graph G , the planar Hamilton path problem is to determine whether G contains a Hamilton path. From [3], there is a planar straight line embedding of G , denoted as G' , such that no three vertices are collinear. Let δ be the minimum distance between a vertex and an edge in G' . Let the vertex set of G' be the blue point set B in the RB-PATH problem. For each vertex v in G' , let $\text{circ}(v)$ be a circle with center v and radius γ satisfying $\gamma < \delta/2$. Let \bar{G}' be the complement graph of G' . For every intersection point between an edge of \bar{G}' and a circle around a vertex of G' , we mark it red. Let R_1 be the set of all these intersection points. For every vertex v of G' , the edges of G' incident with v divide $\text{circ}(v)$ into $\text{deg}(v)$ arcs, where $\text{deg}(v)$ is the degree of v on G' . For each arc not containing any red point, we mark the mid-point of the arc red, and let R_2 be the set of all such midpoints. For every vertex v of G' , if v is not inside the convex hull of red points on $\text{circ}(v)$, then we can choose at most three points on $\text{circ}(v)$ to mark them red such that v is inside the convex hull of red points on $\text{circ}(v)$. Let R_3 be the set of these additional red points. Finally, let $R = R_1 \cup R_2 \cup R_3$ be the set of all the red points. So we have constructed an instance of the RB-PATH problem in polynomial time.

We now show that G' (or G) has a Hamilton path if and only if there is a red path spanning R and a blue path spanning B without crossing. First, suppose that G' has a Hamilton path H . Then this path H spans B . So we only need to find the red path. For each vertex v of G' , we first construct the convex hull of all the red points on $\text{circ}(v)$, denoted by $\text{conv}(v)$. For any edge uv on H , let $u'u''$ be an edge on $\text{conv}(u)$ which intersects uv , and $v'v''$ be an edge on $\text{conv}(v)$ which intersects uv . Delete edges $u'u''$ and $v'v''$, and add edges $u'v'$ and $u''v''$ (suppose that they do not intersect). After running this operation on every edge of H , we have a cycle spanning R and surrounding H . Note that this cycle has no self-intersection because $\gamma < \delta/2$. Arbitrarily deleting an edge from this cycle, we obtain a spanning path of R that does not intersect H .

Conversely, if there is a red path spanning R and a blue path spanning B without crossing, then it follows from the construction of R_1 and B that every edge on the blue path must be an edge of G' . Thus, this blue path is a Hamilton path of G' .

6 Conclusions

Let R be a set of red points and B be a set of blue points such that $R \cap B = \emptyset$ and no three points in $R \cup B$ are collinear. We have designed an algorithm with the optimal running time $O((|R| + |B|) \log(|R| + |B|))$ for constructing a geometric spanning tree of R and a geometric spanning tree of B with the minimum number of crossing points. If collinearity of points is allowed, we have shown that the problem of determining whether there exists a geometric spanning path of R and a geometric spanning path of B without crossing is NP-complete.

Note that the method used to compute crescents can be extended to compute the convex layers of R and B (refer to [2]). Suppose that $\text{CH}(R \cup B)$ contains only red points. Let $R_0 = \text{CH}(R \cup B)$, and then, for $i = 0, 1, 2, \dots$, we can recursively define B_i to be the convex hull of all the blue points inside R_i , and R_{i+1} to be the convex hull of all the red points inside B_i . Suppose that B_h , $h \geq 0$, is the last nonempty hull. Similar to the procedure for computing crescents in the proof of Theorem 6, we can design an $O((|R| + |B|) \log(|R| + |B|))$ time algorithm for computing all R_i and B_i , $0 \leq i \leq h$.

An interesting extension of the red/blue tree problem is the degree constrained red/blue tree problem: Given two degree bounds k_r and k_b , find a geometric spanning tree of R with maximum degree at most k_r and a geometric spanning tree of B with maximum degree at most k_b such that the number of crossing points is minimum. We can also define other variations of the red/blue tree problem. By using a reduction similar to the one used in Theorem 7, we can prove most of them are NP-complete. Note that in the proof of Theorem 7, we use the collinearity as a tool to construct the reduction. We conjecture that the RB-PATH problem is still NP-complete if all the points are in general position.

References

1. Brodal, G.S., Jacob, R.: Dynamic planar convex hull. In: Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science (FOCS 2002), pp. 617–626 (2002)
2. Chazelle, B.: On the convex layers of a planar set. *IEEE Transactions on Information Theory* IT-31, 509–517 (1985)
3. Fary, I.: On straight line representations of planar graphs. *Acta Sci. Math. (Szeged)* 11, 229–233 (1948)
4. Garey, M., Johnson, D.: Crossing number is NP-complete. *SIAM J. Algebraic and Discrete Methods* 4, 312–316 (1983)
5. Hliněný, P.: Crossing number is hard for cubic graphs. *Journal of Combinatorial Theory, Series B* 96, 455–471 (2006)

Undecidability of Cost-Bounded Reachability in Priced Probabilistic Timed Automata^{*}

Jasper Berendsen¹, Taolue Chen², and David N. Jansen¹

¹ Radboud University Nijmegen, Institute for Computing and Information Sciences,
Nijmegen, The Netherlands

² CWI, Department of Software Engineering, Amsterdam, The Netherlands

Abstract. Priced Probabilistic Timed Automata (PPTA) extend timed automata with cost-rates in locations and discrete probabilistic branching. The model is a natural combination of Priced Timed Automata and Probabilistic Timed Automata. In this paper we focus on cost-bounded probabilistic reachability for PPTA, which determines if the maximal probability to reach a goal location within a given cost bound (and time bound) exceeds a threshold $p \in (0, 1]$. We prove undecidability of the problem for simple PPTA in 3 variants: with 3 clocks and stopwatch cost-rates or strictly positive cost-rates. Because we encode a 2-counter machine in a new way, we can also show undecidability for cost-rates in \mathbb{Z} and only 2 clocks.

1 Introduction

Digital technology has been widely deployed in safety-critical situations and real-life environments, which leads to increased interest in computer systems that satisfy quantitative timing constraints. Timed automata [1] are a prominent and well-established formalism for modeling, analysis and verification of such *real-time* systems, which have received much attention both in terms of theoretical and practical developments.

In addition to computation time, systems also use other finite resources, e. g. energy, memory, or bandwidth. In many cases, some resources are scarce; the system should not use more resources than a certain budget. *Priced (or weighted) timed automata* [2,3] model resource use and resource constraints.

Traditional approaches to the formal description of real-time systems usually express the system model purely in terms of nondeterminism. However, many real-life systems, such as multimedia equipment, communication protocols and networks, exhibit random behavior. Thus we may ask for the likelihood that certain properties are satisfied. This suggests the study of *probabilistic* models. In this paper, we investigate *priced probabilistic timed automata* (PPTAs) [4], which are a probabilistic extension of priced timed automata. This model is an orthogonal extension of priced as well as probabilistic timed automata [5].

^{*} Research supported by NWO/EW project 612.000.103 FRAAI, the Dutch Bsik project BRICKS. and the European Community's 7th Framework Programme No. 214755 (QUASIMODO).

One of the most fundamental problems for timed automata and their variants is reachability. In the setting of PPTA, *cost-bounded probabilistic reachability* asks: “Is it possible to reach a goal state with probability $\geq p$ within a given cost (and time) bound?” This problem has been studied in [4], where the authors provided a *semi-algorithm*: If the answer is affirmative or the symbolic state space is finite, the algorithm terminates; however, the decidability of the problem remained open. In this paper, we show its *undecidability*. The proof reduces a 2-counter machine to a PPTA with three clocks; the 2-counter machine does not terminate iff some state in the PPTA is reachable with probability 1. Moreover, the PPTA can be restricted to: 1. either only cost-rates $\in \{0, 1\}$ or only cost-rates > 0 , 2. no difference constraints nor strict constraints, and 3. no probabilistic resets. So, even when cost must increase with time passing, it may be necessary for the semi-algorithm of [4] to investigate infinitely many symbolic states. Undecidability also holds for PPTA with two clocks that allow cost-rates $\in \mathbb{Z}$.

Related Work. Although greatly inspired by [6], there are some thorough changes in our encoding of a 2-counter machine. First, we use a single clock to encode both counters, similar to [7]. We find our encoding simpler, since it uses the third clock only in one subautomaton. Second, [6] shows undecidability in the setting of the logic WCTL on priced timed automata, as well as in the setting of weighted timed games. In both settings the goal state is reached by simulating a terminating execution, or by doing a test after simulating an initial fragment of any execution. The 2-counter machine terminates iff the goal state cannot be avoided indefinitely. For our setting this would not work, because tests are entered probabilistically; the (now probabilistic) choice whether to continue simulation or do a test cannot avoid testing infinitely often.

The other way around, our undecidability results carry over to the setting of [6]. Our Theorem 1 shows a somewhat stronger result, since our PPTA forbid strict guards. Theorem 2 shows a new result on only two clocks, while often three clocks are necessary. Because of the strictly positive cost-rates, Theorem 3 also gives new insight in the setting of [6]. The innovative construction for Theorem 3 ensures that the time to reach the goal state is always 9 time units; it uses the third clock to measure the runtime. Theorem 2 also carries over to the game setting of [7] for two clocks and a lower bound.

Outlook. A possible continuation of this work is by having the slightly different notion of cost-bounded probabilistic reachability as in [4], namely to have $> p$ instead of $\geq p$ on reachability. The semi-algorithm in [4] does not necessarily terminate for $= p$ on probability. Can this crack between the two results be closed?

2 Preliminaries

A *probability distribution* over a finite set Q is a function $\mu : Q \rightarrow [0, 1]$ with $\sum_{q \in Q} \mu(q) = 1$. For set Q' , let $\text{Dist}(Q')$ be the set of distributions over finite subsets of Q' .

A *clock* is a real-valued variable that can be used to measure the elapse of time. A *clock valuation* is a mapping $\mathbb{X} \rightarrow \mathbb{R}_{\geq 0}$, assigning a value to each clock in some finite set \mathbb{X} . Let $\mathbb{R}_{\geq 0}^{\mathbb{X}}$ denote the set of all clock valuations. For $v \in \mathbb{R}_{\geq 0}^{\mathbb{X}}$ and $d \in \mathbb{R}_{\geq 0}$, let $v+d$ denote the clock valuation that maps each $x \in \mathbb{X}$ to $v(x) + d$. For $r \subseteq \mathbb{X}$, let $v[r:=0]$ denote the *reset* of the clocks in r , i.e. $v[r:=0](x)$ equals 0 if $x \in r$ and $v(x)$ otherwise. Valuation $v_{\text{zero}} \in \mathbb{R}_{\geq 0}^{\mathbb{X}}$ assigns 0 to all clocks in \mathbb{X} .

A *zone* or *constraint* is a conjunction of non-strict inequalities where the value of a single clock is compared to an integer. Formally, for the set \mathbb{X} of clocks the set $Zones(\mathbb{X})$ of zones Z is defined by the grammar: $Z ::= x \leq b \mid x \geq b \mid Z \wedge Z$, where $x \in \mathbb{X}$, $b \in \mathbb{N}$. Note that some other definitions [1] allow strict inequalities and inequalities on the difference between clocks, e.g. $x > 2$, $x - y < 3$.

2.1 Priced Probabilistic Timed Automata

The next definition a PPTA differs from [4] by: having no invariants, having only edges that incur cost 0, using our restricted notion of zones, and allowing negative cost-rates.

Definition 1. A PPTA is a tuple $(L, l_{\text{init}}, \mathbb{X}, \text{edges}, \$)$, where L is a finite set of locations; $l_{\text{init}} \in L$ is the initial location; \mathbb{X} is a finite set of clocks; $\text{edges} \subseteq L \times Zones(\mathbb{X}) \times \text{Dist}(2^{\mathbb{X}} \times L)$ is a finite set of edges; and $\$: L \rightarrow \mathbb{Z}$ associates a cost-rate with each location.

For edge $(l, g, p) \in \text{edges}$, l denotes the source location, g the guard (which is a zone), and p a distribution on pairs of a set of clocks to be reset and a destination location. Figure 1 shows a PPTA with clock x . The locations are represented by circles, with branching arrows between them denoting the edges of the PPTA. The initial location l_0 is marked with a dangling arrow. The cost-rates are written next to the locations. Guards (e.g. $x \geq 1$) are next to the source location; the probabilities and resets are at the branches (e.g. probability 0.1 and $x:=0$.) Cost-rate 0, probability 1, and guards that always hold are omitted.

Intuitively, a PPTA behaves as follows. It always is in a state consisting of a location l , a clock valuation v and the amount of cost already incurred. A policy fills in the non-deterministic choice between the outgoing edges to take, or delaying. Only edges with guards satisfying the current valuation are available. Delaying will increase each clock by the amount of delay, and the accumulated cost by the amount of delay times the the cost-rate ($\$(l)$). When taking an edge, one reset set and a destination location are chosen probabilistically, these clocks are reset and the system enters the destination.

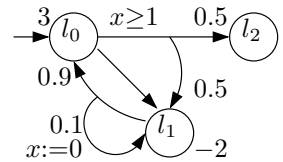


Fig. 1. Example PPTA

Definition 2. A Markov Decision Process (MDP) is a tuple $(S, s_{\text{init}}, \text{Act}, \pi)$, where S is a set of states, $s_{\text{init}} \in S$ is the initial state, Act is a set of action labels, and $\pi \subseteq S \times \text{Act} \times \text{Dist}(S)$ is a probabilistic transition relation such that for each $s \in S$, there exist $a \in \text{Act}$ and $\mu \in \text{Dist}(S)$ such that $(s, a, \mu) \in \pi$.

A (in)finite *run* ω in an MDP $(S, s_{\text{init}}, \text{Act}, \pi)$ is a (in)finite sequence: $s_0 \xrightarrow{a_0, \mu_0} s_1 \xrightarrow{a_1, \mu_1} s_2 \xrightarrow{a_2, \mu_2} \dots$ such that $s_0 = s_{\text{init}}$, $(s_i, a_i, \mu_i) \in \pi$, and $\mu_i(s_{i+1}) > 0$ for all i . Let ω_i denote the i -th state in the run ω , i.e. $\omega_i = s_i$. Let $\text{last}(\omega)$ denote the last state in the finite run ω . A *policy* (also called scheduler, adversary, or strategy) is a function mapping every finite run ω in some MDP $(S, s_{\text{init}}, \text{Act}, \pi)$ to a pair $(a, \mu) \in \text{Act} \times \text{Dist}(S)$ such that $(\text{last}(\omega), a, \mu) \in \pi$. For a policy A , let Runs^A denote the set of all infinite runs that are induced by A . Prob^A denotes the probability measure on Runs^A , defined using classical techniques [8].

Definition 3 (PPTA Semantics). *Given PPTA $\text{Aut} = (L, l_{\text{init}}, \mathbb{X}, \text{edges}, \dot{\$})$, its semantics is the MDP: $\text{MDP}(\text{Aut}) = (S, (l_{\text{init}}, v_{\text{zero}}, 0), \mathbb{R}_{\geq 0}, \pi)$, where $S = L \times \mathbb{R}_{\geq 0}^{\mathbb{X}} \times \mathbb{R}$ so that a state consists of a location, a clock valuation, and the accumulated cost; and $((l, v, c), d, \mu) \in \pi$ if one of the following conditions holds:*

- *time transitions:* $d > 0$ and $\mu(l, v + d, c + \dot{\$}(l)d) = 1$
- *discrete transitions:* $d = 0$ and there exists $(l, g, p) \in \text{edges}$ such that $v \models g$ and for any $(l', v', c) \in S$: $\mu(l', v', c) = \sum_{r \subseteq \mathbb{X} \wedge v' = v[r:=0]} p(r, l')$

Definition 4 (CBPR). *Given PPTA $\text{Aut} = (L, l_{\text{init}}, \mathbb{X}, \text{edges}, \dot{\$})$, cost-bounded probabilistic reachability asks the question: “It is possible to reach location $l_G \in L$ with probability at least $p \in (0, 1]$ and with cost at most $\kappa \in \mathbb{N}$.”, denoted $\exists P_{\geq p} F^{\leq \kappa} l_G$. It holds iff there exists a policy A of $\text{MDP}(\text{Aut})$ such that*

$$\text{Prob}^A \{ \omega \in \text{Runs}^A \mid \exists i \in \mathbb{N}. \omega_i \in \{l_G\} \times \mathbb{R}_{\geq 0}^{\mathbb{X}} \times (-\infty, \kappa] \} \geq p$$

3 Undecidability Results

Our undecidability results hold for restricted PPTA, called *simple PPTA*.

Definition 5 (Simple PPTA). *We call a PPTA $\text{Aut} = (L, l_{\text{init}}, \mathbb{X}, \text{edges}, \dot{\$})$ simple if the resolution of probabilities does not influence the set of clocks being reset: $\forall (l, g, p) \in \text{edges}. \exists r \in 2^{\mathbb{X}}. \forall r' \in 2^{\mathbb{X}}. \forall l' \in L. p(r', l') > 0 \implies r' = r$.*

Theorem 1. *CBPR of simple PPTA with three clocks and $\dot{\$} : L \rightarrow \{0, 1\}$ (stop-watch cost) is undecidable.*

Theorem 2. *CBPR of simple PPTA is undecidable even with two clocks.*

Theorem 3. *CBPR of simple PPTA with three clocks and $\dot{\$} : L \rightarrow \mathbb{N}_{>0}$ (strictly positive cost-rates) is undecidable.*

Note that our definition of policy is *deterministic*: a run is mapped to exactly one distribution. There exist other classes of policies, for which the undecidability results will hold in case the class allows the deterministic policies we have used.

The rest of this work contains the proofs. Sections 3.1–3.6 give the proof of Theorem 1. Sections 3.7 and 3.8 prove Theorems 2 and 3, respectively.

3.1 Proof of Theorem 1

Definition 6. A 2-counter Minsky machine [9] is a computational model, consisting of a finite sequence of instructions, labeled l_1, l_2, \dots, l_H . Computation starts at l_1 and halts at l_H . Instructions l_1, \dots, l_{H-1} are of the following two types, where $c \in \{a, b\}$ is one of the counters:

increment c $l_i : c := c + 1; \text{ goto } l_j;$
test-and-decrement c $l_i : \text{if } c = 0 \text{ then goto } l_k;$
 $\text{else } c := c - 1; \text{ goto } l_j;$

We will encode the halting problem for 2-counter Minsky machine \mathcal{M} using a PPTA Aut with a special goal location l_G that satisfies the following property:

$$\exists \mathbb{P}_{\geq 1} F^{\leq 8} l_G \text{ holds for Aut} \iff \neg(\mathcal{M} \text{ terminates})$$

Aut has one location for each instruction label l_1, \dots, l_H . Each transition, when taken at the correct time, corresponds to the execution of one instruction. After the transition, a test may check whether the right edge was taken at the correct time. There is a unique policy that chooses the correct time and edge in every state and so simulates the execution of \mathcal{M} ; we call it the *fulfilling policy*. Any other policy will fail some test, which implies that it misses l_G or the cost bound with positive probability. So, the fulfilling policy is the only one that may satisfy CBPR. However, if \mathcal{M} terminates, it leads to l_H with positive probability, so the maximal probability to reach l_G is still < 1 . It is well-known that termination of a 2-counter machine is undecidable, implying Theorem 1.

Aut uses only 3 clocks x, y, z ; it is not simple, and it allows resets of the form $x:=y$, where clock x is set to the value of clock y . Section 3.6 shows how Aut can be changed to a simple PPTA with only resets to zero.

Upon entering location l_i (under the fulfilling policy), auxiliary clock $y = 0$, and the values of the counters a and b are encoded by x as: $x = 2^{-a} \cdot 3^{-b}$. A value for x uniquely determines a and b . Since both counters start at 0, we have initial location l_0 with an edge to l_1 guarded by $x = 1$ and reset $y:=0$.

CBPR for $p < 1$ (e.g., $\text{Aut} \models \exists \mathbb{P}_{\geq 0.7} F^{\leq c} l_G$) is also undecidable. Just add a probabilistic choice to the edge from l_0 : enter l_1 with probability p , and let the remaining probability of $1 - p$ go to l_H .

In the rest of this section we assume a uniform distribution on all edges, and a cost-rate of 0 in every location, unless a different cost-rate is explicitly given. We now discuss the subautomata needed to let the fulfilling policy simulate the 2-counter machine.

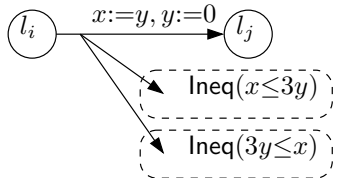


Fig. 2. Automaton for incrementing counter a

3.2 Increment Subautomata

Figure 2 shows the subautomaton for incrementing counter a . We denote the value of x and y upon entering l_i by x_i and y_i , respectively. Assume $x_i = 2^{-a} \cdot 3^{-b}$ (for

some $a, b \in \mathbb{N}$) and $y_i = 0$. (The fulfilling policy guarantees these assumptions.) The automaton ensures that under the fulfilling policy, the value upon entering l_j is $x_j = \frac{1}{2}x_i = 2^{-(a+1)} \cdot 3^{-b}$, which indeed encodes an increment on counter a : Let d_i be the time spent in l_i , and $x_{\text{Ineq}}, y_{\text{Ineq}}$ be the values of the clocks upon entering the Ineq subautomata.

In subautomaton $\text{Ineq}(\varphi)$, l_G is reachable with probability 1 within the cost bound only if φ holds and $0 \leq y_{\text{Ineq}} \leq x_{\text{Ineq}} \leq 2$. Thus the fulfilling policy will only take the edge at a time when $x_{\text{Ineq}} = 3y_{\text{Ineq}}$. Now $y_{\text{Ineq}} = d_i$ and $0 \leq y_{\text{Ineq}} \leq x_{\text{Ineq}}$ due to $y_i = 0$. Since $x_j = y_{\text{Ineq}}$ due to reset $x:=y$, we have:

$$x_i = x_{\text{Ineq}} - d_i = 3y_{\text{Ineq}} - d_i = 3d_i - d_i = 2d_i = 2y_{\text{Ineq}} = 2x_j \tag{1}$$

and $x_{\text{Ineq}} = 3y_{\text{Ineq}} = \frac{3}{2}x_i \leq \frac{3}{2}$. The automaton for incrementing b is the same with the exception that we test for $x_i = 3x_j$ with $\text{Ineq}(x \leq 4y)$ and $\text{Ineq}(4y \leq x)$.

3.3 Power Subautomata

We now introduce an auxiliary automaton called $\text{Power}(k)$. The fulfilling policy will multiply x with a power of $k \in \mathbb{N}$. In particular, a concatenation of $\text{Power}(2)$ and $\text{Power}(3)$ is used to check whether x has the form $2^{-a} \cdot 3^{-b}$: under the fulfilling policy x is doubled a times, leading to $x = 3^{-b}$, and then tripled b times, leading to $x = 1$. If x does not have the required form, it is impossible to reach $x = 1$.

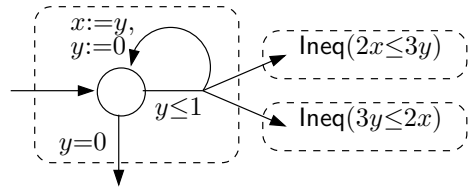


Fig. 3. $\text{Power}(2)$: automaton for multiplying x a number of times by 2

Figure 3 shows $\text{Power}(2)$. The number of times x is doubled is the number of times the loop is taken. The guard $y \leq 1$ excludes a policy that always doubles and never takes the exit edge. Such a policy would pass a test with probability 1, because the probability to stay in the loop indefinitely is 0.

Let x^i be the value of x when entering the location for the i -th time. A similar argument as for Eq. 1 shows that $x^i = \frac{1}{2}x^{i+1}$. The power automaton can only be left with $x = x^i$ for some i , because of the guard $y = 0$, so x upon leaving the power automaton is $2^{i-1} \cdot x^1$. For $\text{Power}(3)$ and $\text{Power}(5)$ (used later), the corresponding tests are $3x = 4y$ and $5x = 6y$, respectively.

3.4 Decrement Subautomata

Figure 4 shows the subautomaton for test-and-decrement of counter a . In location l_i , the fulfilling policy takes the edge from l_i to l_k only if $a = 0$, because the test branch only succeeds if x_i has the form $2^0 \cdot 3^{-b}$. Below, we will see that the fulfilling policy takes the other edge only if $a > 0$.

Decrementing a is very similar to incrementing counters. With the same notations as in Sect. 3.2, assume $x_i = 2^{-a} \cdot 3^{-b}$ (for some $a, b \in \mathbb{N}$) and $y_i = 0$. Then, the Ineq subautomata ensure that the fulfilling policy lets $x_i = \frac{1}{2}x_j$.

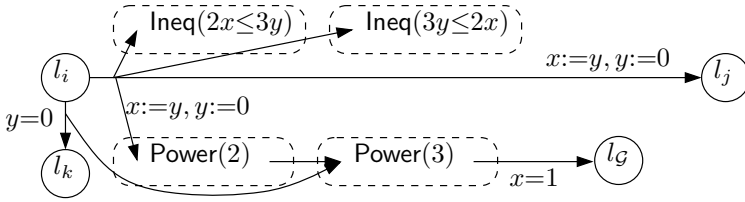


Fig. 4. Automaton for test-and-decrement of counter a

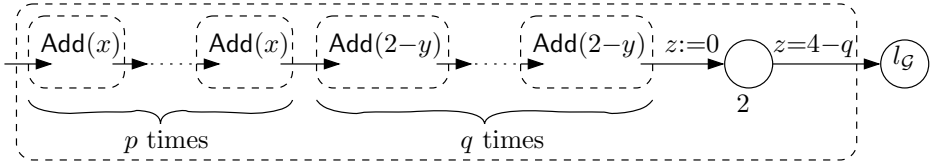


Fig. 5. $\text{Ineq}(px \leq qy)$: automaton for testing $px \leq qy$

The fulfilling policy will take the edge from l_i to l_j only if $a \neq 0$. Otherwise, assume the edge is taken while $a = 0$, then $x_i = 3^{-b}$. Recall that the fulfilling policy will ensure that $x_j = 2x_i = 2 \cdot 3^{-b}$. But then the branch leading from l_i to $\text{Power}(2)$ will not reach l_G , since it would have to divide x by 2.

The construction for test-and-decrement of counter b is very similar: on the edge from l_i to l_k , one would test for $x_i = 2^{-a} \cdot 3^0$, and on the edge from l_i to l_j , one would test for $3x = 4y$.

3.5 Ineq Subautomata

Figure 5 shows the $\text{Ineq}(px \leq qy)$ subautomaton. Location l_G is reached within the cost bound of 8 under a policy only if the clocks satisfy $px \leq qy$ and $0 \leq y \leq x \leq 2$ upon entering Ineq . Subautomata $\text{Add}(x)$ and $\text{Add}(2-y)$ are used to add x respectively $2-y$ to the accumulated cost of a run under any policy that enters and exits the subautomaton, while all clocks have the same values upon exiting as upon entering. The accumulated cost when entering l_G is:

$$px + q(2 - y) + (4 - q)2 = px - qy + 8$$

So l_G is reached within the cost bound of 8 only if $px \leq qy$.

Figure 6 depicts $\text{Add}(x)$. The automaton has the same effect as in [6]. Subautomaton $\text{Add}(2-x)$ is easily obtained by swapping the cost-rates 0 and 1. The reader easily verifies the needed effect of passing through $\text{Add}(x)$ or $\text{Add}(2-x)$.

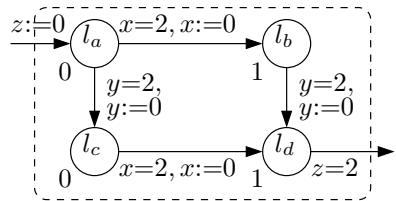


Fig. 6. $\text{Add}(x)$: automaton adding x to the accumulated cost

¹ $\text{Add}(x)$ in [6] contains a glitch: when $y = 1$ on entrance, possibly $y = 0$ on exit.

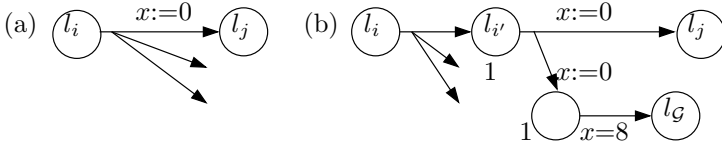


Fig. 7. Removing probabilistic resets

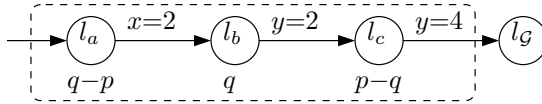


Fig. 8. $\text{Ineq}(px \leq qy)$: automaton for testing $px \leq qy$

3.6 Adaption to Simple PPTA

To render the PPTA simple, we change the encoding as follows. The resets $x:=y, y:=0$ are replaced by $x:=0$. This swaps the role of x and y in the target location, i.e. y now encodes the counters. The fact that the clocks are swapped in some location will be captured by a copy of that location, where x and y are swapped on all guards and resets of outgoing edges.

The obtained PPTA still has resets $x:=0$ that depend on the resolution of probability. Figure 7a shows such an edge, and Fig. 7b shows how we can replace it using an intermediate location $l_{i'}$ and a reset that does not depend on the resolution of probability. The fulfilling policy will not let time advance in $l_{i'}$, because this incurs cost, and upon leaving $l_{i'}$, a test may be invoked to check whether the cost incurred up to that point is still 0.

3.7 Proof of Theorem 2

In this section, we allow PPTA to have any positive or negative integer cost rate. This relaxation will allow us to encode the 2-counter machine with two clocks only, because we can simplify the Ineq subautomata.

Figure 8 shows the alternative Ineq subautomaton. The cost-bound of the CBPR problem is changed to 0. (It may happen that a run exceeds the cost bound temporarily; however, upon entry into l_G , its cost has to be ≤ 0 .) Let d_a, d_b, d_c denote the time that elapses in locations l_a, l_b, l_c respectively. Let x_a, y_a, c_a denote the values of the clocks and accumulated cost when entering l_a . A run that reaches l_G has the following accumulated cost:

$$c_a + (q - p)d_a + qd_b + (p - q)d_c \tag{2}$$

Since all the locations a run visits before entering l_a have cost-rate 0 we have $c_a = 0$. We need to ensure that d_a, d_b, d_c are nonnegative (under the fulfilling policy). $d_a = 2 - x_a$, and non-negativity follows from the fact that when $x_a > 1$ the encoding of the counters is incorrect, which is only possible under a non-fulfilling policy. $d_b = 2 - (y_a + d_a) = x_a - y_a$, and non-negativity follows from the fact

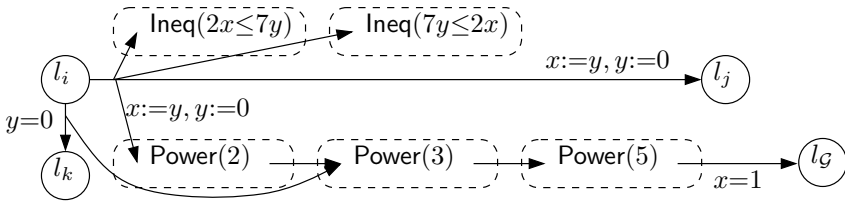


Fig. 9. New automaton for test-and-decrement of counter a

that $y_a \leq x_a$ whenever `Ineq` is entered. Clearly $d_c = 2$. By filling in Eq. 2 we get the following accumulated cost: $(q-p)(2-x_a) + q(x_a-y_a) + (p-q)2 = px_a - qy_a$. Therefore, `Ineq`($px \leq qy$) reaches l_G with cost ≤ 0 iff $px \leq qy$ upon entering.

3.8 Proof of Theorem 3

We now want to construct a simple PPTA with only strictly positive cost rates. As a starting point, we take the PPTA obtained in the previous section. We will again add a third clock z , but now, z is never reset, so it equals the duration of the run in all states.

The PPTA is adapted by adding 6 to all cost-rates. For all locations that had cost-rate 0 this clearly enforces a strictly positive cost-rate. The only negative cost-rates appear in `Ineq` subautomata (Fig. 8), but they are all larger than -6 , so the new rates are all strictly positive.

All runs of the fulfilling policy that reach l_G should have an accumulated cost below the cost bound of the cost-bounded reachability problem. Because of the strictly positive cost-rate, we therefore need an overall time bound for all these runs, which we will show later to be 9. To accommodate the time bound, next to the counters a and b , clock x will encode the integer n , which is used to count the number of times a test-and-decrement instruction decremented any of the two counters. The encoding becomes: $x = 2^{-a} \cdot 3^{-b} \cdot 5^{-n}$.

The new test-and-decrement automaton is shown in Fig. 9. The values for the two `Ineq` automata are changed to accommodate that on entering l_j : $x_j = \frac{2}{5}x_i$ (which corresponds to decrementing a and incrementing n .) From `Power(3)` there is now an edge to `Power(5)` which has the edge guarded by $x = 1$ to l_G . The `Power(5)` automaton is needed here, because this part was used to check the correctness of the encoding by x , which now includes the factor 5^{-n} .

The final change to the total automaton is that on every run where l_G was entered, the run now has to pass by a new location l'_G . Figure 10 depicts l'_G and how from there l_G is reachable. Because z measures the duration of a run, which is bounded by 9 (as explained below), and time is spent in l'_G until z becomes 9, the additional cost for any run is: $9 \cdot 6 = 54$. Indeed the cost bound for the CBPR problem is changed to 54.

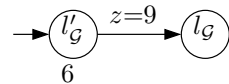


Fig. 10. Subautomaton to reach l_G in exactly 9 time units

We will now show that every run that enters l'_G has a duration bounded by 9. First of all 1 time unit is spent in l_0 . Under the fulfilling policy, as long as

no `Ineq` or `Power` subautomaton is entered, every passage through an increment or decrement subautomaton multiplies x with $\frac{1}{2}$, $\frac{1}{3}$, $\frac{2}{5}$ or $\frac{3}{5}$, so the new value of x is at most $\frac{3}{5}$ times its old value. Further, the time spent in some subautomaton is equal to the new value of x . (If in a test-and-decrement subautomaton, the tested counter is $= 0$, then x is not changed and no time is spent in the subautomaton, so we can ignore this case in the runtime calculation.) Therefore, the total runtime until entering some `Power` or `Ineq` automaton is less than $1 + \sum_{i=1}^{\infty} (\frac{3}{5})^i = 2\frac{1}{2}$.

Similarly, one can see that each iteration in a concatenation of `Power` subautomata takes at most $\frac{1}{2}$ times the time of the next iteration, and the last iteration (all under the fulfilling policy) takes time 1. Therefore, the maximal time spent in `Power` subautomata is $\sum_{i=0}^{\infty} (\frac{1}{2})^i = 2$.

Finally, an `Ineq` subautomaton takes at most 4 time units. Summing up, we get a total upper bound on the runtime of $2\frac{1}{2} + 2 + 4 \leq 9$ upon entering l_G .

References

1. Alur, R., Dill, D.L.: A theory of timed automata. *Theoretical Computer Science* 126(2), 183–235 (1994)
2. Alur, R., Torre, S.L., Pappas, G.J.: Optimal paths in weighted timed automata. In: Di Benedetto, M.D., Sangiovanni-Vincentelli, A.L. (eds.) *HSCC 2001*. LNCS, vol. 2034, pp. 49–62. Springer, Heidelberg (2001)
3. Behrmann, G., Fehnker, A., Hune, T., Larsen, K.G., Pettersson, P., Romijn, J., Vaandrager, F.W.: Minimum-cost reachability for priced timed automata. In: Di Benedetto, M.D., Sangiovanni-Vincentelli, A.L. (eds.) *HSCC 2001*. LNCS, vol. 2034, pp. 147–161. Springer, Heidelberg (2001)
4. Berendsen, J., Jansen, D.N., Katoen, J.P.: Probably on time and within budget: On reachability in priced probabilistic timed automata. In: *QEST*, pp. 311–322. IEEE Computer Society Press, Los Alamitos (2006)
5. Kwiatkowska, M., Norman, G., Segala, R., Sproston, J.: Automatic verification of real-time systems with discrete probability distributions. *Theoretical Computer Science* 282(1), 101–150 (2002)
6. Bouyer, P., Brihaye, T., Markey, N.: Improved undecidability results on weighted timed automata. *Inf. Process. Lett.* 98(5), 188–194 (2006)
7. Bouyer, P., Fahrenberg, U., Larsen, K.G., Markey, N., Srba, J.: Infinite runs in weighted timed automata with energy constraints. In: Cassez, F., Jard, C. (eds.) *FORMATS 2008*. LNCS, vol. 5215, pp. 33–47. Springer, Heidelberg (2008)
8. Kemeny, J., Snell, J., Knapp, A.: *Denumerable Markov Chains*, 2nd edn. Springer, New York (1976)
9. Minsky, M.L.: *Computation: finite and infinite machines*. Prentice-Hall, Inc., Upper Saddle River (1967)

A Computational Proof of Complexity of Some Restricted Counting Problems

Jin-Yi Cai^{1,*}, Pinyan Lu², and Mingji Xia^{1,3}

¹ Computer Sciences Department, University of Wisconsin
Madison, WI 53706, USA
jyc@cs.wisc.edu

² Microsoft Research Asia
Beijing, 100190, P.R. China
lupinyan@gmail.com

³ State Key Laboratory of Computer Science,
Institute of Software, Chinese Academy of Sciences
Beijing 100190, P.R. China
xmjljx@gmail.com

Abstract. We explore a computational approach to proving *intractability* of certain counting problems. More specifically we study the complexity of Holant of 3-regular graphs. These problems include concrete problems such as counting the number of vertex covers or independent sets for 3-regular graphs. The high level principle of our approach is algebraic, which provides sufficient conditions for *interpolation* to succeed. Another algebraic component is *holographic reductions*. We then analyze in detail polynomial maps on \mathbb{R}^2 induced by some combinatorial constructions. These maps define sufficiently complicated *dynamics* of \mathbb{R}^2 that we can only analyze them computationally. We use both numerical computation (as intuitive guidance) and symbolic computation (as proof theoretic verification) to derive that a certain collection of combinatorial constructions, in myriad combinations, fulfills the algebraic requirements of proving #P-hardness. The final result is a dichotomy theorem for a class of counting problems.

1 Introduction

In this paper we study some counting problems which can be described in the following way. We are given a graph $G = (V, E)$. At each vertex $v \in V$ there is a function f_v , and at each edge $e \in E$ there is a function g_e . We also call these functions f_v and g_e *signatures*. These functions take 0-1 inputs and output real values in \mathbb{R} . Now consider all 0-1 assignments σ at each *end* of every edge $e = (x, y)$, i.e., a value $\sigma(e, x)$ and $\sigma(e, y)$. The *counting problem* is to compute $\sum_{\sigma} \prod_v f_v(\sigma|_v) \prod_e g_e(\sigma|_e)$, where the sum is over all 0-1 assignments σ of products of function evaluations over all $v \in V$ and $e \in E$. Here $\sigma|_v$ denotes the values assigned locally by σ at v , i.e., the ends of all edges incident to v , and $\sigma|_e$

* Supported by NSF CCF-0830488 and CCF-0511679.

denotes the values assigned by σ at the two ends of e . If each f_v is the EQUALITY function (of arity = $\deg(v)$), then σ can be thought of as 0-1 assignments over the vertex set V . Similarly if each g_e is the EQUALITY function (of arity two), then σ can be taken as 0-1 assignments over E .

For example, choosing EQUALITY for every edge, the problems of counting matchings or perfect matchings correspond to taking the AT-MOST-ONE or EXACT-ONE function at each vertex, respectively. Similarly counting all vertex covers on a 3-regular graph $G = (V, E)$ corresponds to choosing f_v to be the EQUALITY function of arity three, and g_e the OR function on two inputs. Yet another example is Independent Set, which corresponds to EQUALITY for f_v and AT-MOST-ONE for g_e . This framework of counting problems is called Holant problems [45], and in general the assignments σ can take values in any finite set $[q]$. Classically, when f_v is fixed to be EQUALITY, and each edge is given the same Boolean function (but σ takes values in $[q]$) this problem is known as graph homomorphism problem (or H -colorings or H -homomorphisms, or partition functions) [10,11,12,13]. Here H is a fixed directed or undirected graph (with possible self loops) given by a $q \times q$ Boolean adjacency matrix. A mapping $\sigma : V(G) \rightarrow V(H)$ is a homomorphism iff for every edge $(x, y) \in E(G)$, $H(\sigma(x), \sigma(y)) = 1$. Then the quantity $\sum_{\sigma} \prod_{(x,y) \in E(G)} H(\sigma(x), \sigma(y))$ counts the number of H -homomorphisms. Vertex cover is the special case where the two-vertex graph H is $(\{0, 1\}, \{(0, 1), (1, 0), (1, 1)\})$. Dichotomy theorems (i.e., the problem is either in P or #P-hard, depending on H) for H -homomorphisms with undirected graphs H and directed acyclic graphs H are given in [11] and [10] respectively. H -homomorphisms can also be studied for more general functions than Boolean valued functions. A dichotomy theorem for any symmetric matrix H with non-negative real entries is proved in [2]. Very recently Goldberg et. al. in a most impressive 73-page paper [12] have proved a dichotomy theorem for any real symmetric matrix H . We will make use of these results [2,12].

Another related incarnation of these problems is known as Constrained Satisfaction Problems (CSP) [17,8,9]. In a Boolean CSP, there is a set of Boolean variables represented by vertices U on the left hand side (LHS) of a bipartite graph (U, W, E) . The right hand side (RHS) vertices W represent constraint functions. It is usually implicitly assumed that each vertex in U is labeled by an EQUALITY function and each vertex in W is labeled by a constraint function. Thus EQUALITY of arbitrary arity is implicitly assumed to exist in input instances. If each vertex $w \in W$ is of degree 2 and is assigned the same function, then effectively we can treat w as “an edge” (by merging the two edges incident to w), and we return to the setting of H -homomorphisms. Furthermore if each $u \in U$ has degree 3 then this is effectively a 3-regular graph. We call a bipartite graph (U, W, E) 2-3 regular if $\deg(u) = 3$ and $\deg(w) = 2$ for $u \in U$ and $w \in W$. As indicated, this encompasses 3-regular graphs. It turns out that if EQUALITY gates of arbitrary arity are freely available in possible input graphs then it is technically easier to prove #P-hardness. For Holant problems the EQUALITY gates are not freely available unless explicitly given, proofs of #P-hardness become more challenging, because we are more constricted to design

gadgets in possible reductions. Furthermore there are indeed cases within this class of counting problems where the problem is $\#P$ -hard for general graphs, but solvable in P when restricted to 3-regular (or 2-3 regular) graphs.

In this paper we consider 2-3 regular graphs (U, W, E) where each $u \in U$ is assigned the EQUALITY function (of arity 3) and each $w \in W$ is assigned a real symmetric function on two bits. We denote a symmetric function on n bits as $[f_0, f_1, \dots, f_n]$ where f_i is the value of the function on inputs of Hamming weight i . Then our problem can be denoted as $\#[1, 0, 0, 1] \mid [x_0, x_1, x_2]$ for some $x_0, x_1, x_2 \in \mathbb{R}$. Our main result in this paper is a complexity dichotomy theorem for this class of problems.

It turns out that studying counting problems in this framework has a close connection with holographic algorithms and reductions [16]. One can transform the general counting problem $\#[y_0, y_1, y_2, y_3] \mid [x_0, x_1, x_2]$ on 2-3 regular graphs for any pair of symmetric functions to either $\#[1, 0, 0, 1] \mid [z_0, z_1, z_2]$ or $\#[1, 1, 0, 0] \mid [z_0, z_1, z_2]$ by holographic reductions [4]. In [4] a dichotomy theorem was shown for all problems in this class where x_i and y_j are 0-1 valued. The two cases $\#[1, 0, 0, 1] \mid [z_0, z_1, z_2]$ or $\#[1, 1, 0, 0] \mid [z_0, z_1, z_2]$ correspond to a certain characteristic polynomial having distinct roots or double roots, with the first case being the generic case of distinct roots. Thus our problem $\#[1, 0, 0, 1] \mid [x_0, x_1, x_2]$ for $x_0, x_1, x_2 \in \mathbb{R}$ corresponds to the case with two distinct real characteristic roots. By holographic reductions our dichotomy theorem for $\#[1, 0, 0, 1] \mid [x_0, x_1, x_2]$ has extensions to more general forms. The framework of Holant problems was formally introduced in our previous work; we refer to [4, 5] for formal definitions and notations. The problems studied in this paper are a very restricted class, over 2-3 graphs, but we find it the simplest class for which it is still non-trivial to prove a dichotomy theorem. It is also a simple class which includes some interesting combinatorial problems. Compared to CSP problems, generally it is more difficult to prove dichotomy theorems for Holant problems where equality gate is not for free. As shown in [5], the 2-3 regular graphs are the most basic and also technically the most difficult cases. The dichotomy theorems of general cases in [5] are reduced to these 2-3 regular ones.

The absence of EQUALITY gates of arbitrary arity in problem specification is a real hindrance to proving $\#P$ -hardness. Proofs of previous dichotomy theorems make extensive use of constructions called thickening, stretching and pinning. Unfortunately all these constructions require the availability of EQUALITY gates of arbitrary arity to carry out.

Our approach is to reduce H -homomorphism problems (where vertices take 0-1 values) to our problem. The former is known to be $\#P$ -hard. This amounts to proving the reduction

$$\#\{=1, =2, =3, \dots, =k, \dots\} \mid [x_0, x_1, x_2] \leq_T \#[1, 0, 0, 1] \mid [x_0, x_1, x_2].$$

We use a set of signatures on one side to mean that any signature from that set can be used for vertices on that side of the bipartite graph. The desired EQUALITY gates $\{=1, =2, =3, \dots, =k \dots\}$ will be “produced” by *simulation* in a chain of reductions.

The main effort of this paper is to prove that a suitable collection of combinatorial constructions succeed in the aggregate. We give a “computational” proof of this fact. The constructions give rise to a set of polynomial maps on \mathbb{R}^2 . These maps define sufficiently complicated *dynamics* of \mathbb{R}^2 that we can only analyze them computationally. We use computation in our investigations in two separate ways. First we use numerical computation (mainly Matlab™) to guide our choice and pruning of combinatorial designs. Second we use symbolic computation (mainly CylindricalDecomposition in Mathematica™) to produce proofs about semi-algebraic sets. Along the way many “engineering” approaches were needed to coax symbolic computation to produce a definite result.

2 A Dichotomy Theorem and Reduction Chain

Our main theorem is the following dichotomy theorem.

Theorem 1. *The counting problem $\#[1, 0, 0, 1] \mid [x_0, x_1, x_2]$ is $\#P$ -hard unless one of the following conditions holds: (1) $x_1^2 = x_0x_2$; (2) $x_0 = x_2 = 0$ or $x_1 = 0$; (3) $x_0 = x_1 = -x_2$ or $x_0 = -x_1 = -x_2$; the problem $\#[1, 0, 0, 1] \mid [x_0, x_1, x_2]$ is polynomial time computable in these three cases.*

We remark that if we restrict ourselves to planar graphs, there is a 4th category of tractable cases $x_0 = x_2$, which can be solved in polynomial time by holographic algorithms [16,3].

If $x_1 = 0$, the problem is easily computable in polynomial time. So we consider the case $x_1 \neq 0$, and by a scalar factor, we can assume $x_1 = 1$. Then the problem $\#[1, 0, 0, 1] \mid [a, 1, b]$ can be described by a point (a, b) in the real plane \mathbb{R}^2 .

Now we give a chain of reductions. For any (a, b) such that $ab \neq 1$ and $(a, b) \notin \{(0, 0), (1, -1), (-1, 1)\}$, the problem $\#\{=1, =2, =3, \dots, =k, \dots\} \mid [a, 1, b]$ is $\#P$ -hard, while for all the exceptional cases, the problem is tractable in P [2,12]. The tractability of our problem follows from this. To show $\#P$ -hardness we use the following chain of reductions,

$$\begin{aligned} \#\{=1, =2, =3, \dots, =k, \dots\} \mid [a, 1, b] &\leq_T \#[1, 0, 0, 1] \mid \{[a, 1, b], [1, 0, 1]\} & (1) \\ &\leq_T \#[1, 0, 0, 1], [1, 0, 1] \mid [a, 1, b] & (2) \\ &\leq_T \#[1, 0, 0, 1] \mid \{[a, 1, b], [1, 1]\} & (3) \\ &\leq_T \#[1, 0, 0, 1] \mid [a, 1, b]. & (4) \end{aligned}$$

The goal of this reduction chain is to “simulate” EQUALITY of arbitrary arity. Step (1) is easy. With $[1, 0, 1]$ on the RHS and $[1, 0, 0, 1]$ on the LHS we can simulate any $=_k$. To prove step (2) we use the following lemma. It shows that if we have $[1, 0, 1]$ on the LHS, we can do stretching and interpolate $[1, 0, 1]$ on the RHS. It is a special case of Lemma 3.4 in [11] by Dyer and Greenhill:

Lemma 1. *If $ab \neq 1$ and \mathcal{F} is a set of signatures, then*

$$\#\mathcal{F} \mid \{[a, 1, b], [1, 0, 1]\} \leq_T \#\mathcal{F} \cup \{[1, 0, 1]\} \mid [a, 1, b].$$

However we don't have the signature $[1, 0, 1]$ on LHS yet. The way we accomplish this is to realize a unary signature $[1, 1]$ on the RHS. If we have $[1, 1]$ on the RHS, we can realize $[1, 0, 1]$ on the LHS by the small gadget in Fig. 1, which proves step (3).

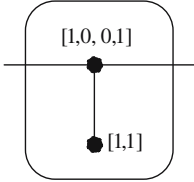


Fig. 1. A small gadget

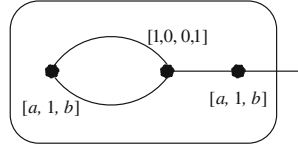


Fig. 2. A gadget for $[1, 1]$ when $a = b$

Then, the main task is step (4): to realize $[1, 1]$ on the RHS. If $a = b \notin \{0, -1\}$, we can realize $[1, 1]$ by the gadget in Fig. 2. The signature of the \mathcal{F} -gate is $[a^2 + a, a^2 + a]$, and we can take the common factor $a^2 + a \neq 0$ out to get $[1, 1]$ given that $a \notin \{0, -1\}$. Note that $a = b = 0$ or $a = b = -1$ fall in the tractable cases in Theorem 1.

If $a \neq b$, we do not know how to realize $[1, 1]$ directly for a generic pair (a, b) . However, it turns out that we can interpolate all the unary functions on the RHS. This is our main lemma in this paper.

Lemma 2. *If $ab \neq 1$, $a \neq b$ and $(a, b) \notin \{(1, -1), (-1, 1)\}$, then*

$$\#[1, 0, 0, 1] \mid \{[a, 1, b]\} \cup \mathcal{U} \leq_T \#[1, 0, 0, 1] \mid [a, 1, b],$$

where \mathcal{U} denotes the set of all unary signatures.

We note that when $a = b$, the reduction in Lemma 2 does not hold. So the case $a = b$ must be handled separately as above.

3 Interpolation Method

In this section, we discuss the interpolation method we will use for our main lemma. Polynomial interpolation is a powerful tool in the study of counting problems initiated by Valiant and further developed by Vadhan, Dyer and Greenhill [15, 11]. We want to show that for all unary signatures $f = [x, y]$, we have $\#[1, 0, 0, 1] \mid \{[a, 1, b], [x, y]\} \leq_T \#[1, 0, 0, 1] \mid [a, 1, b]$, under some conditions on a and b . Let $\Omega = (G, [1, 0, 0, 1] \mid \{[a, 1, b], [x, y]\})$ be a signature grid. Here $[x, y]$ appears on the RHS. We want to compute Holant_Ω in polynomial time using an oracle for $\#[1, 0, 0, 1] \mid [a, 1, b]$.

Let V_f be the subset of vertices in G assigned f in Ω . Suppose $|V_f| = n$. These vertices are on the RHS, all with degree 1, and all connected to some vertex on the LHS of degree 3. We can classify all 0-1 assignments σ in the holant sum

according to how many vertices in V_f whose incident edge is assigned a 0 or a 1. Then the holant value can be expressed as

$$\text{Holant}_\Omega = \sum_{0 \leq i \leq n} c_i x^i y^{n-i}, \tag{5}$$

where c_i is the sum over all edge assignments σ , of products of evaluations at all $v \in V(G) - V_f$, where σ is such that exactly i vertices in V_f have their incident edges assigned 0 (and $n - i$ have their incident edges assigned 1.) If we can evaluate these c_i , we can evaluate Holant_Ω .

Now suppose $\{G_s\}$ is a sequence of \mathcal{F} -gates using signature pairs $[1, 0, 0, 1] \mid [a, 1, b]$. Each G_s has one dangling edge which is to be connected externally to a vertex of degree 3. Denote the signature of G_s by $f_s = [x_s, y_s]$, for $s = 0, 1, \dots$. If we replace each occurrence of f by f_s in Ω we get a new signature grid Ω_s on signature pairs $[1, 0, 0, 1] \mid [a, 1, b]$ with

$$\text{Holant}_{\Omega_s} = \sum_{0 \leq i \leq n} c_i x_s^i y_s^{n-i}. \tag{6}$$

One can evaluate Holant_{Ω_s} by oracle access to $\#[1, 0, 0, 1] \mid [a, 1, b]$. Note that the same set of values c_i occurs. We can treat c_i in (6) as a set of unknowns in a linear system. The idea of interpolation is to find a suitable sequence $\{f_s\}$ such that the evaluation of Holant_{Ω_s} gives a linear system (6) of full rank, from which we can solve for all c_i .

In this paper, the sequence $\{G_s\}$ will be constructed recursively using suitable gadgetry. There are two gadgets in a recursive construction: one gadget has arity 1, giving the initial signature $g = [x_0, y_0]$; the other has arity 2, giving the recursive iteration. It is more convenient to use a 2×2 matrix A to denote it. We remark that the dangling edge of the arity 1 gadget is expected to connect externally to a vertex of degree 3; from the gadget of arity 2, one dangling edge is to be connect externally to a vertex of degree 3 and the other to a vertex of degree 2. So we can recursively connect them as in Figure 3 and get $\{G_s\}$.

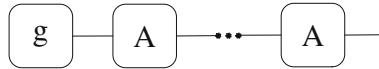


Fig. 3. Recursive construction

The signatures of $\{G_s\}$ have the relation $\begin{bmatrix} x_s \\ y_s \end{bmatrix} = A \begin{bmatrix} x_{s-1} \\ y_{s-1} \end{bmatrix}$, where $A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$ and $g = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}$. In the following, we will call this gadget pair (A, g) the recursive construction. It follows from lemma 6.1 in [15] that

Lemma 3. *Let α, β be the two eigenvalues of A . If $\det(A) \neq 0$, g is not a column eigenvector of A (nor the zero vector), and α/β is not a root of unity, then the recursive construction (A, g) can be used to interpolate all the unary signatures.*

Notice that both A and g are functions of (a, b) . Here we relax the condition that $\frac{\alpha}{\beta}$ is not a root of unity to $|\frac{\alpha}{\beta}| \neq 1$ so that all the conditions can be described by polynomial equalities or inequalities of (a, b) . We denote by $[Ag, g]$ the 2×2 matrix with first column Ag and second column g . Then $\det[Ag, g] = 0$ is equivalent to g being a column eigenvector of A (or the zero vector). If $\text{tr}A \neq 0$ and the discriminant $(\text{tr}A)^2 - 4 \det(A) > 0$, then it is easy to see that the two eigenvalues α and β have unequal norms.

Definition 1. *The failure set of a recursive construction (A, g) is the following semi-algebraic set $\mathcal{F}(A, g)$:*

$$[\det(A) = 0] \text{ or } [\text{tr}A = 0] \text{ or } [(\text{tr}A)^2 - 4 \det(A) \leq 0] \text{ or } [\det[Ag, g] = 0] .$$

Denote by

$$E = \{(a, b) \in \mathbb{R}^2 \mid ab = 1 \text{ or } a = b \text{ or } (a, b) = (1, -1) \text{ or } (a, b) = (-1, 1)\},$$

the exceptional cases of Lemma 2. We prove there are a finite number of gadgets (A_i, g_i) , where $i = 1, 2, \dots, C$, such that

$$\bigcap_i \mathcal{F}(A_i, g_i) \subseteq E. \tag{7}$$

4 Computational Proof

We prove Lemma 2 by establishing (7). We will give an account of the many steps taken to overcome various difficulties. Some of the difficulties are not of a logical nature, but a matter of computational complexity, in a practical sense. It will be a combination of mathematical derivation (in a traditional sense) and an “engineering” undertaking. We find it amusing that we must contend with practical computational complexity in proving theorems of computational complexity.

We will construct some combinatorial gadgets. As described earlier each gadget will depend on two components: an initiation component and an iterative component. We start with the simplest gadget in Fig 4 and Fig 5. (In our figures, unless otherwise specified, all vertices of degree 3 and 2 have signatures $[1, 0, 0, 1]$ and $[a, 1, b]$, respectively.)

The gadget in Fig 4 has a unary signature $[a^2+b, a+b^2]$, and must be externally connected to a vertex of degree 3. The gadget in Fig 5 has one dangling edge connecting to a degree 2 vertex and another dangling edge connecting to a degree 3 vertex. The matrix A for the gadget in Fig 5 is $\begin{bmatrix} a(a^2+b) & a+b^2 \\ a^2+b & b(a+b^2) \end{bmatrix}$.

We consider the failure set $\mathcal{F}(A, g)$ for this gadget. The inequality is our main concern as the equalities define a lower dimensional set. We now focus on the *main failure set* $\mathcal{F}^*(A) = \{(a, b) \in \mathbb{R}^2 - E \mid (\text{tr}(A))^2 - 4 \det(A) \leq 0\}$. This set is depicted in Fig 6. We remark that this main failure set only depends on the iterative gadget A and in the following discussion we focus on this component.

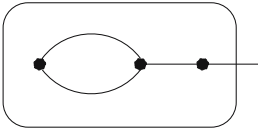


Fig. 4. The initiation component g

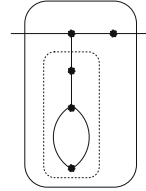


Fig. 5. The iterative component A

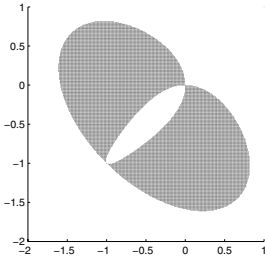


Fig. 6. The Failure set of the first gadget

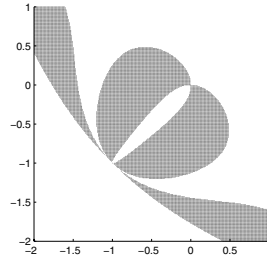
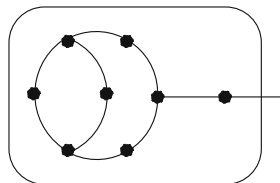


Fig. 7. The failure set using g'

This picture is produced by numerical computation. We will use numerical computation, not as proof, but as intuitive guidance in our search of gadgets. For example, suggested by the numerical computation, after some standard estimation we can prove (traditionally) that this set $\mathcal{F}^*(A)$ is bounded by the square $[-2, 1] \times [-2, 1]$. A consequence of this is that for every $(a, b) \notin [-2, 1] \times [-2, 1]$, except on a lower dimensional set, the problem is already proved to be $\#P$ -hard. Another side benefit of this boundedness is that, going forward, all numerical detective work will be restricted to a bounded region, which would have been hard to do without the boundedness.

Our goal, then, is to somehow shrink this failure set, by finding good gadgets. In Figure 5, the dashed box can be replaced by another unary gadget whose dangling edge is to be connected to a vertex of degree 3. Let this unary gadget have signature $[c, d]$, then the iterative component will have signature $A = \begin{bmatrix} ac & d \\ c & bd \end{bmatrix}$. Thus a natural idea is to design another gadget replacing that part with another unary gadget. Here is another such gadget.



This gadget has the unary signature $g' = [a^5 + 3a^2 + ab^2 + 2b + b^4, b^5 + 3b^2 + ba^2 + 2a + a^4]$, and can be used instead of g in the dashed box. The result of using g' instead of g is illustrated in Figure 7.

If we take numerical computation as trustworthy, then outside of the intersection of Figures 6 and 7, modulo a lower dimensional set, #P-hardness has already been established. Our hope then is to find enough gadgets such that the intersection becomes empty (as a subset of $\mathbb{R}^2 - E$). We note crucially that the intersection, together with the lower dimensional set, is a semi-algebraic set. Thus it is decidable, by Tarski's theorem [14], whether a particular collection of gadgets produces an empty intersection. Thus to prove that for all (a, b) not in the known tractable set the problem is #P-hard, "all we need to do" is to find a sufficient number of gadgets, and apply Tarski's theorem. Of course this plan can only succeed if the statement is actually true, and we can in fact find a finite number of gadgets whose intersection of failure sets is indeed empty. Certainly there is no point to apply Tarski's theorem when numerical computation indicates that the gadgets found so far are manifestly insufficient.

Off we go to hunt for more gadgets. The next idea is to use the iterative construction for a different purpose. We will use the iterative construction now to construct many unary signatures, each to be used inside the dashed box for the original iterative construction. More precisely, if we use g (or g') as the initial unary signature inside the dashed box, and iterate the construction k times, we will obtain k new unary signatures, say, g_1, g_2, \dots, g_k , each of which can be used as the initial signature $[c, d]$ inside the dashed box to start the iterative construction for the purpose of interpolation.

After some numerical computation the evidence is that while the intersection of failure sets gets thinner, these gadgets are still not enough. The next idea is that in our iterative construction for the initial unary signatures, we can use either g or g' in the dashed box interchangeably, per each iteration. Thus, to iterate this process k times, we can produce 2^k initial signatures usable as $[c, d]$, with which to start its own iterative construction for the purpose of interpolation. After some experiment (numerical computation again) we decided that this is not so trivial. And so we started a computer search, with iteration depth $k = 1, 2, \dots$, and with a random choice of g or g' per each iteration.

Our computation reveals that at depth $k = 15$ certain sequences g and g' 's seem to have produced a collection of $[c, d]$'s to start the iterative construction, whose failure sets have an empty intersection. We then hand-pick and prune it down: A particular sequence of 7 copies of g and g' in succession produced a collection, whose failure sets seem to have an empty intersection. All of this is not proved, but strongly suggested by numerical computation.

At this point, it seems that we just need to hand this to Tarski's theorem. But we encountered an unexpected problem. The 7 gadgets produced a well defined semi-algebraic set which is presumably empty and this fact is decidable; however, the emptiness computation simply won't terminate. (In fact it did not terminate even for 6 gadgets.) Thus our problem has turned into a practical misfortune: fewer gadgets are not sufficient (numerical evidence); more gadgets

seem to suffice, but they are too complicated to handle analytically and beyond the capability of symbolic computation. We should note that this insistence on proofs beyond numerical evidence is absolutely necessary; we had many instances in this proof where numerically indicated assertions are actually false.

We next assessed what feature of a gadget appears to have the greatest impact on the practical performance of the decision procedure on semi-algebraic sets. And this appears to be the degree of the polynomial, which translates to the number of degree 2 vertices. For 2-3 regular graphs, this is proportional to the number of degree 3 vertices. We call this number m . By a parity argument, taking into account of dangling edges, it can be shown that m must be even.

So we systematically enumerated all gadgets with $m = 2, 4$ or 6 . There are over 170 gadgets with $m = 6$. On the other hand, symbolic computation cannot even handle a single gadget with $m = 8$ (the computation does not terminate). Yet, numerically even all gadgets with $m = 6$ together do not suffice.

At this point we decided to modify our strategy. Instead of looking for a set of gadgets which will completely cover all the points in $\mathbb{R}^2 - E$, we will settle for a set of gadgets with relatively low degree signature polynomials ($m \leq 6$, so that symbolic computation can handle) and whose failure set is *small* and *easy to delineate*. We search for such gadgets numerically, and once we settle on such a set of gadgets, we will bound it by a *box* defined by piece-wise linear segments such as a triangle. We use symbolic computation to confirm that the box indeed contain the failure set. Outside such boxes the problem is already proved $\#P$ -hard. Then we deal with the boxes separately.

The next idea is somewhat different. We consider an \mathcal{F} -gate with two dangling edges, both to be connected externally to vertices of degree 3. In Figure 8 we depict such an \mathcal{F} -gate. Note that such an \mathcal{F} -gate can replace a vertex of degree 2 everywhere. It can be verified that its signature is $[a^2(a^2 + b) + a + b^2, a(a^2 + b) + b(a + b^2), a^2 + b + b^2(a + b^2)]$. Logically, if the counting problem with the above transformed signature is $\#P$ -hard, then so is the counting problem with $[a, 1, b]$.

The dehomogenized form of the signature map is

$$f_1 : (a, b) \mapsto \left(\frac{a^2(a^2 + b) + a + b^2}{a^3 + 2ab + b^3}, \frac{a^2 + b + b^2(a + b^2)}{a^3 + 2ab + b^3} \right). \tag{8}$$

The subset of (a, b) in a box which is mapped by f_1 to a point within the box is a semi-algebraic set. Because of the relaxation of the original failure set to the box, the defining polynomials of this semi-algebraic set is of relatively low degree. Had we not enlarged the failure set to the box, this degree would have been too high for symbolic computation.

In fact we will need another \mathcal{F} -gate (Fig. 9), where it has the dehomogenized form of transformation

$$f_2 : (a, b) \mapsto \left(\frac{a(a^3 + 1) + a + b^2}{a^3 + ab + 1 + b^3}, \frac{a^2 + b + b(1 + b^3)}{a^3 + ab + 1 + b^3} \right). \tag{9}$$

Note that at least one map of f_1 and f_2 is well defined at every $(a, b) \in \mathbb{R}^2 - E$, since $a^3 + 2ab + b^3$ and $a^3 + ab + 1 + b^3$ simultaneously vanish only at $ab = 1$.

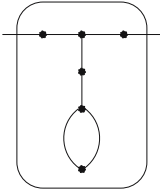


Fig. 8. One edge transformation

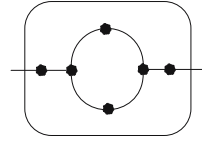


Fig. 9. Another edge transformation

There is another idea that we use to lower the degree, which makes symbolic computation feasible. It can be seen that the various signatures are symmetric functions of a and b . Once we make a change of coordinates, such that the y -axis is the line $a = b$, then the functions all become even functions of x . Also it appears that the point $(a, b) = (-1, -1)$ is where most of the trouble reside (at least numerically). Thus we use the transformation $a = -\sqrt{x} + y - 1, b = \sqrt{x} + y - 1$, and we still get signatures which are polynomial functions of x and y . This transformation further lowers the degree. In fact from now on we operate in the xy plane, forming our boxes there. We also note that on the xy plane we only need to consider $x > 0$. (Note that $x = 0$ corresponds to the line $a = b$ and is excluded in $\mathbb{R}^2 - E$.)

It turns out that we can find seven gadgets (one with $m = 2$, one with $m = 4$ and five with $m = 6$) such that the combined failure set can be proved by symbolic computation to be the union of: (1) a small region bounded by a small box (2) a small number of curves defined by polynomial equations (each with several branches), and (3) a small number of isolated points. It is somewhat awkward to bound the curves in part (2) individually. However we can prove by symbolic computation that it is contained in the box $0 < x < 0.14$ and $0 \leq y \leq 1$.

Let B be the union of two boxes from part (1) and (2). It can be proved by symbolic computation that both f_1 and f_2 are well defined within the box for part (1) and for any point of part (2) belonging to the curves. Every point (x, y) not in B and not in part (3) has its corresponding $[a, 1, b]$ already proved #P-hard. We use (8) and (9) (and in one case, a third rational transformation corresponding to another \mathcal{F} -gate) to prove that for all points in part (3), after a finite number of transformations they all fall outside of the union of three parts.

For points in B , we apply the rational transformations (8) and (9), again in combination and in iteration. Each iteration is as follows. Starting with B , in one iteration we define $\tilde{B} = \{(x, y) \in B \mid f_1(x, y) \in B \text{ and } f_2(x, y) \in B\}$. \tilde{B} is a smaller subset of B . We then bound \tilde{B} by a slightly larger new “box” B' . B' is still smaller than B , and this re-bounding is necessary since our symbolic computation can only handle polynomials of degrees with an absolute upper bound. Then we replace B with B' , and iterate.

After several iterations the “box” becomes a very thin strip, but it does not vanish. Our final knock is to realize that the “box” after several iterations has become so small, that we can in fact apply one of the seven gadgets to eliminate it completely in one step. This concludes our description of the proof of (7).

References

1. Bulatov, A.A., Dalmau, V.: Towards a dichotomy theorem for the counting constraint satisfaction problem. *Inf. Comput.* 205(5), 651–678 (2007)
2. Bulatov, A.A., Grohe, M.: The complexity of partition functions. *Theor. Comput. Sci.* 348(2-3), 148–186 (2005)
3. Cai, J.-Y., Lu, P.: Holographic Algorithms: From Art to Science. In: *Proceedings of STOC 2007*, pp. 401–410 (2007)
4. Cai, J.-Y., Lu, P., Xia, M.: Holographic Algorithms by Fibonacci Gates and Holographic Reductions for Hardness. In: *FOCS 2008*, pp. 644–653 (2008)
5. Cai, J.-Y., Lu, P., Xia, M.: Holant Problems and Counting CSP. In: *STOC 2009* (to appear, 2009)
6. Collins, G.: Quantifier Elimination for Real Closed Fields by Cylindric Algebraic Decomposition. In: Brakhage, H. (ed.) *GI-Fachtagung 1975*. LNCS, vol. 33, pp. 134–183. Springer, Heidelberg (1975)
7. Creignou, N., Hermann, M.: Complexity of Generalized Satisfiability Counting Problems. *Inf. Comput.* 125(1), 1–12 (1996)
8. Creignou, N., Khanna, S., Sudan, M.: Complexity classifications of boolean constraint satisfaction problems. *SIAM Monographs on Discrete Mathematics and Applications* (2001)
9. Dyer, M.E., Goldberg, L.A., Jerrum, M.: The Complexity of Weighted Boolean #CSP CoRR abs/0704.3683 (2007)
10. Dyer, M.E., Goldberg, L.A., Paterson, M.: On counting homomorphisms to directed acyclic graphs. *J. ACM* 54(6) (2007)
11. Dyer, M.E., Greenhill, C.S.: The complexity of counting graph homomorphisms. *Random Struct. Algorithms* 17(3-4), 260–289 (2000)
12. Goldberg, L.A., Grohe, M., Jerrum, M., Thurley, M.: A complexity dichotomy for partition functions with mixed signs. CoRR abs/0804.1932 (2008)
13. Hell, P., Nešetřil, J.: On the complexity of H-coloring. *J. Combin. Theory Ser. B* 48, 92–110 (1990)
14. Tarski, A.: *A Decision Method for Elementary Algebra and Geometry*, Univ. of Calif. (1951)
15. Vadhan, S.P.: The Complexity of Counting in Sparse, Regular, and Planar Graphs. *SIAM J. Comput.* 31(2), 398–427 (2001)
16. Valiant, L.G.: Holographic Algorithms (Extended Abstract). In: *Proc. 45th IEEE Symposium on Foundations of Computer Science*, pp. 306–315 (2004)

Block-Graph Width*

Maw-Shang Chang¹, Ling-Ju Hung¹, Ton Kloks, and Sheng-Lung Peng²

¹ Department of Computer Science and Information Engineering
National Chung Cheng University
Ming-Shiun, Chia-Yi 621, Taiwan
{mschang,hunglc}@cs.ccu.edu.tw

² Department of Computer Science and Information Engineering
National Dong Hwa University
Shoufeng, Hualien 97401, Taiwan
slpeng@mail.ndhu.edu.tw

Abstract. The \mathcal{G} -width of a class of graphs \mathcal{G} is defined as follows. A graph G has \mathcal{G} -width k if there are k independent sets N_1, \dots, N_k in G such that G can be embedded into a graph $H \in \mathcal{G}$ such that for every edge e in H which is not an edge in G , there exists an i such that both endpoints of e are in N_i . For the class \mathfrak{B} of block graphs we show that \mathfrak{B} -width is NP-complete and we present fixed-parameter algorithms.

1 Introduction

For two sets A and B we write $A + B$ and $A - B$ instead of $A \cup B$ and $A \setminus B$. We write $A \subseteq B$ if A is a subset of B with possible equality and we write $A \subset B$ if A is a subset of B and $A \neq B$. For a set A and an element x we write $A + x$ instead of $A + \{x\}$ and $A - x$ instead of $A - \{x\}$.

We denote edges of an undirected simple graph $G = (V, E)$ as (x, y) and we call x and y the endvertices of the edge. For a vertex x we write $N(x)$ for its set of neighbors and for a subset $W \subseteq V$ we write $N(W) = \bigcup_{x \in W} N(x) - W$. We write $N[x] = N(x) + x$ for the *closed* neighborhood of x and for a subset W we write $N[W] = N(W) + W$. A vertex x is *universal* if $N[x] = V$ and it is *isolated* if $N(x) = \emptyset$. Usually we use $n = |V|$ to denote the number of vertices of G and $m = |E|$ to denote the number of edges.

For a graph $G = (V, E)$ and a subset $S \subseteq V$ of vertices we write $G[S]$ for the subgraph *induced by* S , that is, the graph with S as its set of vertices and with those edges of E that have both endvertices in S . For a subset $W \subseteq V$ we write $G - W$ for the graph $G[V - W]$. A *hole* in G is an induced subgraph isomorphic to a cycle of length at least 4. For a vertex x we write $G - x$ rather than $G[V - x]$.

Definition 1. *Let \mathcal{G} be a class of graphs which contains all cliques. The \mathcal{G} -width of a graph G is the minimum number k of independent sets N_1, \dots, N_k in G such that there exists an embedding $H \in \mathcal{G}$ of G such that for every edge $e = (x, y)$ in H which is not an edge of G there exists an i with $x, y \in N_i$.*

* This research is supported by the National Science Council of Taiwan under grant NSC97-2221-E-194-055.

The collection of independent sets \mathbb{N}_i , $i = 1, \dots, k$ for which there exists an embedding H as stipulated in Definition 1 is called a *witness* of H .

In this paper we investigate the width-parameter for the class \mathfrak{B} of block graphs, henceforth called the *block-graph width*, or \mathfrak{B} -width. If a graph G has \mathfrak{B} -width k then we call G also a k -probe block graph. We refer to the *partitioned case* of the problem when a collection of independent sets \mathbb{N}_i , $i = 1, \dots, k$ is a part of the input. For historical reasons we call the set of vertices $\mathbb{P} = V - \bigcup_{i=1}^k \mathbb{N}_i$ the set of *probes* and the vertices of $\bigcup_{i=1}^k \mathbb{N}_i$ the set of *nonprobes*.

Block graphs were introduced in [7]. They are special ptolemaic graphs, *i.e.*, gem-free chordal graphs, in which every maximal 2-connected subgraph is a clique. We adapt the definition as it is presented in [1, Definition 10.2.3], to allow for disconnected graphs.

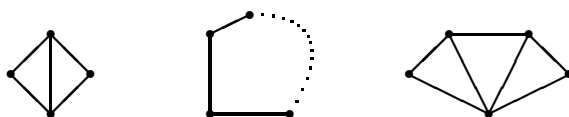


Fig. 1. A diamond, a hole, and a gem. A graph has rankwidth at most 1 if and only if it has no vertex minor isomorphic to the gem [10].

Definition 2 ([1,7]). *A graph G is a block graph if every biconnected component, or ‘block,’ is complete. Equivalently, G is a block graphs if G is chordal and has no induced subgraph isomorphic to $K_4 - e$, that is, the diamond.*

2 \mathfrak{B} -Width Is Fixed-Parameter Tractable

In this section we show that for constant k , k -probe block graphs can be recognized in $O(n^3)$ time.

Definition 3 ([11]). *A rank-decomposition of a graph $G = (V, E)$ is a pair (T, τ) where T is a ternary tree and τ a bijection from the leaves of T to the vertices of G . Let e be an edge in T and consider the two sets A and B of leaves of the two subtrees of $T - e$. Let M_e be the submatrix of the adjacency matrix of G with rows indexed by the vertices of A and columns indexed by the vertices of B . The width of e is the rank over $\text{GF}(2)$ of M_e . The width of (T, τ) is the maximum width over all edges e in T and the rankwidth of G is the minimum width over all rank-decompositions of G .*

Lemma 1. *Block graphs have rankwidth at most one.*

Proof. Obviously, the class of graphs with rankwidth at most one is hereditary. This class is exactly the class of distance-hereditary graphs [2,11]. Every block graph is a ptolemaic graph, and the ptolemaic graphs are the distance-hereditary graphs that are chordal (see, *e.g.*, [1]). \square

Theorem 1. *k-probe block graphs have rankwidth at most 2^k .*

Proof. Consider a rank-decomposition (T, τ) with width 1 for an embedding H of G . Consider an edge e in T and assume that M_e is an all-1s-matrix. Each independent set N_i creates a 0-submatrix in M_e . If $k = 1$ this proves that the rankwidth of G is at most 2. In general, for $k \geq 0$, note that there are at most 2^k different neighborhoods from one leaf-set of $T - e$ into the other. *A fortiori*, the rank of M_e is at most 2^k . \square

Lemma 2. *A graph is a block graph if and only if every connected induced subgraph is a clique or else has a cutvertex.*

Proof. Let $G = (V, E)$ be a block graph and let $G[W]$ be a subgraph induced by $W \subseteq V$. Assume $G[W]$ is connected. Then $G[W]$ is a block graph, since by Definition 2, ‘to be a block graph’ is a hereditary property. If $G[W]$ is not complete then it has a cutvertex by Definition 2.

Assume every connected induced subgraph of a graph G has a cutvertex or is a clique. Then every biconnected component of G is a clique. Thus G is a block graph. \square

Lemma 3. *Suppose that G is a k-probe block-graph and suppose that H is a minimal k-probe block-graph embedding of G . Then a vertex is a cutvertex of H if and only if it is a cutvertex of G .*

Proof. Suppose G has ℓ biconnected components C_1, C_2, \dots, C_ℓ . Let H^* be the graph $(V, \cup_{i=1}^{\ell} E(H[C_i]))$. Note that a vertex is a cutvertex of H^* if and only if it is a cutvertex of G . We prove the lemma by showing that H^* is a k-probe block-graph embedding of G .

Suppose that H^* is not a block graph. Then H^* has a forbidden induced subgraph which is a diamond or a hole. Let F be a set of vertices that induces a forbidden subgraph. Thus $H^*[F]$ is either a diamond or a hole. All forbidden induced subgraphs are biconnected thus F is contained in some C_i . But every $H^*[C_i]$ is a block. This proves the lemma. \square

Corollary 1. *A graph G is a k-probe block graph if and only if there exist independent sets $N_i, i = 1, \dots, k$ such that every connected induced subgraph of G either*

- (1) *has a cutvertex, or*
- (2) *for every pair of vertices x and y either $(x, y) \in E$, or there exists $i \in \{1, \dots, k\}$ such that $\{x, y\} \subseteq N_i$.*

Theorem 2. *For each $k \geq 0$ there exists an $O(n^3)$ algorithm which checks whether a graph G with n vertices is a k-probe block graph. Thus \mathfrak{B} -width is in FPT.*

Proof. k-Probe block graphs have bounded rankwidth. Every graph problem specified by a formula of monadic second-order logic (without edge set quantifications) can be solved in $O(n^3)$ time for graphs of bounded rankwidth [3,6,11]. By Corollary 1, the recognition of k-probe block graphs is such a problem. \square

3 Partitioned k-Probe Block Graphs

Obviously, the result of the previous section holds as well when the collection of independent sets $\mathbb{N}_1, \dots, \mathbb{N}_k$ is a part of the input. Thus for each k there is an $O(n^3)$ algorithm that checks whether a graph G with k independent sets \mathbb{N}_i can be embedded into a block graph with these independent sets as a witness. However, there are a few drawbacks to this solution. First of all, Theorem 2 only shows the *existence* of an $O(n^3)$ recognition algorithm. *A priori*, it is unclear how to obtain the algorithm. Furthermore, the constants involved in the algorithm make the solution impractical. Already there is an exponential blow-up when one moves from \mathfrak{B} -width to rankwidth.

In this section we show that there exists a polynomial-time algorithm for the recognition of partitioned k -probe block graphs.

By Corollary 1 we have:

Proposition 1. *A graph G with independent sets \mathbb{N}_i , $i = 1, \dots, k$, is a partitioned k -probe block graph if and only if every biconnected component of G , with the induced independent sets, is a partitioned k -probe complete graph.*

Theorem 3. *For every k there exists an $O(n^2)$ -time algorithm to check whether a graph G , equipped with a collection of k independent sets is a partitioned k -probe block graph.*

Proof. By Proposition 1 it is sufficient to describe an algorithm which checks if a partitioned graph can be embedded into a complete graph. To check if a partitioned k -probe graph is a k -probe complete graph, we can simply add all the possible edges and check if it is a complete graph.

A biconnected-component decomposition of a graph can be found in linear time. It takes $O(n^2)$ time to add all possible edges in each biconnected component and to check if the embedding of each biconnected component is a clique. Hence a partitioned k -probe block graph can be recognized in $O(n^2)$ time. \square

Remark 1. Note that the algorithm described in Theorem 3 is *fully* polynomial. As a corollary we obtain that also the *sandwich* problem for complete graphs can be solved in $O(n^2)$ time. This result was independently obtained in [4].

Remark 2. It is interesting that we can decompose a partitioned k -probe complete graph into a *k-geometric graph* $T(k)$. The vertices of $T(k)$ are the 2^k 0, 1-vectors of length k . Two of these vertices a and b are adjacent in $T(k)$ if the vectors have no 1 in any common entry. A $T(k)$ -*decomposition* of G maps a vertex x of G to the vertex $L(x)$ of $T(k)$. Thus the probes are mapped to the all-0 vector. We call the collection of vertices that are mapped to the same vertex in $T(k)$, the *bags* of this $T(k)$ -decomposition. Note that, if vertices are in bags that are adjacent in $T(k)$, then no edge can be added between them since they are not in any common \mathbb{N}_i . This gives the proof that the unpartitioned case of complete-width is in FPT. The bags are equivalent classes. Two vertices are in the same bag if they have the same neighborhood in G .

4 A Fixed-Parameter Algorithm to Compute \mathfrak{B} -Width

In this section we present an explicit algorithm which checks for each k whether the \mathfrak{B} -width of a graph G is at most k .

Definition 4. A label of a vertex x is a $0, 1$ -vector of length k . Let $\{\mathbb{N}_i \mid i = 1, \dots, k\}$ be a witness of an embedding. A label $L(x)$ is an indicator of this witness if the i^{th} component of $L(x)$ is 1 if and only if $x \in \mathbb{N}_i$.

We use a decomposition tree T on the biconnected components of G defined as follows.

Definition 5. A biconnected-component decomposition for a graph G is a pair (T, \mathcal{S}) , where T is a tree and $\mathcal{S} = \{S_i \mid i \in V(T)\}$ a collection of bags in one-to-one correspondence with the vertices of T , such that

- (i) \mathcal{S} is the collection of biconnected components of G , and
- (ii) if i and j are two adjacent nodes in T then $S_i \cap S_j$ is a cutvertex in G .

A biconnected-component decomposition (T, \mathcal{S}) of a graph G can be computed in linear time [13].

Definition 6. Let i be a node in T . An input set is a set $L(x)$ of labels for each vertex $x \in S_i$, such that:

1. If x is not a cutvertex of G , or if x is the unique cutvertex which separates S_i from its parent in T , then $L(x)$ is the collection of all $0, 1$ -vectors of length k .
2. If x is a cutvertex which separates S_i from one of its children S_j , then $L(x)$ is the set of indicators of x for witnesses of embeddings of the graph G_j induced by the bags in the subtree rooted at j .

Definition 7. Let S_i be a biconnected component of G . Define the following equivalence relations on the vertices of $B_i = G[S_i]$.

- (a) two nonadjacent vertices x and y are equivalent if $N_{B_i}(x) = N_{B_i}(y)$, and
- (b) two adjacent, universal vertices x and y in B_i are equivalent.

The representative R_i of S_i is the graph defined on the equivalence classes, where two classes C_1 and C_2 are adjacent in R_i if there are adjacent vertices $x \in C_1$ and $y \in C_2$.

Lemma 4. If R_i has more than 2^k vertices, then the \mathfrak{B} -width of G is more than k .

Proof. If the \mathfrak{B} -width of G is at most k then the biconnected component S_i is embedded as a clique. For every indicator L the set $C_L = \{x \mid L(x) = L\}$ is a set of equivalent vertices. Since there are at most 2^k of these indicators, there can be at most 2^k vertices in R_i since each vertex is a union of some of the sets C_L . \square

Definition 8. Let R_i be a representative with vertex set $V(R_i)$. For each vertex $C \in V(R_i)$ let $Q(C)$ be a subset of labels. The set $\mathcal{Q} = \{Q(C) \mid C \in V(R_i)\}$ is a feasible extension if

- i. If C is an independent set then every pair of labels $Q_1, Q_2 \in Q(C)$ has a 1 in a common entry.
- ii. If C is a clique then every pair of labels $Q_1, Q_2 \in Q(C)$ has no 1 in any common entry.
- iii. If C_1 and C_2 are adjacent in R_i then every pair $Q_1 \in Q(C_1)$ and $Q_2 \in Q(C_2)$ have no 1 in any common entry.
- iv. If C_1 and C_2 are not adjacent in R_i then every pair $Q_1 \in Q(C_1)$ and $Q_2 \in Q(C_2)$ have a 1 in some common entry.

Consider the input set of a node i in the biconnected-component decomposition. Consider two children p and q of i incident with the same cutvertex $x \in S_i$. A *reduction* of the input set replaces both $L(p)$ and $L(q)$ by $L(p) \cap L(q)$. Thus a reduced input set has only one set of labels for each cutvertex in S_i .

Lemma 5. Let i be a node in the biconnected-component decomposition and let G_i be the graph induced by the bags in the subtree of T rooted at node i . For a vertex $x \in S_i$ let C_x be the equivalence class of R_i that contains x . Then G_i has \mathfrak{B} -width at most k if and only if there exists a feasible extension \mathcal{Q} of R_i such that each vertex x has a label $L(x)$ in the reduced input set which is an element of $Q(C_x)$.

Proof. Assume there exists a witness $\{\mathbb{N}_i \mid i = 1, \dots, k\}$ for the graph G_i . Consider the labels $L(x)$ for $x \in S_i$ of this witness. Define a labeling \mathcal{Q} on the equivalence classes C of R_i as follows: For each equivalence class C let $Q(C) = \{L(x) \mid x \in C\}$. We claim that \mathcal{Q} is a feasible extension of R_i . Indeed, the conditions of Definition 8 are easy to check.

To prove the converse, consider a feasible extension \mathcal{Q} of R_i such that each vertex x has a label $L(x)$ in the reduced input which is an element of $Q(C_x)$. Choose such a label for each $x \in S_i$. By definition of a (reduced) input set, there exist embeddings for each graph G_j induced by bags in the subtree rooted at cutvertex $s \in S_j \cap S_i$ with label $L(s)$. By Definition 8 of a feasible extension, the independent sets extend to S_i since any two vertices with labels that have a 1 in a common entry are nonadjacent. \square

Theorem 4. For every k there exists an $O(n^3)$ -time algorithm which check if the \mathfrak{B} -width of a graph G is at most k .

Proof. The algorithm computes a witness for an embedding (if it exists) by dynamic programming on the biconnected-component decomposition tree. By Lemma 5 it is sufficient to prove that the set of feasible labelings of each representative R_i can be computed in constant time. There are at most 2^k possible labels. Thus there are $O(2^{2^k})$ subsets of labels that may be assigned to the different equivalence classes of R_i . By Lemma 4, R_i has $O(2^k)$ vertices. Thus there are $O(2^{2^{2^k}})$ feasible extensions \mathcal{Q} for a representative R_i . For each, the algorithm checks if the input labeling satisfies Lemma 5. \square

5 \mathfrak{B} -Width Is NP-Complete

Let \mathfrak{T} be the collection of complete graphs (cliques). We first show that \mathfrak{T} -width is NP-complete.

Theorem 5. *\mathfrak{T} -Width is NP-complete.*

Proof. Let G be a partitioned k -probe complete graph. Thus every non-edge of G has its endvertices in one of the independent sets N_i . Thus the collection N_i forms a clique-cover of the edges of \bar{G} . This shows that a graph G has \mathfrak{T} -width at most k if and only if the edges of \bar{G} can be covered with k cliques. The problem to cover the edges of a graph by a minimum number of cliques is NP-complete [9]. \square

Theorem 6. *\mathfrak{B} -Width is NP-complete.*

Proof. Assume that there is a polynomial-time algorithm to compute \mathfrak{B} -width. We show that we can use that algorithm to compute \mathfrak{T} -width. Let G be a graph for which we wish to compute \mathfrak{T} -width. Add two universal vertices to G , that is, two adjacent vertices ω_1 and ω_2 both adjacent to all other vertices of G . Let G' be this graph. Note that the only way to embed G' into a block graph is to make it a clique, since G' is biconnected. The minimum number of independent sets needed for this embedding is exactly the \mathfrak{T} -width of G . This proves the NP-completeness of \mathfrak{B} -width. \square

6 A Finite Obstruction Set

Let $\mathfrak{T}(k)$ be the class of graphs with \mathfrak{T} -width at most k . The following theorem shows that $\mathfrak{T}(k)$ can be characterized by a finite set of forbidden induced subgraphs. A similar result was obtained in [12]. Basically, graphs with complete width at most k , and similar structures, are well-quasi-ordered by an induced-subgraph relation. The result can be derived from Higman's classic work on well-quasi-orderings [5].

Theorem 7. *There exists a set $F(k)$ of graph with at most $2^{k+1}+1$ vertices such that $G \in \mathfrak{T}(k)$ if and only if G has no element of $F(k)$ as an induced subgraph.*

Proof. Let F be a graph with \mathfrak{T} -width more than k and assume that for any vertex x of F the graph $F - x$ has \mathfrak{T} -width at most k . Let x be a vertex of F and consider an embedding of $F - x$. Then the vertices of $F - x$ can be partitioned into at most 2^k modules, of which one is a clique and the others are independent sets. Consider such a module M and consider $N(x) \cap M$. If M has more than 2 vertices, then two of them must be a twin in F . Since we assume that F is a minimal forbidden induced subgraph, it cannot contain such a twin. It follows that each module in $F - x$ has at most two vertices. Thus F has at most $2^{k+1} + 1$ vertices. \square

We have not found a proof that the class $\mathfrak{B}(k)$ of k -probe block graphs can be characterized by forbidden induced subgraphs.

7 Concluding Remarks

The recognition problem of probe interval graphs was introduced by Zhang *et al.* [8,14]. This problem stems from the physical mapping of chromosomal DNA of humans and other species. Since then probe graphs of many other graph classes have been investigated by various authors. In this paper we generalized the problem to the width-parameter of graph classes. So far we restricted to classes of graphs that have bounded rankwidth. For classes such as threshold graphs and cographs we were able to show that the width parameter is fixed-parameter tractable. One class for which this is still open is the class of distance-hereditary graphs.

Acknowledgement

Ton Kloks is currently a guest of the Department of Computer Science and Information Engineering of National Chung Cheng University. He gratefully acknowledges the funding for this research by the National Science Council of Taiwan.

References

1. Brandstädt, A., Le, V.B., Spinrad, J.P.: Graph classes: A survey. In: SIAM Monographs on Discrete Mathematics and Applications, Philadelphia (1999)
2. Chang, M.-S., Hsieh, S.-Y., Chen, G.-H.: Dynamic programming on distance-hereditary graphs. In: Leong, H.-V., Jain, S., Imai, H. (eds.) ISAAC 1997. LNCS, vol. 1350, pp. 344–353. Springer, Heidelberg (1997)
3. Courcelle, B., Oum, S.: Vertex minors, monadic second-order logic, and a conjecture by Seese. *Journal of Combinatorial Theory, Series B* 97, 91–126 (2007)
4. Golumbic, M.C., Kaplan, H., Shamir, R.: Graph sandwich problems. *Journal of Algorithms* 19, 449–473 (1995)
5. Higman, G.: Ordering by divisibility in abstract algebras. *Proceedings of the London Mathematical Society* 2, 326–336 (1952)
6. Hliněný, P., Oum, S., Seese, D., Gottlob, G.: Width parameters beyond tree-width and their applications. *Computer Journal* 51, 326–362 (2008)
7. Kay, D.C., Chartrand, G.: A characterization of certain ptolemaic graphs. *Canadian Journal of Mathematics* 17, 342–346 (1965)
8. McMorris, F.R., Wang, C., Zhang, P.: On probe interval graphs. *Discrete Applied Mathematics* 88, 315–324 (1998)
9. Orlin, J.: Contentment in graph theory: covering graphs with cliques. *Indagationes Mathematicae* 39, 406–424 (1977)
10. Oum, S.: Rank-width and vertex-minors. *Journal of Combinatorial Theory, Series B* 95, 79–100 (2005)
11. Oum, S.: Graphs of bounded rank-width. PhD Thesis. Princeton University, Princeton (2005)
12. Petkovšek, M.: Letter graphs and well-quasi-order by induced subgraphs. *Discrete Mathematics* 244, 375–388 (2002)
13. Tarjan, R.: Depth-first search and linear graph algorithms. *SIAM Journal of Computing* 1, 146–160 (1972)
14. Zhang, P., Schon, E.A., Fischer, S.G., Cayanis, E., Weiss, J., Kistler, S., Bourne, P.E.: An algorithm based on graph theory for the assembly of contigs in physical mapping of DNA. *CABIOS* 10, 309–317 (1994)

Minimum Vertex Ranking Spanning Tree Problem on Permutation Graphs

Ruei-Yuan Chang, Guanling Lee, and Sheng-Lung Peng*

Department of Computer Science and Information Engineering
National Dong Hwa University, Hualien 974, Taiwan
slpeng@mail.ndhu.edu.tw

Abstract. The minimum vertex ranking spanning tree problem on graph G is to find a spanning tree T of G such that the minimum vertex ranking of T is minimum among all possible spanning trees of G . In this paper, we propose a linear-time algorithm for this problem on permutation graphs. It improves a previous result that runs in $O(n^3)$ time where n is the number of vertices in the input graph.

1 Introduction

Let $G = (V, E)$ be a finite, simple, and undirected graph. Let $n = |V|$ and $m = |E|$. Let $N(v) = \{u \mid (u, v) \in E\}$. In this paper, we denote $|G| = |V(G)|$ for convenient. A shortest path P between $u, v \in V(G)$ is a path with $|P|$ is minimum among all possible paths between u, v . The *diameter* of G is the longest shortest path among all possible shortest paths in G .

A *vertex ranking* of a graph G is a function $\gamma : V \rightarrow \mathbb{N}$ such that every u, v path in G with $\gamma(u) = \gamma(v)$ contains at least one vertex w in the path with $\gamma(w) > \gamma(u)$. In a vertex ranking γ , $\gamma(v)$ is called the *rank* of v . A vertex ranking γ is *minimum* if the largest rank used in γ is the minimum among all possible vertex rankings of G . We denote the largest rank used in the minimum vertex ranking of G by $rank(G)$. The *minimum vertex ranking problem* on G is to find a minimum vertex ranking of G . This problem has applications to, *e.g.*, communication network design, planning efficient assembly of products in manufacturing systems [4,5,6,12], and VLSI layout design [3,7].

The *minimum vertex ranking spanning tree* (MVRST for short) problem on G is to find a spanning tree of G such that the vertex ranking of the spanning tree is minimum among all possible spanning trees of G [9]. This problem has applications in scheduling the parallel assembly of a multi-part product from its components and in relational database. Miyata *et al.* proved that the decision version of MVRST problem is NP-complete on general graphs [8]. However, it can be solved in $O(n^3)$, $O(n^3)$, $O(n^5)$, and $O(n^5 \log^4 n)$ time on interval graphs [9], permutation graphs [10], outerplanar graphs [11], and series-parallel graphs [1], respectively. Recently, Chang *et al.* showed that the problem on interval graphs, split graphs, and cographs can be solved in linear time [2].

* Corresponding author.

In this paper, we propose a linear-time algorithm for the MVRST problem on permutation graphs. Our result improves an $O(n^3)$ -time result of [10].

2 Permutation Graphs

Let π be a permutation of $(1, \dots, n)$. The *matching diagram* of π is obtained as follows. Write the integers $(1, \dots, n)$, horizontally from left to right. Underneath, write the integers (π_1, \dots, π_n) , also horizontally from left to right. Draw n straight line segments connecting the two 1's, the two 2's, and so on. A graph is a *permutation graph* if it is isomorphic to the intersection graph of the line segments of a matching diagram.

Masuyama and Nakayama proposed an $O(n^3)$ algorithm for solving the MVRST problem on permutation graphs [10]. The idea proposed in [10] is as follows. At the beginning, they found a path P which is the shortest among the four shortest paths from v_1 to v_n , v_1 to v_{π_n} , v_{π_1} to v_n , and v_{π_1} to v_{π_n} . Then they proved that each vertex in $V \setminus P$ is adjacent to some vertex in each path of the four shortest paths [10]. Next, they partition the vertices in $V \setminus P$ into three subsets. Finally, they construct a spanning tree T with minimum vertex ranking by linking the vertices in $V \setminus P$ to P directly or via another vertices in $V \setminus P$. Actually, they use the dynamic programming technique to determine how to link the vertices in $V \setminus P$. However, it takes much time in many useless cases. To reduce the time complexity, we try to construct the MVRST directly without considering those useless cases.

In this paper, we use diameter as P . Note that it is not hard to check one of the four shortest paths mentioned above must be a diameter. It is easy to see that the rank of any spanning tree of G must be greater than or equal to the rank of the diameter. In [10], they proved the lower bound of a path and the upper bound of the MVRST in permutation graphs. Therefore, we can easily obtain the following lemma.

Lemma 1. *For a permutation graph G , the MVRST T satisfies the following inequality: $\text{rank}(P) \leq \text{rank}(T) \leq \text{rank}(P) + 1$ where P is a diameter of G .*

Our idea on permutation graphs is similar to the one used in [2] for interval graphs. The difference is that permutation graphs contain C_4 . It makes us need to consider more cases during linking the vertices in $V \setminus P$ to P . As mentioned in [2], the minimum vertex ranking of a path is not unique. Thus it is hard to determine how to link the vertices in $V \setminus P$ to P . To construct the MVRST of G , we need more powerful property. If the minimum vertex ranking of a path could be unique, then we can easily determine how to link the vertices in $V \setminus P$. Fortunately, if a path satisfies some properties, then there is a unique minimum vertex ranking for this path. The following lemmas mentioned in [2] show the properties.

Lemma 2 ([2]). *Let P be a path with $|P| = 2^r - 1$ where $r \in \mathbb{N}$. Then, the minimum vertex ranking of P is unique and $\text{rank}(P) = r$.*

Lemma 3 ([2]). *Let P_1 and P_2 be two paths such that $2^{\lfloor \log_2 |P_1| \rfloor} - 1 < |P_1| \leq 2^{\lfloor \log_2 |P_1| \rfloor + 1} - 1 = |P_2|$. Then, $\text{rank}(P_1) = \text{rank}(P_2)$. Moreover, if P_3 is a path with $|P_3| > |P_2|$, then $\text{rank}(P_3) > \text{rank}(P_2)$.*

Our idea is that we link the vertices in $V \setminus P$ to P by our connecting ways and we map (and extend) P to the corresponding path P^* whose vertex ranking is unique and $\text{rank}(P) = \text{rank}(P^*)$. Then we determine the case that when will the rank of MVRST be $\text{rank}(P)$ or $\text{rank}(P) + 1$ by checking whether we can map P to P^* or not. If we can map P to P^* , then we obtain an MVRST with rank equal to $\text{rank}(P)$. If not, then we show that the rank of the MVRST of G is at least $\text{rank}(P) + 1$. By Lemma 1, it is optimal. More specifically, if $|P| < 2^r - 1$ for some $r \in \mathbb{N}$, we try to link the vertices in $V \setminus P$ to P as many as possible before using total ranking numbers of P^* such that P cannot map to P^* with $|P^*| = 2^{\lfloor \log_2 |P| \rfloor + 1} - 1$ by Lemma 3. If we can link all the vertices in $V \setminus P$ before using total ranking numbers of P^* such that P can map to P^* , we obtain an MVRST whose rank is equal to $\text{rank}(P)$. Otherwise, we will do the following operation. Let $P = [u_1, u_2, \dots, u_d]$ and γ' be an optimal ranking on the subpath $P \setminus \{u_1, u_d\}$. The *levelup* operation on P is a ranking γ on P by letting $\gamma(u_1) = \gamma(u_d) = 1$ and $\gamma(u_i) = \gamma'(u_i) + 1$ for $1 < i < d$. If we do this operation on P , we can directly link every vertex v in $V \setminus P$ to a vertex in $\{u_2, \dots, u_{d-1}\}$ by letting $\gamma(v) = 1$. It lets us obtain a vertex ranking spanning tree of G . In this paper, we call the number $2^{\lfloor \log_2 |P| \rfloor + 1} - 1$ the *corresponding minimum vertex ranking bound* (CMVRB for short) of the path P . We will use CMVRB to compute the mapping from P to P^* . The following lemma mentioned in [2] shows that we can do *levelup* on P for some P .

Lemma 4 ([2]). *Let P be a diameter of permutation graph G . If $|P|$ is either $2^{\lfloor \log_2 |P| \rfloor}$ or $2^{\lfloor \log_2 |P| \rfloor} + 1$, then the rank of the MVRST of G is $\lfloor \log_2 |P| \rfloor + 1$.*

Let P be the diameter of the permutation graph $G = (V, E)$ mentioned above. In [10], vertices in $V \setminus P$ can be partitioned into three subsets, namely, V'_1 , V'_2 , and V'_3 as follows.

- V'_1 contains the vertices that are exactly adjacent to one vertex of P .
- V'_2 contains the vertices which are adjacent to two or three consecutive vertices of P .
- V'_3 contains the vertices that are adjacent to two vertices of P and there is a vertex between these two vertices in P .

The rank of the vertex in P' which is directly connected by the vertex v in V'_1 is ranked at least 2 and $\gamma(v) = 1$. The rank of the vertex in P' which is connected to a vertex v in V'_1 via the vertex $u \in V'_2$ or $u \in V'_3$ is ranked more than 2 and $\gamma(v) = 1, \gamma(u) = 2$.

For our algorithm, we partition V'_1 into three subsets, namely, V'_{1-1}, V'_{1-3} , and V'_{1-2} as follows.

- V_1 contains the vertices in V'_1 which have no neighbor in V'_1 .
- V'_{1-1} contains the vertices in V'_1 whose connected vertex in P is also connected by V_1 .

- V'_{1-3} contains the vertices in V'_1 whose connected vertex u in P is not connected by V_1 and can connect to both of $\{u_1, u_2\} = N(u) \cap P$ via some vertex w in V'_2 or V'_3 .
- V'_{1-2} contains the vertices in V'_1 whose connected vertex u in P is not connected by V_1 and can connect to only one of $N(u) \cap P$ via some vertex w in V'_2 or V'_3 .

Figure 1 gives an example of these three subsets.

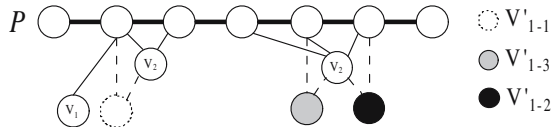


Fig. 1. $V_1, V'_{1-1}, V'_{1-3},$ and V'_{1-2}

Since P is the diameter of G and the vertices in V_2 (respectively, V'_{1-3}) is adjacent to two or three (respectively, three) consecutive vertices of P , we can easily obtain the following lemma. Thus we can process the vertices in $V_2 \cup V'_{1-3}$ after solving the other cases.

Lemma 5. *Let T be the MVRST of $G[V'_2 \cup V'_{1-3} \cup P]$. Then $rank(T) = rank(P)$.*

There are some vertices in $V \setminus P$ which have a unique connecting way. Consider the vertices in V_1 . They have only one choice to connect to P . So we connect vertices of V_1 directly. The proofs of the following two lemmas are easy, so we omit them.

Lemma 6. *Let $v \in V_1$ be adjacent to $u \in P$. Then there exists a minimum vertex ranking γ of $G[P \cup V_1]$ such that $\gamma(u) \geq 2$ and $\gamma(v) = 1$.*

Lemma 7. *Let T be the MVRST of $G[V'_{1-1} \cup V_1 \cup P]$. Then $rank(T) = rank(G[V_1 \cup P])$.*

As showed in Lemmas 6 and 7 we can connect V_1 and V'_{1-1} at the same time and the vertices in P which are connected by V_1 and V'_{1-1} must be ranked more than 1 in a minimum vertex ranking. Thus, all we have to do is to connect vertices of V'_{1-2} and V_3 to P such that the rank is as unchanged as possible. Since the vertices in P which are connected by V_1 and V'_{1-1} must be ranked more than 1 in some minimum vertex ranking, we mark these vertices *black* to denote these vertices to be ranked more than 1. By the following lemma proved in [2], we also mark the second and the second to last vertices of P black since the first and the last vertices of P can be treated as the vertices in V_1 or V'_{1-1} .

Lemma 8 ([2]). *Let P be a path. Then there exists a minimum vertex ranking of P such that the first and last vertices in P are both ranked 1.*

Consider the minimum vertex ranking for the path with unique minimum vertex ranking. The number of vertices of the path must be odd and the rank of every vertex in odd (respectively, even) position must be 1 (respectively, more than 1). For an arbitrary minimum vertex ranking of P , if two adjacent vertices or an endvertex is ranked more than 1, there can be an ignored hidden vertex which is ranked as 1. This is because when we are going to map P to P^* with $|P^*| = CMVRB$, any two consecutive vertices in P^* are not both ranked more than 1. So we have to ignore one vertex in odd position of P^* for mapping. In this paper, we call such an ignored hidden vertex a *division*. Clearly, if two adjacent vertices in P are marked black, then there exists a division between these two vertices. We denote the division set $D_s = \{u \mid u \text{ is a division between two consecutive black vertices}\}$. Also, the following lemma proposed in [2] implies that if the number of vertices of a subpath between two black vertices in P is even, then there is at least one division in this subpath.

Lemma 9 ([2]). *If P is a path with $|P|$ being even, then there must be at least one division in P .*

These black vertices partition P into P_1, P_2, \dots, P_s with $|P_i| > 0$ for $1 \leq i \leq s$. For P^* , it is clear that all the odd vertices can be ranked as 1 and all the even vertices can be ranked with $k > 1$. Since all the black vertices in P must be ranked more than 1, we have to keep the black vertices to be located in the even position in P^* . Note that we also count position of each division. Hence we can extend the even path P_i to an odd path P' such that the black vertices can be in the even location of P' . We denote $\mathbb{P}^{even} = \{P_i \mid 1 \leq i \leq s \text{ and } |P_i| \text{ is even}\}$. Then we can count the number of divisions in even subpaths by $|\mathbb{P}^{even}|$. We have the following lemma according to Lemma 9.

Lemma 10. *There exists a division in each element of \mathbb{P}^{even} .*

Let *free vertex number* (FVN for short) be the number of vertices that can be added into P such that the number of vertices of the resulting path P^* is $CMVRB$. Then we can compute $FVN = CMVRB - |D_s| - |\mathbb{P}^{even}| - |P|$. For example as Figure 2, suppose that P contains ten vertices and the black vertices are in the location as shown in Figure 2. Then we have $FVN = (2^4 - 1) - 2 - 1 - 10 = 2$.

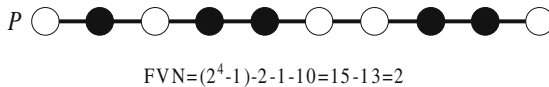


Fig. 2. Compute FVN

Now, we are going to connect the vertices in V'_{1-2} and V'_3 to P . There are three ways to connect $v \in V'_{1-2}$ to P . Let $u \in P \cap N(v)$. The first way is to connect v to u directly when $\gamma(u) > 1$. The second way is to connect v to a vertex w in $\{V_2 \cup V'_3\} \cap N(v)$ when $\gamma(u) = 1$ and there exists a vertex $u' \in N(w) \cap P$ with $\gamma(u') > 2$. These two ways would not decrease FVN since they do not change the minimum vertex ranking and the size of P . When both two ways cannot work, then we have the last way as follows. We choose one of the two vertices in $N(v) \cap P$ and change its ranking number to the next mapping ranking number, *i.e.* we map it to the next vertex of P^* . However, the last operation makes $|D_s|$ increase 1 and FVN decrease 1. If $FVN = 0$ and there are still some vertices in V'_{1-2} that can only connect to P by the last way, then it will increase the rank of P such that the remaining vertices can be added to the spanning tree. That is, we have to do *levelup* on P . Figures 3(a), 3(b), and 3(c) show the three connecting ways respectively.

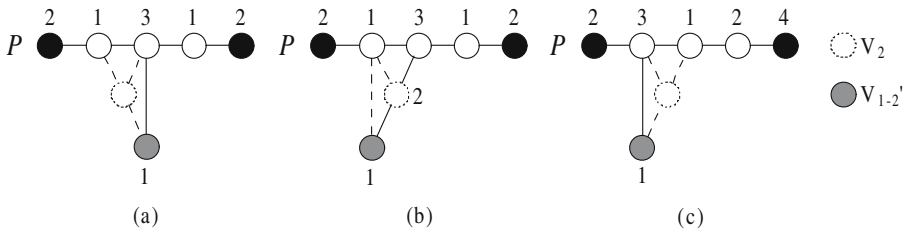


Fig. 3. The three connecting ways for vertices in V'_{1-2}

Now we consider the vertex $v \in V'_3$. Although v is adjacent to two vertices u and w in P , these two vertices are not consecutive. Thus we cannot make sure if any one of the ranks of u and w is greater than 2 or not. This is because the class of permutation graphs is not C_4 -free. To connect v , we do it by a similar way as the vertices in V'_{1-2} . The first way is choosing one of u and w whose rank is more than 1. If none of these two vertices is ranked more than 1, then we do the following way. The Second way is choosing a vertices in $N(v) \setminus P$ which can connect to a vertex in P whose rank is more than 2. Then we connect v via this vertex to P . If both two ways cannot work, then we do the last way. The last way is as follows. Since vertices in V'_{1-2} may connect to P via vertices in V'_3 , we could keep the structure. The last connecting way is that we choose one of u and w to give the rank more than 1, *i.e.*, we map it to the next vertex of P^* . However, the last operation makes the number of divisions increase 1 but FVN decrease 1. Figure 4 shows the three connecting ways.

Note that some vertices in V'_3 may be adjacent to some black vertices in P . Before starting to process the vertices in $V'_{1-2} \cup V'_3$, we mark these vertices in V'_3 “OK” since these vertices can do the first connecting way. For convenience, we let V_{ok} denote the vertex set in which every vertex is marked “OK”. Then we mark the vertices in P which can be connected to V'_{1-2} as *red* to denote that these vertices need to be considered especially. Also, we mark the left vertex in

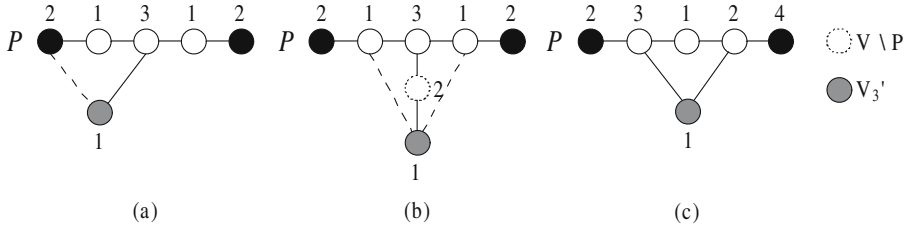


Fig. 4. The three connecting ways for vertices in V'_3

P which is adjacent to the vertex in $V'_3 \setminus V_{ok}$ *yellow*. If a vertex in P is adjacent to a vertex in V'_{1-2} and a vertex in $V'_3 \setminus V_{ok}$, then we mark this vertex *purple*. The following lemma shows that the connecting order of V'_{1-2} and $V'_3 \setminus V_{ok}$ will not affect the rank of the MVRST of G . Because of the limitation of the space of this paper, we omit the proof of this lemma.

Lemma 11. *Let P be a diameter of a permutation graph G with $v, v' \in V'_{1-2}$ and $w, w' \in V_3$. Then connecting v, v', w , or w' first leads to the same result.*

Hence by Lemma 11 we can start to scan the vertices in P from left to right to link vertices in V'_{1-2} and $V'_3 \setminus V_{ok}$. If the red vertex $u \in P$ which is adjacent to $v \in V'_{1-2}$ is going to be mapped to the odd position of P^* , then v can only be applied the second or third connecting way. Otherwise, v can be applied the first operation. Our Procedure `Process-Red-Vertices` is as follows.

Procedure. `Process-Red-Vertices`(u)

- for** $v \in N(u) \cap V'_{1-2}$ **do**
 - check if $\exists w \in N(v) \cap \{V'_2 \cup V'_3\}$ such that $\exists u' \in N(w) \cap P$ whose position is in $4x$ -th position in P ;
 - if** w exist **then**
 - mark v OK;
- if** $\forall v \in N(u) \cap V'_{1-2}$ is marked OK **then**
 - connect v to w and w to u' ;
 - $rank(v) = 1$ and $rank(w) = 2$;
- else**
 - do the third operation;
 - $FVN = FVN - 1$;
 - $j = j + 1$;
 - $division = division + 1$;
 - connect v to u and $rank(v) = 1$;

mark u white;

If the yellow vertex $u \in P$ which is adjacent to $v \in V'_3 \setminus V_{ok}$ is going to be mapped to the odd position of P^* , then we will add a division between

the next two vertices in P such that the other one vertex in $N(v) \cap P$ is mapped to the vertex in P^* with ranking number more than 1. Our Procedure **Process-Yellow-Vertices** is as follows.

```

Procedure. Process-Yellow-Vertices( $u$ )
  if  $u$  is yellow and its position  $j \bmod 2 \neq 0$  then
    if the right neighbor vertex  $u' \in P$  is red or purple then
      map  $u$  to  $j$ -th position in  $P^*$ ;
      connect  $N(u') \cap V'_{1-2}$  to  $u'$ ;
      if  $u'$  is in the  $4x$ -th position in  $P$  then
         $\lfloor$  connect  $N(u') \cap V'_3$ ;
      else
         $\lfloor$  mark  $N(u') \cap V'_3$  "OK";
      mark  $u$  and  $u'$  white and map  $u'$  to  $j + 1$ -th position in  $P^*$ ;
      map the right neighbor vertex of  $u'$  to  $j + 3$ -th position in  $P^*$ ;
       $division = division + 1$  and  $FVN = FVN - 1$ ;

```

We now consider the case of purple vertices. If the purple vertex $u \in P$ which is adjacent to $v \in V'_{1-2}$ and $v' \in V'_3 \setminus V_{ok}$, then we first check if for all $v \in V'_{1-2}$ can connect to a vertex in P which is mapped to a vertex in P^* whose ranking number is more than 2. If it does, then we process the purple vertex as a yellow vertex. Otherwise, we process it as a red vertex.

We keep the position j to compute if u is mapped to the even position of P^* . Note that the divisions are also computed in j . If u is mapped to the odd position in P^* , then we will do the operation mentioned above. That is, if u is marked red and needs to do the third operation, then we will let j increase 1 which means that we map u to the next vertex in P^* . At the same time, we also let $|D_s| = |D_s| + 1$ and $FVN = FVN - 1$ to determine if we need to do levelup. Note that each P_i in \mathbb{P}^{even} has at least one division. Therefore, when we finish scanning a subpath, we will increase FVN since we have computed a division in a subpath in \mathbb{P}^{even} . A good property of P^* is that for all vertices in $4x$ -th positions in P^* where $x \in \mathbb{N}$, the ranking numbers of these vertices are more than 2. That is, we can determine if a vertex in P is mapped to a vertex in P^* with ranking number more than 2 by seeing if $j \bmod 4 = 0$. When we finish the vertices in $N(u) \cap \{V'_{1-2} \cup V'_3\}$, we mark u white to denote this vertex has been processed and keep going to scan the next vertex in P . If $FVN = 0$ and there is at least one vertex $v \in V'_{1-2} \cup V'_3$ which cannot do the first or second operation, we know that the rank limitation is broken. Hence we need to increase the rank of the resulting path. Thus we do *levelup*. The detail of our algorithm **MVRST-Permutation** is as follows.

Due to the space limitation of the paper, we omit the proof of the following theorem.

Theorem 1. *The MVRST of a permutation graph can be determined in linear time.*

Algorithm. MVRST-Permutation**Data:** A Permutation graph G .**Result:** An MVRST T of G .

1. Find a diameter P from the four shortest paths of G ;
2. **if** $|P| \leq 2^{\lfloor \log_2 |P| \rfloor} + 1$ **then**
 - └ done (i.e., $\text{rank}(T) = \lfloor \log_2 |P| \rfloor + 1$);
3. Partition $V \setminus P$ into $V_1, V'_{1-1}, V'_{1-3}, V'_{1-2}, V'_3$, and V'_2 ;
4. Connect V_1 and V'_{1-1} to P and mark the connected vertices in P black;
5. Partition P into P_1, P_2, \dots, P_s according to the black vertices;
- 5-1. Compute \mathbb{P}^{even} and D_s ;
- 5-2. $FVN = CMVRB - |P| - |\mathbb{P}^{\text{even}}| - |D_s|$;
- 5-3. **if** $FVN < 0$ **then**
 - └ done (i.e., $\text{rank} = \lfloor \log_2 |P| \rfloor + 2$ and do *levelup*);
- 5-4. **if** $V'_{1-2} \cup V'_3 = \emptyset$ **then**
 - └ done (i.e., $\text{rank} = \lfloor \log_2 |P| \rfloor + 1$);
6. Mark the vertices in P which can be connected to V'_{1-2} red, $V'_3 \setminus V_{ok}$ yellow, and both V'_{1-2} and V'_3 purple;
7. Scan the vertex u of P from left to right (initially, the position $j = 1$):
- 7-1. **if** the position $j \bmod 2 \neq 0$ and u is red **then**
 - └ Process-Red-Vertices(u);
 - └ scan the next vertex of u in P ;
- 7-2. **if** the position $j \bmod 2 \neq 0$ and u is yellow **then**
 - └ Process-Yellow-Vertices(u);
 - └ scan the next vertex of u' in P ;
- 7-3. **if** the position $j \bmod 2 \neq 0$ and u is purple **then**
 - └ **if** $\forall v \in N(u) \cap V'_{1-2} \exists w \in V'_2 \cup V'_3$ such that $\exists u' \in N(w) \cap P$ is mapped to $4x$ -th position in P^* **then**
 - └ do Step 7-2;
 - └ **else**
 - └ do Step 7-1;
- 7-4. **if** u is black **then**
 - └ **if** the left-hand vertex u' of u is mapped to $2x$ -th vertex in P^* **then**
 - └ map u to the $(j + 1)$ -th vertex in P^* ;
 - └ $\text{division} = \text{division} + 1$ and $FVN = FVN - 1$;
 - └ **if** u' is not black and u' is in an even subpath **then**
 - └ **if** $\text{division} > 1$ **then**
 - └ $FVN = FVN + 1$;
 - └ **if** u' is black **then**
 - └ $FVN = FVN + 1$;
 - └ $\text{division} = 0$;
 - └ **if** $FVN < 0$ **then**
 - └ done (i.e., $\text{rank} = \lfloor \log_2 |P| \rfloor + 2$ and do *levelup* on original P);
 - └ scan the next vertex of u in P ;
8. Connect $a \in V'_2$ to the even position of P ;
- Connect $b \in V'_{1-3}$ to the $4x$ -th position of P where $x \in \mathbb{N}$;
- Finally, we obtain T and $\text{rank}(T) = \lfloor \log_2 |P| \rfloor + 1$;
- return** The spanning tree T ;

3 Conclusion

In this paper, we show that the MVRST problem on permutation graphs can be solved in linear time. Currently, only few results about this problem are known. It seems that the diameter is important for this problem if the diameter is also a dominating path in the input graph. Thus, it seems that our idea can be generalized to the class of AT-free graphs.

References

1. Bhattacharjee, A., Hasan, C.S., Kashem, M.A.: An Algorithm for Solving the Minimum Vertex Ranking Spanning Tree Problem on Series-Parallel Graphs. In: 4th International Conference on Electrical and Computer Engineering, pp. 328–332 (2006)
2. Chang, R.-Y., Lee, G., Peng, S.-L.: Minimum Vertex Ranking Spanning Tree Problem on Some Classes of Graphs. In: International Conference on Intelligent Computing, pp. 758–765 (2008)
3. Deng, H., Guha, S., Sen, A.: On a Graph Partition Problem with Application to VLSI Layout. *Information Processing Letters* 43, 87–94 (1992)
4. Greenlaw, R., Schäffer, A.A., de la Torre, P.: Optimal Edge Ranking of Trees in Polynomial Time. *Algorithmica* 13, 592–618 (1995)
5. Iyer, A.V., Ratliff, H.D., Vijayan, G.: Parallel Assembly of Modular Products—an Analysis. Technical Report, Georgia Institute of Technology (1988)
6. Iyer, A.V., Ratliff, H.D., Vijayan, G.: On Edge Ranking Problems of Trees and Graphs. *Discrete Applied Mathematics* 30, 43–52 (1991)
7. Leiserson, C.E.: Area Efficient Graph Layouts for VLSI. In: 21st Annual IEEE Symposium of Foundations of Computer Science, pp. 270–281 (1980)
8. Masuyama, S., Miyata, K., Nakayama, S., Zhao, L.: NP-Hardness Proof and an Approximation Algorithm for the Minimum Vertex Ranking Spanning Tree Problem. *Discrete Applied Mathematics* 154, 2402–2410 (2006)
9. Masuyama, S., Nakayama, S.: An Algorithm for Solving the Minimum Vertex Ranking Spanning Tree Problem on Interval Graphs. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* 86-A(5), 1019–1026 (2003)
10. Masuyama, S., Nakayama, S.: An $O(n^3)$ Time Algorithm for Obtaining the Minimum Vertex Ranking Spanning Tree on Permutation Graphs. In: 4th Japanese-Hungarian Symposium on Discrete Mathematics and Its Applications (2005)
11. Masuyama, S., Nakayama, S.: A Polynomial Time Algorithm for Obtaining a Minimum Vertex Ranking Spanning Tree in Outerplanar Graphs. *IEICE Transactions on Information and Systems* 89-D(8), 2357–2363 (2006)
12. Nevins, J., Whitney, D.: *Concurrent Design of Products and Processes*. McGraw-Hill, New York (1989)

On Parameterized Exponential Time Complexity

Jianer Chen¹, Iyad A. Kanj², and Ge Xia³

¹ Department of Computer Science and Engineering, Texas A&M University,
College Station, TX 77843, USA

chen@cs.tamu.edu

² School of CTI, DePaul University, 243 S. Wabash Avenue, Chicago, IL 60604, USA

ikanj@cs.depaul.edu

³ Department of Computer Science, Lafayette College, Easton, PA 18042, USA

gexia@cs.lafayette.edu

Abstract. In this paper, we show that special instances of parameterized NP-hard problems are as difficult as the general instances in terms of their subexponential time computability. For example, we show that the PLANAR DOMINATING SET problem on degree-3 graphs can be solved in $2^{o(\sqrt{k})}p(n)$ parameterized time if and only if the general PLANAR DOMINATING SET problem can. Apart from their complexity theoretic implications, our results have some interesting algorithmic implications as well.

1 Introduction

Parameterized complexity theory [12] was motivated by the observation that many important NP-hard problems in practice are associated with a parameter whose value usually falls within a small or a moderate range. Thus, taking the advantage of the small size of the parameter may significantly speedup the computation. Formally, a *parameterized problem* consists of instances of the form (x, k) , where x is the *problem description* and k is an integer called the *parameter*. A parameterized problem is *fixed parameter tractable* if it can be solved by an algorithm of running time $f(k)n^{O(1)}$, where f is a function independent of the input size $n = |x|$.

Recently, a lot of progress has been made in the design of efficient algorithms for parameterized problems. As a case study, consider a canonical problem in parameterized complexity theory—the parameterized VERTEX COVER problem: given a graph G and a parameter k , decide if G has a vertex cover of at most k vertices. Since the development of the first parameterized algorithm for the problem which runs in $O(kn + 2^k k^{2k+2})$ time (described in [3]), there has been a long list of improved algorithms for the problem, whose running time is of the form $c^k n^{O(1)}$, where c is a constant progressively shown to be bounded by 1.3248, 1.3196, 1.2918, 1.2852, 1.2832, and 1.2745. The current best algorithm [10] runs in time $O(1.2738^k + kn)$ and uses polynomial space. It is natural to ask whether it is possible to reduce c from 1.2738 to a constant that is arbitrarily close to 1. More generally, we would like to know whether a parameterized problem can be solved in time $2^{\delta f(k)} n^{O(1)}$ for every constant $\delta > 0$.

A parameterized problem is solvable in *parameterized subexponential time* if it can be solved in time $2^{o(k)}p(n)$, where p is a polynomial. Very few parameterized NP-hard problems are known to be solvable in parameterized subexponential time, most of which are problems restricted to planar graphs. Alber et al. [1] gave parameterized subexponential time algorithms for the PLANAR VERTEX COVER, PLANAR INDEPENDENT SET, and PLANAR DOMINATING SET problems that run in time $2^{O(\sqrt{k})n}$. In particular, improving the upper bounds on the running time of subexponential time algorithms for PLANAR DOMINATING SET has been receiving a lot of attention [11, 14, 17]. Currently, the most efficient algorithm for PLANAR DOMINATING SET is by Fomin and Thilikos, and runs in $O(2^{15.13\sqrt{k}}n)$ time [14].

On the other hand, deriving lower bounds on the precise complexity of parameterized NP-hard problems has also started attracting more and more attention [4, 5, 7, 8]. Most of the known results in this line of research assume the so called Exponential Time Hypothesis (ETH): n -variable 3-SAT cannot be solved in time $2^{o(n)}$. Cai and Juedes [4] proved that certain parameterized problems such as VERTEX COVER, MAX CUT, MAX C-SAT cannot be solved in $2^{o(k)}p(n)$ time unless ETH fails, which is unlikely according to the common belief among researchers in the field. Similarly, they also showed that certain constraint parameterized problems such as PLANAR VERTEX COVER, PLANAR INDEPENDENT SET, and PLANAR DOMINATING SET cannot be solved in $2^{o(\sqrt{k})}p(n)$ time unless ETH fails. Subsequently, Chen et al. [5, 7, 8] showed that a large class of parameterized problems, including WEIGHTED SAT, DOMINATING SET, HITTING SET, SET COVER, and FEATURE SET cannot be solved in time $f(k)n^{o(k)}$, for any function f , unless the first level $W[1]$ of the W-hierarchy collapses to FPT.

In this paper we show that restricted instances of well-know parameterized NP-hard problems are as difficult as the general instances in terms of their parameterized subexponential time computability. In particular, we show that the PLANAR DOMINATING SET problem on degree-3 graphs (henceforth abbreviated PLANAR-3DS) can be solved in $2^{o(\sqrt{k})}p(n)$ (p is a polynomial) parameterized time if and only if the general PLANAR DOMINATING SET (abbreviated PLANAR-DS) problem can. Our results parallel the result in [16] for the INDEPENDENT SET problem, in the context of the standard exponential time computability.

Apart from their complexity theoretic implications, our results also have an algorithmic flavor. For instance, in our proof of the above mentioned result we give a reduction from PLANAR-DS to PLANAR-3DS. This reduction shows that if PLANAR-3DS can be solved in time $O(2^{5\sqrt{k}/7}n)$, then the PLANAR-DS problem can be solved in time $O(2^{15\sqrt{k}}n)$. Given that the currently most efficient algorithm for PLANAR-DS has running time $O(2^{15.13\sqrt{k}}n)$ [14], and that the structure of the PLANAR-3DS problem looks much simpler than that of PLANAR-DS, one could see a possibility of improving the algorithms for PLANAR-DS by working on PLANAR-3DS.

Throughout the paper, we assume basic familiarity with graphs and standard NP-hard problems. The reader is referred to [11, 15] for more details.

2 Preliminaries

We first give precise definitions for the notion that a parameterized problem can be solved in parameterized subexponential time.

Definition 1. A parameterized problem Q is solvable in time $O(2^{\delta f(k)}p(n))$ (p is a polynomial) for every constant $\delta > 0$ if there exists a parameterized algorithm A for Q such that, on any given instance (x, k) of Q with $|x| = n$, and any constant $\delta > 0$, the running time of the algorithm A is bounded by $h_\delta 2^{\delta f(k)}p(n)$, where h_δ is independent of k and n .

Definition 2. A parameterized problem Q is solvable in time $2^{o(f(k))}q(n)$, where q is a polynomial, if there exists a nondecreasing unbounded function $r(k)$ such that the problem Q can be solved in time $O(2^{f(k)/r(k)}q(n))$, where the constant hidden in the $O()$ notation is independent of k and n .

Based on the above definitions, we have the following equivalence theorem that can be derived from Lemma 16.1 of [13].

Theorem 1 ([13], Lemma 16.1). *Let $f(k)$ be a nondecreasing and unbounded function, and let Q be a parameterized problem. Then the following statements are equivalent:*

- (1) Q can be solved in time $O(2^{\delta f(k)}p(n))$ for every constant $\delta > 0$, where p is a polynomial;
- (2) Q can be solved in time $2^{o(f(k))}q(n)$, where q is a polynomial.

3 VC and VC-3

A set of vertices C is a *vertex cover* for a graph G if every edge in G is incident to at least one vertex in C . In the parameterized VC problem (shortly the VC problem) we are given a pair (G, k) as input, where G is an undirected graph and k is a positive integer (the parameter), and we are asked to decide if G has a vertex cover of size bounded by k . The VC-3 problem is the set of instances of the VC problem in which the underlying graph has degree bounded by 3. For a graph G , denote by $\tau(G)$ the size of a minimum vertex cover of G . We will show in this section that the VC-3 problem can be solved in parameterized subexponential time if and only if the general VC problem can. Let (G, k) be an instance of the VC problem. We will need the following propositions.

Proposition 1. [NT-Theorem] ([219]). *There is an $O(\sqrt{nm})$ time algorithm that, given a graph G of n vertices and m edges, constructs two disjoint subsets C_0 and V_0 of vertices in G such that*

- (1) *Every minimum vertex cover of $G(V_0)$ plus C_0 forms a minimum vertex cover for G ; and*
- (2) *A minimum vertex cover of $G(V_0)$ contains at least $|V_0|/2$ vertices.*

Proposition 1 allows us to assume, without loss of generality, that in an instance (G, k) of the VC problem, the graph G contains at most $2k$ vertices.

Let v be a degree-2 vertex in the graph G with two neighbors u and w such that u and w are not adjacent. We construct a new graph G' as follows: remove the vertices $v, u,$ and w and introduce a new vertex v_0 adjacent to all neighbors of the vertices u and w in G (of course except the vertex v). We say that the graph G' is obtained from the graph G by *folding* the vertex v . See Figure 1 for an illustration of this operation.

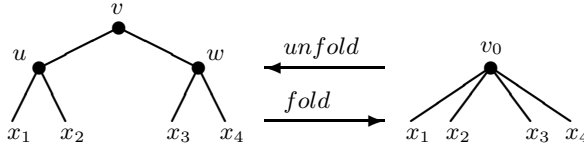


Fig. 1. Vertex folding and unfolding

Proposition 2 ([9]). *Let G' be a graph obtained by folding a degree-2 vertex v in a graph G , where the two neighbors of v are not adjacent to each other. Then $\tau(G) = \tau(G') + 1$. Moreover, a minimum vertex cover for G can be constructed from a minimum vertex cover for G' in linear time.*

We define an inverse operation of the folding operation that we call *unfold*. Given a vertex v_0 in a graph G where the degree of the vertex $d(v_0) > 3$, and with neighbors x_1, \dots, x_r (in an arbitrary order), we construct a graph G' as follows. Remove v_0 and introduce three new vertices $v, u,$ and w . Connect v to u and w , connect u to x_1 and x_2 , and connect w to x_3, \dots, x_r (see Figure 1). For the parameterized VC problem, we accordingly increase the parameter k by 1.

From Proposition 2, we know that $\tau(G') = \tau(G) + 1$. Moreover, the $unfold(v_0)$ operation replaces v_0 with three new vertices: v of degree 2, u of degree 3, and w of degree $d(v_0) - 1$. Now if $d(w) > 3$, we can apply the $unfold(w)$ operation, and so on, until all the newly introduced vertices have degree bounded by 3. It is easy to check that exactly $d(v_0) - 3$ operations are needed to replace v_0 by new vertices each having a degree bounded by 3. Let us call this iterative process initiated at the vertex v_0 *iterative-unfold* (v_0) . If G'' is the resulting graph from G after applying $iterative-unfold(v_0)$, then from the above discussion we have $\tau(G'') = \tau(G) + d(v_0) - 3$. Since each $unfold()$ operation increases the number of vertices in the graph by 2, the number of vertices n'' in G'' is $n + 2d(v_0) - 6$, where n is the number of vertices in G .

Theorem 2. *The VC-3 problem can be solved in $2^{o(k)}p(n)$ time if and only if the VC problem can be solved in $2^{o(k)}q(n)$ time, where n is the number of vertices in the graph, and p, q are two polynomials.*

Proof. Obviously, if VC can be solved in $2^{o(k)}q(n)$ time then so can VC-3. To prove the other direction, suppose that VC-3 can be solved in $2^{o(k)}p(n)$ time for

some polynomial p . By Theorem [1](#), VC-3 can be solved in time $O(2^{\epsilon k} p(n))$ for every $0 < \epsilon < 1$. To show that VC can be solved in time $2^{o(k)} q(n)$, by Theorem [1](#), it suffices to show that it can be solved in $O(2^{\delta k} q(n))$ time (q is a polynomial) for every $0 < \delta < 1$. Let (G, k) be an instance of the VC problem, and let $0 < \delta < 1$ be given. Consider the scheme in Figure [2](#).

VC-scheme

Input: an instance (G, k) of VC and a constant $0 < \delta < 1$

Output: a vertex cover C of G of size bounded by k in case it exists

1. apply Proposition [1](#) to G ; (without loss of generality, assume the resulting instance is (G, k))
2. let $d = \lceil -2 \lg(2^{\delta/2} - 1) / \delta \rceil$;
3. **while** there exists a vertex v of degree $> d$ **do**
 branch by **either** include v in the vertex cover and recursively call **VC-scheme** $(G - v, k - 1)$ **or** include $N(v)$ in the vertex cover and recursively call **VC-scheme** $(G - v - N(v), k - |N(v)|)$;
4. **while** there exists a vertex v of degree > 3 **do**
 iterative-unfold (v) ;
5. call the VC-3 scheme on the resulting graph and parameter, with $\epsilon = \delta / (4d - 10)$, to compute the desired vertex cover C' in case it exists;
6. **if** C' exists **then** using Proposition [2](#) and C' output the desired vertex cover C of G ;

Fig. 2. A scheme for VC

We are implicitly assuming that at each step the above algorithm, the graph and the parameter are updated appropriately. Keeping this in mind, it is not difficult to see the correctness of the above algorithm. The only step that may need additional explanation is step 3. Basically, in step 3, we remove large degree vertices by branching on them and creating subproblems. For any vertex v in G , it is easy to see that either there exists a minimum vertex cover containing v , or a minimum vertex cover containing its set of neighbors $N(v)$. In the first case, we remove the vertex v from G and reduce the parameter k by 1. In the latter case, we remove the vertex v and its neighbors $N(v)$ from G and reduce the parameter k by $|N(v)|$. In both cases, we recursively call **VC-scheme** on the new graph and the new parameter. Thus, the branch in step 3 is correct. Also note that at the end of step 4 every vertex in the resulting graph has degree bounded by 3. The correctness of the other steps follows from Proposition [1](#) and Proposition [2](#).

We analyze the running time of the algorithm. By proposition [1](#), step 1 takes polynomial time in n , and the resulting parameter is not larger than the initial parameter k . In the branching of step 3, we reduce the parameter k either by 1 or by $|N(v)| \geq d + 1$. Let $T(k)$ denote the number of leaves in the resulting branch-tree as a function of the parameter k . Then we have a recurrence relation

$T(k) \leq T(k - d - 1) + T(k - 1)$, which has a solution $T(k) = O(r^k)$, where r is the unique root of the polynomial $p(x) = x^{d+1} - x^d - 1$ in the interval $[1, 2]$ (see [9]). It is easy to verify that, with the choice of d in step 2, this recurrence relation has solution $T(k) = O(2^{\delta k/2})$ [9]. In step 4 we apply the subroutine **iterative-unfold** to every vertex of degree > 3 . For every vertex v in the graph of degree > 3 , **iterative-unfold**(v) can increase the parameter by no more than $d(v) - 3 \leq d - 3$, and the number of vertices in the graph by no more than $2(d - 3)$, since at this point of the algorithm the degree of the graph is bounded by d (note that $d > 3$). By Proposition 1, the number of vertices in the graph is bounded by $2k$, and hence, after step 4, the new parameter k' is bounded by $k + 2k(d - 3) = (2d - 5)k$, and the number of vertices n' is bounded by $2k + 4k(d - 3) = (4d - 10)k$. Clearly, the running time of step 4 is polynomial. In step 5, the **VC-3** scheme is called with $\epsilon = \delta/(4d - 10)$. By our assumption, the **VC-3** scheme runs in time $O(2^{\epsilon k'} p(n))$. It follows that step 5 takes time $O(2^{\delta(2d-5)k/(4d-10)} p(n)) = O(2^{\delta k/2} p(n))$. Since step 3 creates $T(k) = O(2^{\delta k/2})$ subproblems, and each subproblem can be solved in time $O(2^{\delta k/2} q(n))$, for some polynomial q , the total running time of the algorithm is $O(2^{\delta k/2} \cdot 2^{\delta k/2} q(n)) = O(2^{\delta k} q(n))$. The theorem follows. \square

4 Planar-DS and Planar-3DS

A *dominating set* D in a graph G is a set of vertices such that every vertex in G is either in D or adjacent to a vertex in D . The parameterized PLANAR-DS problem takes as input a pair (G, k) , where G is a planar graph, and asks to decide if G has a dominating set of size bounded by k . The PLANAR-3DS problem is the PLANAR-DS problem restricted to graphs of degree bounded by 3. For a graph G , denote by $\eta(G)$ the size of a minimum dominating set in G . Let (G, k) be an instance of the PLANAR-DS problem. We will need the following propositions.

Proposition 3. ([6]). *There is an $O(n^3)$ time algorithm that, given an instance (G, k) of PLANAR-DS, where G has n vertices, produces an instance (G', k') of PLANAR-DS, where G' has n' vertices, such that: (1) $n' \leq n$ and $k' \leq k$; (2) $n' \leq 67k'$; (3) G' has a dominating set of size $\leq k'$ if and only if G has a dominating set of size $\leq k$; and (4) from a solution D' of G' a solution D of G can be constructed in linear time.*

By Proposition 3, we can assume that in an instance (G, k) of the PLANAR-DS problem, the graph G contains at most $67k$ vertices.

Assume that the planar graph G is embedded in the plane. Let v be a vertex in the graph G of degree > 3 and let w_1, \dots, w_r , $r > 3$, be the neighbors of v . Without loss of generality, assume that they appear in a counter-clockwise order around v . We construct a new graph G' from G as follows. Remove v and introduce four new vertices x, x', y, y' . Connect x to w_1 and w_2 , y to w_3, \dots, w_r , x' to x , y' to y , and x' to y' . We say that the graph G' is obtained from the graph G by *expanding* the vertex v . It is clear that this operation can be carried

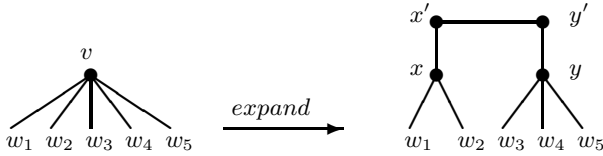


Fig. 3. Vertex expansion

out while preserving the planarity of the graph. See Figure 3 for an illustration of this operation.

The following theorem shows that the above vertex expansion operation can be used to reduce the maximum vertex-degree of a graph without significantly reducing the difficulty of the minimum dominating set problem.

Theorem 3. *Let G' be a graph obtained by expanding a vertex v of degree > 3 in a planar graph G . Then $\eta(G) = \eta(G') - 1$. Moreover, a minimum dominating set for G can be constructed from a minimum dominating set for G' in linear time.*

Proof. We first show that $\eta(G') \leq \eta(G) + 1$. Let D be a minimum dominating set for G . If D contains v , then clearly $(D - \{v\}) \cup \{x, y\}$ is a dominating set for G' of size $\eta(G) + 1$. If D does not contain v , then D must contain at least one vertex in $\{w_1, \dots, w_r\}$ (since v must be dominated). If D contains a vertex in $\{w_1, w_2\}$ then $D \cup \{y'\}$ is a dominating set for G' of size $\eta(G) + 1$, whereas if D contains a vertex in $\{w_3, \dots, w_r\}$, then $D \cup \{x'\}$ is a dominating set for G' of size $\eta(G) + 1$. It follows that in all cases G' has a dominating set of size $\eta(G) + 1$, and hence $\eta(G') \leq \eta(G) + 1$. Now to prove that $\eta(G) \leq \eta(G') - 1$, let D' be a minimum dominating set for G' . We distinguish the following cases.

Case 1. D' contains both x and y . In this case $(D' - \{x, y, x', y'\}) \cup \{v\}$ is a dominating set for G of size bounded by $\eta(G') - 1$.

Case 2. D' contains exactly one vertex in $\{x, y\}$, without loss of generality, let this vertex be x (the other case is symmetrical). Then D' must contain at least one vertex in $\{x', y'\}$. Thus, $(D' - \{x, x', y'\}) \cup \{v\}$ is a dominating set for G of size bounded by $|D'| - 1 = \eta(G') - 1$.

Case 3. D' does not contain any vertex in $\{x, y\}$, then D' has to contain at least one vertex in $\{x', y'\}$. If D' contains at least one vertex in $\{w_1, \dots, w_r\}$, then $D' - \{x', y'\}$ is a dominating set for G of size bounded by $\eta(G') - 1$. On the other hand if D' does not contain any vertex in $\{w_1, \dots, w_r\}$, then D' must contain both x' and y' in order to dominate x and y . Now $(D' - \{x', y'\}) \cup \{v\}$ is a dominating set for G' of size $\eta(G') - 1$.

Thus, in all cases G has a dominating set of size bounded by $\eta(G') - 1$. It follows that $\eta(G) \leq \eta(G') - 1$, and hence, $\eta(G') = \eta(G) + 1$. Moreover, given a dominating set D' of G' , it should be clear how the corresponding dominating set D of G can be constructed in linear time according to one of the above three cases. □

If v is a vertex in G such that $d(v) > 3$, the operation $\text{expand}(v)$ replaces v with four new vertices: x of degree 3, x' of degree 2, y' of degree 2, and y of degree $d(v) - 1$. If $d(y) > 3$, we can apply the $\text{expand}(y)$ operation, and so on, until all the newly introduced vertices have degree bounded by 3. Again exactly $d(v) - 3$ operations are needed to replace v by new vertices each having a degree bounded by 3. We denote this iterative process initiated at the vertex v *iterative-expand*(v). If G'' is the resulting graph from G after applying *iterative-expand*(v), then we have $\eta(G'') = \eta(G) + d(v) - 3$, and the number of vertices n'' of G'' is $n + 3d(v) - 9$.

Theorem 4. *The PLANAR-3DS problem can be solved in $2^{o(\sqrt{k})}p(n)$ time if and only if the PLANAR-DS problem can be solved in $2^{o(\sqrt{k})}q(n)$ time, where n is the number of vertices in the graph, and p, q are two polynomials.*

Proof. The proof of this theorem has the same flavor as Theorem 2. First if PLANAR-DS can be solved in $2^{o(\sqrt{k})}q(n)$ time then so can PLANAR-3DS. To prove the other direction, we suppose that PLANAR-3DS can be solved in time $O(2^{\epsilon\sqrt{k}}p(n))$ for any $0 < \epsilon < 1$, and for some polynomial p , and we show that PLANAR-DS can be solved in $O(2^{\delta\sqrt{k}}q(n))$ time (q is a polynomial) for every $0 < \delta < 1$. By Theorem 1, this will be sufficient. Let (G, k) be an instance of the PLANAR-DS problem, and let $0 < \delta < 1$ be given. Consider the scheme in Figure 4.

DS-scheme
 Input: an instance (G, k) of PLANAR-DS and a constant $0 < \delta < 1$
 Output: a dominating set D of G of size bounded by k in case it exists

1. apply Proposition 3 to G ;
2. **while** there exists a vertex v of degree > 3 **do** **iterative-expand**(v);
3. call the PLANAR-3DS scheme on the resulting graph and parameter, with $\epsilon = \delta/\sqrt{403}$, to compute the desired dominating set D' in case it exists;
4. **if** D' exists **then** using Theorem 3 and D' output the desired dominating set D of G ;

Fig. 4. A scheme for Planar-DS

The analysis of the algorithm and its correctness follows a similar line to that of Theorem 2. However, few things need to be clarified. First, after step 1, we know by Proposition 3 that the number of vertices n in G is bounded by $67k$. In step 2, the **iterative-expand**() operation increases both the parameter and the number of vertices in G . Let G' be the resulting graph at the end of step 2, and let k' and n' be the parameter and number of vertices in G' , respectively. Each call to **iterative-expand**(v), where $d(v) > 3$, increases k by $d(v) - 3$ and n by $3d(v) - 9$. It follows that

$$\begin{aligned}
 k' &= k + \sum_{v \in G, d(v) > 3} (d(v) - 3) \leq k + \sum_{v \in G} d(v) \\
 &\leq k + 6n - 12 < k + 402k = 403k.
 \end{aligned}
 \tag{1}$$

The last two inequalities follow from the fact that the number of edges in a planar graph of n vertices is bounded by $3n - 6$ [11], and from Proposition 3. Similarly, we can show that $n' \leq 19n$. It is easy now to see that the theorem follows. \square

Observing that by Inequality (1), $\sqrt{k'} < \sqrt{403k} < 21\sqrt{k}$, we have the following corollary.

Corollary 1. *If the PLANAR-3DS problem can be solved in $2^{5\sqrt{k}/7}p(n)$ time then the PLANAR-DS problem can be solved in $2^{15\sqrt{k}}q(n)$ time, where n is the number of vertices in the graph, and p, q are two polynomials.*

5 MAX-SAT and MAX-CUT

The parameterized MAX-SAT problem is defined as follows. Given a boolean formula F in conjunctive normal form and a positive integer k , decide if F has a truth assignment that satisfies k or more clauses. The parameterized MAX-3SAT problem is the parameterized MAX-SAT problem restricted to formulas in which each clause contains at most three literals. Mahajan and Raman [18] showed the following.

Proposition 4 ([18]). *Given a formula F and a positive integer k , then in linear time, we can compute a formula F' and a positive integer $k' \leq k$ with the length of the formula $|F'| \in O(k'^2)$, such that F has an assignment satisfying at least k clauses if and only if F' has an assignment satisfying at least k' clauses. Moreover, such an assignment for F is computable from an assignment for F' in linear time.*

Using Proposition 4, the standard reduction from SAT to 3-SAT [15], and Theorem 1, we can show the following theorem.

Theorem 5. *If parameterized MAX-3SAT can be solved in time $2^{o(\sqrt{k})}p(n)$ then parameterized MAX-SAT can be solved in time $2^{o(k)}q(n)$, where p and q are two polynomials.*

In the parameterized MAX-CUT problem we are given an undirected graph G and a positive integer k , and we are asked to decide if the vertex set of G can be partitioned into two parts so that at least k edges cross the partitioning. The parameterized MAX-3CUT problem is the parameterized MAX-CUT problem on graphs of degree bounded by 3. Using the techniques similar to those used in the previous sections, we have the following theorem.

Theorem 6. *Parameterized MAX-3CUT can be solved in time $2^{o(k)}p(n)$ if and only if parameterized MAX-CUT can be solved in time $2^{o(k)}q(n)$, where p and q are two polynomials.*

References

1. Alber, J., Bodlaender, H.L., Ferneau, H., Niedermeier, R.: Fixed parameter algorithms for Dominating Set and related problems on planar graphs. *Algorithmica* 33, 461–493 (2002)
2. Bar-Yehuda, R., Even, S.: A local-ratio theorem for approximating the weighted vertex cover problem. *Annals of Discrete Mathematics* 25, 27–46 (1985)
3. Buss, J., Goldsmith, J.: Nondeterminism within P. *SIAM Journal on Computing* 22, 560–572 (1993)
4. Cai, L., Juedes, D.: On the existence of subexponential parameterized algorithms. *Journal of Computer and System Sciences* 67(4), 789–807 (2003)
5. Chen, J., Chor, B., Fellows, M.R., Huang, X., Juedes, D.W., Kanj, I.A., Xia, G.: Tight lower bounds for certain parameterized NP-hard problems. *Information and Computation* 201(2), 216–231 (2005)
6. Chen, J., Fernau, H., Kanj, I.A., Xia, G.: Parametric duality and kernelization: lower bounds and upper bounds on kernel size. *SIAM Journal on Computing* 37(4), 1077–1106 (2007)
7. Chen, J., Huang, X., Kanj, I.A., Xia, G.: Linear FPT reductions and computational lower bounds. In: *STOC 2004*, pp. 212–221 (2004)
8. Chen, J., Huang, X., Kanj, I.A., Xia, G.: Strong Computational Lower Bounds via Parameterized Complexity. *Journal of Computer and System Sciences* 72(8), 1346–1367 (2006)
9. Chen, J., Kanj, I.A., Jia, W.: Vertex cover: further observations and further improvements. *Journal of Algorithms* 41, 280–301 (2001)
10. Chen, J., Kanj, I.A., Xia, G.: Improved Parameterized Upper Bounds for Vertex Cover. In: Kráľovič, R., Urzyczyn, P. (eds.) *MFCS 2006*. LNCS, vol. 4162, pp. 238–249. Springer, Heidelberg (2006)
11. Diestel, R.: *Graph Theory*. Springer, New York (1996)
12. Downey, R.G., Fellows, M.R.: *Parameterized Complexity*. Springer, Heidelberg (1999)
13. Flum, J., Grohe, M.: *Parameterized Complexity Theory*. Springer, Heidelberg (2006)
14. Fomin, F., Thilikos, D.: Dominating sets in planar graphs: branch-width and exponential speed-up. *SIAM Journal on Computing* 36(2), 281–309 (2006)
15. Garey, M., Johnson, D.: *Computers and Intractability: A Guide to the Theory of NP-completeness*. Freeman, San Francisco (1979)
16. Johnson, D., Szegedy, M.: What are the least tractable instances of max independent set? In: *SODA 1999*, pp. 927–928 (1999)
17. Kanj, I.A., Perkovic, L.: Improved parameterized algorithms for planar dominating set. In: Diks, K., Rytter, W. (eds.) *MFCS 2002*. LNCS, vol. 2420, pp. 399–410. Springer, Heidelberg (2002)
18. Mahajan, M., Raman, V.: Parameterizing above guaranteed values: MAX-SAT and MAX-CUT. *Journal of Algorithms* 31, 335–354 (1999)
19. Nemhauser, G.L., Trotter, L.E.: Vertex packing: structural properties and algorithms. *Mathematical Programming* 8, 232–248 (1975)

Best-Order Streaming Model

Atish Das Sarma, Richard J. Lipton, and Danupon Nanongkai

Georgia Institute of Technology
{atish,rjl,danupon}@cc.gatech.edu

Abstract. We study a new model of computation called *stream checking* on graph problems where a space-limited verifier has to verify a proof sequentially (i.e., it reads the proof as a stream). Moreover, the proof itself is nothing but a reordering of the input data. This model has a close relationship to many models of computation in other areas such as data streams, communication complexity, and proof checking and could be used in applications such as cloud computing.

In this paper we focus on graph problems where the input is a sequence of edges. We show that checking if a graph has a perfect matching is impossible to do deterministically using small space. To contrast this, we show that randomized verifiers are powerful enough to check whether a graph has a perfect matching or is connected.

1 Introduction

This paper is motivated by three fundamental questions that arise in three widely studied areas in theoretical computer science - streaming algorithms, communication complexity, and proof checking. The first question is how efficient can space restricted streaming algorithms be. The second question, is whether the hardness of a communication problem holds for every partition of the input. Finally, in proof checking, the question is how many (extra) bits are needed for the verifier to establish a proof in a restricted manner. Before elaborating these questions, we first describe one application that motivates our model.

Many big companies such as Amazon [1] and salesforce.com are currently offering *cloud computing* services. These services allow their users to use the companies' powerful resources for a short period of time, over the Internet. They also provide some softwares that help the users who may not have knowledge of, expertise in, or control over the technology infrastructure ("in the cloud") that supports them. [2] These services are very helpful, for example, when a user wants a massive computation over a short period of time.

Now, let's say that you want the cloud computer to do a simple task such as checking if a massive graph is strongly connected. Suppose that the cloud computer gets back to you with an answer "Yes" suggesting that the graph is strongly connected. What do you make of this? What if there is a bug in the code, or what if there was some communication error? Ideally one would like a

¹ http://www.ebizq.net/blogs/saasweek/2008/03/distinguishing_cloud_computing/

way for the cloud to *prove* to you that the answer is correct. This proof might be long due to the massive input data; hence, it is impossible to keep everything in your laptop's main memory. Therefore, it is more practical to read the proof as a *stream* with a small working memory. Moreover, the proof should not be too long – one ideal case is when the proof is the input itself (in different order). This is the model considered in this paper.

Coincidentally, this model has connections with many previously studied models in many areas. We now continue with describing previous models studied specifically in the stream, computational complexity and proof checking domains and contrast them with our model.

Data Streams: The basic premise of streaming algorithms is that one is dealing with a humongous data set, too large to process in main memory. The algorithm has only sequential access to the input data; this called a *stream*. In certain settings, it is acceptable to allow the algorithm to perform multiple passes over the stream. However, for many applications, it is not feasible to perform more than a single pass. The general streaming algorithms framework has been studied extensively since the seminal work of Alon, Matias, Szegedy [3].

Models diverge in the assumptions made about what order the algorithm can access the input elements. The most stringent restriction on the algorithm is to assume that the input sequence is presented to the algorithm in an adversarial order. A slightly more relaxed setting, that has also been widely studied is where the input is assumed to be presented in randomized order [7,15,16]. However, even a simple problem like finding median, which was considered in the earliest paper in the area by Munro and Patterson [24], in both input orders, was shown recently [7] to require many passes even when the input is in a random order (to be precise, any $O(\text{polylog } n)$ algorithm requires $\Omega(\log \log n)$ passes). This might be undesirable.

More bad news: Graph problems are extremely hard when presented in an adversarial order. In [19], one of the earliest paper in this area, it was shown that many graph problems require prohibitively large amount of space to solve. It is confirmed by the more recent result [11] that most graph problems cannot be solved efficiently in a few passes. Since then, new models have been proposed to overcome this obstruction. Feigenbaum et. al. [12] proposed a relaxation of the memory restriction in what is called the semi-stream model. Aggarwal et al. [2] proposed that if the algorithm has a power to sort the stream in one pass then it is easier to solve some graph problems (although not in one or constant passes). Another model that has been considered is the W-Stream (write-stream) model [26,8]. While the algorithm processes the input, it may also *write* a new stream to be read in the next pass.

We ask the following fundamental question:

If the input is presented in the best order possible, can we solve problems efficiently?

A precise explanation is reserved for the models in Section 2; however, intuitively, this means that the algorithm processing the stream can decide on a *rule* on

the order in which the stream is presented. We call this the best-order stream model. For an example, if the rule opted by the algorithm is to read the input in sorted order, then this is equivalent to the single pass sort stream model. Another example of a rule, for graphs presented as edge streams could be that the algorithm requires all edges incident on a vertex to be presented together. This is again equivalent to a graph stream model studied earlier called an incidence model (and corresponds to reading the rows of the adjacency matrix one after the other). A stronger rule could be that the algorithm asks for edges in some perfect matching followed by other edges. As we show in this paper, this rule leads to checking if the graph has a perfect matching and as a consequence shows the difference between our model and the sort-stream model.

It would be nice to obtain a characterization of problems that can be solved by a poly-log space, single pass, best-order stream algorithm. Studying this model, like all other related streaming models, is likely to yield new insights and might lead to an improvement of worst case analysis and an adjustment of models.

Communication Complexity: Another closely related model is the communication complexity model [27,20]. This model was extensively studied and found many applications in many areas. In the basic form of this model, two players, Alice and Bob, receive some input data and they want to compute some function together. The question is how much communication they have to make to accomplish the task. There are many variations of how the input is partitioned. The worst-case [21] and the best-case [25] partition models are two extreme cases that are widely studied over decades. The worst case asks for the partition that makes Alice and Bob communicate the most while the best case asks for the partition that makes the communication smallest. Moreover, even very recently, there is a study for another variation where the input is partitioned according to some known distribution (see, e.g., [6]). The main question is whether the hardness of a communication problem holds for almost every partition of the input, as opposed to holding for perhaps just a few atypical partitions.

The communication complexity version of our model (described in Section 2) asks the following similar question: Does the hardness of a communication problem hold for *every* partition of the input? Moreover, our model can be thought of as a more extreme version of the best-case partition communication complexity. We explain this in more details in Section 2.

Proof Checking: From a complexity theoretic standpoint, our model can be thought of as the case of proof checking where a polylog-space verifier is allowed to read the proof as a stream; additionally, the proof must be the input itself in a different order.

We briefly describe some work in the field of proof checking and its relation to our setting. The field of probabilistically checkable proofs (PCPs) [4,5,9] deals with verifier querying the proof at very few points (even if the data set is large and thus the proof) and using this to guarantee the proof with high probability. While several variants of proof checking have been considered, we only state the

most relevant ones. A result most related to our setting is by Lipton [23] where it showed that membership proofs for NP can be checked by probabilistic logspace verifiers that have one-way access to the proof and use $O(\log n)$ random bits. In other words, this result almost answers our question except that the proof is not the reordered input.

Another related result that compares streaming model with other models is by Feigenbaum et. al. [10] where the problem of testing and spot-checking on data streams is considered. They define sampling-tester and streaming-tester. A sampling-tester is allowed to sample some (but not all) of the input points, looking at them in any order. A streaming-tester, on the other hand is allowed to look at the entire input but only in a specific order. They show that some problems can be solved in a streaming-tester but not by a sampling-tester, while the reverse holds for other problems. Finally, we note that our model (when we focus on massive graphs) might remind some readers of the problem of property testing in massive graphs [13].

Notice that in all of the work above, there are two common themes. The first is verification using *small space*. The second is some form of *limited access* to the input. The limited access is either in the form of sampling from the input, limited communication, or some restricted streaming approach. Our model captures both these factors.

Our Results

In this paper, we partially answer whether there are efficient streaming algorithms when the input is in the best order possible. We give a negative answer to this question for the deterministic case and show an evidence of a positive answer for the randomized case. Our positive results are similar in spirit to W-stream and Sort-stream papers [2,8,26].

For the negative answer, we show that the space requirement is too large even for a simple answer of checking if a given graph has a perfect matching deterministically. In contrast, this problem, as well as the connectivity problem, can be solved efficiently by randomized algorithms.

Organization: The rest of the paper is organized as follow. In Section 2 we describe our stream proof checking model formally and also define some of the other communication complexity models that are well-studied. The problem of checking for distinctness in a stream of elements is discussed in Section 3. This is a building block for most of our algorithms. The following section, Section 4 talks about how perfect matchings can be checked in our model. We discuss the problem of stream checking graph connectivity in Section 5. Our techniques can be extended to a wide class of graph problems such as checking for regular bipartiteness, non-bipartiteness, hamiltonian cycles etc. While we are unable to mention all details in this paper due to space limitations, we describe the key ideas for these problems in Section 6. Finally, we conclude in Section 7 by stating some insights drawn from this paper, mention open problems and describe possible future directions.

2 Models

In this section we explain our main model and other related models that will be useful in subsequent sections.

2.1 Stream Proof Model

Recall the streaming model where an input is in some order e_1, e_2, \dots, e_m where m is the size of the input. Consider any function f that maps these input stream to $\{0, 1\}$. The goal of the typical one-pass streaming model is to calculate f using the smallest amount of memory possible.

In the *stream proof* model, we consider any function f that is order-independent. Our main question is how much space a one-pass streaming algorithm needs to compute f if the input is provided in the best order. Formally, for any function s of m and any function f , we say that a language L determined by f is in the class $\text{STREAM-PROOF}(s(m))$ if there exists an algorithm \mathcal{A} using space at most $s(m)$ such that if $f(e_1, e_2, \dots, e_m) = 1$ then there exists a permutation π such that $\mathcal{A}(e_{\pi(1)}, e_{\pi(2)}, \dots, e_{\pi(m)})$ answers 1; otherwise, $\mathcal{A}(e_{\pi(1)}, e_{\pi(2)}, \dots, e_{\pi(m)})$ answers 0 for *every* permutation π .

The other way to see this model is to consider the situation where there are two players in the setting, *prover* and *verifier*. The job of the prover is to provide the stream in some order so that the verifier can compute f using smallest amount of memory possible. We assume that the prover has unlimited power but restrict the verifier to read the input in a streaming manner.

The model above can be generalized to the following.

- $\text{STREAM}(p, s)$: A class of problems that, when presented with best-order, can be checked by a deterministic streaming algorithm \mathcal{A} using p passes $O(s)$ space.
- $\text{RSTREAM}(p, s)$: A class of problems that, when presented with best-order, can be checked by a randomized streaming algorithm \mathcal{A} using p passes $O(s)$ space and with correct probability more than $1/2$.

It is important to point out that when the input is presented in a specified order, we still need to check that the adversary is not *cheating*. That is, we indeed need a way to verify that we receive the input based on the rule we asked for. This often turns out to be the difficult step.

To contrast this model with well-studied communication complexity models, we first define a new communication complexity model, magic-partition, that closely relates to our proof checking model.

2.2 Magic-Partition Communication Complexity

In this subsection, we define magic-partition communication complexity which will be the main tool to prove the lower bound of the best-order streaming model.

Recall that in the standard 2-player communication complexity, Alice and Bob gets input x and y and want to compute $f(x, y)$. We usually consider when

the input is partitioned in an adversarial order, i.e., we partition input into x and y in such a way that Alice and Bob have to communicate as many bits as possible.

For the magic-partition communication complexity, we consider the case when x and y are partitioned in the best way possible. One way to think of this protocol is to imagine that there is an oracle who looks at the input and then decides how to divide the data between Alice and Bob so that they can compute f using smallest number of communicated bits. We restrict that the input data must be divided equally between Alice and Bob.

Let us consider an example. Suppose the input is a graph G . Alice and Bob might decide that the graph be broken down in topological sort order, and Alice receives the first half of the total edges, starting with edges incident on the vertices (traversing them in topological order). It is important to note the distinction that Alice and Bob actually have not seen the input; but they specify a *rule* by which to partition the input, when actually presented.

The following lemma is the key to prove our lower bound results.

Lemma 1. *For any function f , if the (deterministic) magic-partition communication complexity of f is at least s , for some s , then for any p and t such that $(2p - 1)t < s$, $f \notin \text{STREAM}(p, t)$.*

Proof. Suppose that the lemma is not true; i.e., f has magic-partition communication complexity at least s , for some s , but there is a best-order streaming algorithm \mathcal{A} that computes f using p passes and t space such that $(2p - 1)t < s$. Consider any input e_1, e_2, \dots, e_n . Let π be a permutation such that $e_{\pi(1)}, e_{\pi(2)}, \dots, e_{\pi(n)}$ is the best ordering of the input for \mathcal{A} . Then, define the partition of the magic-partition communication complexity by allocating $e_{\pi(1)}, e_{\pi(2)}, \dots, e_{\pi(\lfloor n/2 \rfloor)}$ to Alice and the rest to Bob.

Alice and Bob can simulate \mathcal{A} as follows. First, Alice simulates \mathcal{A} on $e_{\pi(1)}, e_{\pi(2)}, \dots, e_{\pi(\lfloor n/2 \rfloor)}$. Then, she sends the data on memory to Bob. Then, Bob continues simulating \mathcal{A} using data given by Alice (as if he simulates \mathcal{A} on $e_{\pi(1)}, e_{\pi(2)}, \dots, e_{\pi(\lfloor n/2 \rfloor)}$ by himself). He then sends the data back to Alice and the simulation of the second pass of \mathcal{A} begins.) Observe that this simulations need $2p - 1$ rounds of communication and each round requires at most t bits. Therefore, Alice and Bob can compute f using $(2p - 1)t < s$ bits, contradicting the original assumption. \square

Note that this type of communication complexity should not be confused with the best-partition communication complexity (defined below). Also, the converse of the above lemma clearly does not hold.

We now describe some previously studied communication complexity models that resemble ours.

2.3 Related Models

Best-case partition Communication Complexity. For this model, Alice and Bob can pick how to divide the data among them (must be half-half)

before they see the input. Then, the adversary gives an input that makes them communicate the most.

This model was introduced by Papadimitriou and Sipser [25] and heavily used for proving lower bounds for many applications (see [20] and references therein).

Similar to the best-partition communication complexity, this model makes many problems easier to solve than the traditional worst case communication complexity where the worst case input is assumed. One example is the set disjointness problem. In this problem, two n -bit vectors x and y that is a characteristic vector of two sets X and Y are given. Alice and Bob have to determine if $X \cap Y = \emptyset$.

In the worst case communication complexity, it is proved that Alice has to send roughly n bits to Bob when x is given to Alice and y is given to Bob. However, for the best-partition case, they can divide the input this way: $x_1, y_1, x_2, y_2, \dots, x_{n/2}, y_{n/2}$ go to Alice and the rest go to Bob. This way, each of them can check the disjointness separately.

We note that this model is different from the magic-partition model in that, in this model the players have to pick how data will be divided before they see the input data. For example, if the data is the graph of n vertices then, for any edge (i, j) , Alice and Bob have to decide who will get this edge if (i, j) is actually in the input data. However, in the magic-partition model, Alice and Bob can make a more complicated partitioning rule such as giving $(1, 2)$ to Alice *if* the graph is connected. (In other words, in the magic-partition model, Alice and Bob have an oracle that decide how to divide an input *after* he sees it).

One problem that separates these model is the connectivity problem. Hajnal et al. [17] showed that the best-case partition communication complexity of connectivity is $\Theta(n \log n)$. In contrast, we show that $O((\log n)^2)$ is possible in our model in this paper.

Nondeterministic Communication Complexity. Alice and Bob receives x and y respectively. An oracle, who sees x and y , wants to convince them that “ $f(x, y) = 1$ ”. He does so by giving them a proof. Alice and Bob should be able to verify the proof with small amount of communication.

Example: $f(x, y) = 1$ if $x \neq y$ where x and y are n -bit strings. The proof is simply the number i where $x_i \neq y_i$. Alice and Bob can check the proof by exchanging x_i and y_i . If $x = y$ then there is no proof and Alice and Bob can always detect the fake proof.

This model is different from our model because our model has no proof but the oracle’s job is to help Alice and Bob find the answer (whether $f(x, y)$ is 0 or 1) by appropriately partitioning the input.

3 Detecting Duplicate and Checking Distinctness

In this section, we consider the following problem which is denoted by `DISTINCT`. Given a stream of n numbers a_1, a_2, \dots, a_n where $a_i \in \{1, 2, \dots, n\}$. We want to check if every number appears exactly once (i.e., no duplicate). This problem

appears as a main component in solving all the problems we considered and we believe that it will be useful in every problem.

Our goal in this section is to find a one-pass algorithm for this problem. An algorithm for this problem will be an important ingredient of all algorithm we consider in this paper. In this section, we show that 1) any deterministic algorithm for this problem needs $\Omega(n)$ space, and 2) there is a randomized algorithm that solves this problem in $O(\log n)$ space with error probability $\frac{1}{n}$.

3.1 Space Lower Bound of Deterministic Algorithms

Since checking for distinctness is equivalent to checking if there is a duplicate, a natural problem to use as a lower bound is the *set disjointness problem*. We define a variation of this problem called *full set disjointness problem*, denoted by F-DISJ.

For this problem, a set $X \subseteq N$ is given to Alice and $Y \subseteq N$ is given to Bob where $N = \{1, 2, 3, \dots, n\}$ and $|X| + |Y| = n$ ²

Now we show that F-DISJ is hard for the deterministic case. The proof is the same as the proof of the set disjointness problem.

Theorem 1. *The communication complexity of F-DISJ is $\Omega(n)$.*

Proof. Consider the fooling set $F = \{(A, \bar{A}) : \forall A \subseteq N\}$. Since $|F| = 2^n$, the number of bits needed to sent between Alice and Bob is at least $\log |F| = \Omega(n)$. \square

The communication complexity lower bound of F-DISJ implies the space lower bound of DISTINCT.

Corollary 1. *Any deterministic algorithm for DISTINCT needs $\Omega(n)$ space.*

This lower bound is for worst-case input. The reason we mention this here is because this is an inherent difficulty in our algorithms. Our randomized algorithms use randomness only to get around this step of checking distinctness.

3.2 Randomized Algorithm

In this subsection we present a randomized one-pass algorithm that solves this problem using $O(\log n)$ space. This algorithm is based on the *Fingerprinting Sets* technique introduced by Lipton [22,23]. Roughly speaking, given a multi-set $\{x_1, x_2, \dots, x_k\}$, its *fingerprint* is defined to be

$$\prod_{i=1}^k (x_i + r) \pmod p$$

² Note that this problem is different from the well-known set disjointness problem in that we require $|X| + |Y| = n$. Although the two problems are very similar, they are different in that the set disjointness problem has $\Omega(n)$ randomized algorithm while the F-DISJ has an $O(\log n)$ randomized protocol (shown in the next section). We also note that the lower bound of another related problem called k -disjointness problem ([20, example 2.12] and [18]) does not imply our result neither.

where p is a random prime and $r \in \{0, 1, \dots, p-1\}$. We use the following property of the fingerprints.

Theorem 2. [23] *Let $\{x_1, x_2, \dots, x_k\}$ and $\{y_1, y_2, \dots, y_k\}$ be two multi-sets. If the two sets are equal then their fingerprints are always the same. Moreover, if they are unequal, the probability that they get the same fingerprints is at most*

$$O\left(\frac{\log b + \log m}{bm} + \frac{1}{b^2m}\right)$$

where all numbers are b -bit numbers and $m = \max(k, l)$ provided that the prime p is selected randomly from interval

$$[(bm)^2, 2(bm)^2].$$

Now, to check if a_1, a_2, \dots, a_n are all distinct, we simply check if the fingerprints of $\{a_1, a_2, \dots, a_n\}$ and $\{1, 2, \dots, n\}$ are the same. Here, $b = \log n$ and $m = n$. Therefore, the error probability is at most $1/n$.

Remark: We note that the fingerprinting sets can be used in our motivating application above. That is, when the cloud compute sends back a graph as a proof, we have to check whether this “proof” graph is the same as the input graph we sent. This can be done using the fingerprinting set. This enables us to concentrate on checking the stream without worrying about this issue in the rest of the paper.

We also note that the recent result by Gopalan et al. [14] can be modified to solve DISTINCT as well.

4 Perfect Matching

This section is devoted to the study of perfect matchings. We discuss lower bounds as well as upper bounds.

Problem: Given the edges of a graph G in a streaming manner e_1, e_2, \dots, e_m , we want to compute $f(e_1, \dots, e_m)$ which is 1 if and only if G has a perfect matching. Let n be the number of vertices. We assume that the vertices are labeled $1, 2, \dots, n$.

We now present the main upper bound of this section and follow it up with the checking protocol in the proof.

Theorem 3. *The problem of determining if there exists a perfect matching can be solved by a randomized algorithm in $O(\log n)$ space best-order stream checking.*

Proof. Protocol: The prover sends $n/2$ edges of a perfect matching to the verifier, followed by the “sign” which can be implemented by flipping the order of vertices in the last edge. Then the prover sends the rest edges. The verifier has to check three things.

1. Find out n by counting the number of edges before the “sign” is given.
2. Check if the first $n/2$ edges form a perfect matching. This can be done by checking if the sum of the labels of vertices in the first $n/2$ edges equals $1 + 2 + 3 + \dots + n$.
3. Check if there are n vertices. This is done by checking that the maximum vertex label is at most n .

The verifier outputs 1 if the input passes all the above tests. The correctness of this protocol is straightforward. \square

In the next subsection, we present a lower bound.

4.1 Hardness

We show that deterministic algorithms have $\Omega(n)$ lower bound if the input is reordered in an explicit way; i.e., each edge cannot be split. This means that an edge is either represented in the form (a, b) or (b, a) . The proof follows by a reduction from the magic-partition communication complexity (cf. Section 2) of the same problem by using Lemma 1.

Theorem 4. *If the input can be reordered only in an explicit way then any deterministic algorithm solving the perfect matching problem needs $\Omega(n)$ space.*

Proof. Let n be an even integer. Let $g(n)$ denote the number of matchings in the complete graph K_n . Observe that $g(n) = \frac{n!}{(n/2)!2^{n/2}}$. Denote these matchings by $M_1, M_2, \dots, M_{g(n)}$. Let \mathcal{P} be any magic-partition protocol. For any integer i , let A_i and B_i be the best partition of M_i according to \mathcal{P} (A_i and B_i are sent to Alice and Bob respectively). Observe that for any i , there are at most $g(n/2)^2$ matchings that vertices are partitioned the same way as M_i . (I.e., if we define $C_i = \{v \in V \mid \exists e \in A_i \text{ s.t. } v \in e\}$ then for any $i, |\{j \mid C_i = C_j\}| \leq g(n/2)^2$.) This is because $n/2$ vertices on each side of the partition can make $g(n/2)$ different matchings.

Therefore, the number of matchings such that the vertices are divided differently is at least

$$\frac{n!}{(n/2)!2^{n/2}} \left(\frac{(n/4)!2^{n/4}}{(n/2)!} \right)^2 = \binom{n}{n/2} / \binom{n/2}{n/4} \geq \binom{n/2}{n/4}$$

where the last inequality follows from the fact that $\binom{n}{n/2}$ is the number of $n/2$ -subsets of $\{1, 2, \dots, n\}$ and $\binom{n/2}{n/4}^2$ is the number of parts of these subsets.

In particular, if we let M_{i_1}, \dots, M_{i_t} , where $t = \binom{n/2}{n/4}$, be such matchings then for any $j \neq k, (A_{i_j}, B_{i_k})$ is not a perfect matching. Now, let $t' = \log t$. Note that $t' = \Omega(n)$. Consider the problem $\text{EQ}_{t'}$ where Alice and Bob each gets a t' -bit vector x and y , respectively. They have to output 1 if $x = y$ and 0 otherwise. By [20, example 1.21], $D(\text{EQ}_{t'}) \geq t' + 1 = \Omega(n)$.

Now we reduce $\text{EQ}_{t'}$ to our problem: Map x to M_{i_x} and y to M_{i_y} . Now, $x = y$ if and only if (M_{i_x}, M_{i_y}) is a perfect matching. This shows that the magic-partition communication complexity of the matching problem is $\Omega(n)$. \square

Note the the above lower bound is asymptotically tight since there is an obvious protocol where Alice sends Bob all vertices she has (using $O(n)$ bits of communication).

5 Graph Connectivity

Graph connectivity is perhaps the most basic property that one would like to check. However, even graph connectivity does not admit space-efficient algorithms in traditional streaming models. There is an $\Omega(n)$ lower bound for randomized algorithms. To contrast this, we show that allowing the algorithm the additional power of requesting the input in a specific order allows for very efficient, $O((\log n)^2)$ space algorithms for testing connectivity.

Problem: We consider a function where the input is a set of edges and $f(e_1, e_2, \dots, e_m) = 1$ if and only if G is connected. As usual, let n be the number of vertices of G . As before, we assume that vertices are labeled $1, 2, 3, \dots, n$.

We will prove the following theorem.

Theorem 5. *Graph connectivity can be solved by a randomized algorithm in $O((\log n)^2)$ space best-order stream checking.*

Proof. We use the following lemma which is easy to prove.

Lemma 2. *For any graph G of n edges, G is connected if and only if there exists a vertex v and trees T_1, T_2, \dots, T_q such that for all i ,*

- *there exists a unique vertex $u_i \in V(T_i)$ such that $uv \in E(T_i)$, and*
- *$|V(T_i)| \leq 2n/3$ for all i .*

Suppose G is connected, i.e., G is a tree. Let v and T_1, T_2, \dots, T_q be as in the lemma. Define the order of G to be

$$\text{Order}(G) = vu_1, \text{Order}(T'_1), vu_2, \text{Order}(T'_2), \dots, vu_q, \text{Order}(T'_q)$$

where $T'_i = T_i \setminus \{vu_i\}$. Note that T'_i is a connected tree and so we present edges of T'_i recursively.

Now, when edges are presented in this order, the checker can check if the graph is connected as follows. First, the checker reads vu_1 . He checks if T'_1 is connected by running the algorithm recursively. Note that he stops checking T'_1 once he sees vu_2 . Next, he repeats with vu_2 and T'_2 and so on.

The space needed is for vu_i and for checking T'_i . I.e., $\text{space}(|G|) = \text{space}(\max_i |T_i|) + O(\log n)$. That is, $\text{space}(n) \leq \text{space}(2n/3) + O(\log n)$. This gives the claimed space bound.

Note that the checker has to make sure every vertex appears in the graph. He does so by applying result in Section 3 once to each vertex v used as a root (as in above) and all leaf nodes of the tree. Also note that if G is not connected then such ordering cannot be made and the algorithm above will detect this fact. \square

6 Further Results

The previous sections give us a flavor of the results that can be obtained in this model. We describe a few more and mention the intuition behind the protocol (without giving details, due to space constraints).

6.1 Bipartite k -Regular Graph

The point is that a k -regular bipartite graph can be decomposed into k disjoint sets of perfect matchings. So the adversary can do this and present each of the perfect matchings one after the other. Now our previously described algorithm can be used to verify each perfect matching. In addition, a fairly simple algorithm can take care of verifying that we indeed receive k different sets (and to also know when one perfect matching ends and the new one is presented).

6.2 Hamiltonian Cycle

It can be shown that $\text{HAMILTONIAN-CYCLE} \in \text{RSTREAM}(1, \log n)$. The intuition is for the protocol to request the hamiltonian cycle first (everything else is ignored). The checker then checks if the first n edges presented indeed form a cycle; this requires two main facts. First that every two consecutive edges share a vertex, and the n -th edge shares a specific vertex with the first. This is easy. The second key step is to check that these edges indeed span all n vertices (and not go through same vertex more than once). This can be done by using the set distinctness approach.

6.3 Non-bipartiteness

Non-bipartiteness of graphs can again be checked in our model by requesting the adversary to present an odd length cycle. Verifying that this is indeed a cycle and that it is of odd length is again done in a manner very similar to verifying hamiltonian cycle.

We do not have an algorithm to verify general *bipartiteness* of graphs and leave it as an open question.

7 Conclusions

This paper describes a new model of stream checking, that lies at the intersection of extremely well-studied and foundational fields of computer science. Specifically, the model connects several settings relating to proof checking, communication complexity, and streaming algorithms. The motivation for this paper, however, draws from recent growth in data sizes and the advent of powerful cloud computing architectures and services. The question we ask is, can verification of certain properties (on any input) be accompanied with a streaming proof of the fact? The checker should be able to verify that the prover is not cheating.

We show that if the checker (or algorithm in the streaming algorithms setting) is given the power of choosing a specific rule for the prover to send the input, then many problems can be solved much more efficiently in this model than in the previous models.

While non-obvious, our algorithms and proofs are fairly simple. However, the nice aspect is that it uses several interesting techniques and areas such as fingerprinting, and covert channels. Fingerprinting is used in a crucial way to randomly test for distinctness of a set of elements presented as a stream. The protocol between the prover and check also allows for covert communication (which gives covert channels a positive spin as opposed to previous studies in security and cryptography). While the prover is only allowed to send the input, re-ordered, the prover is able to encode extra bits of information with the special ordering requested by the checker. The difficulty in most of our proof techniques is in how the checker or algorithm verifies that the prover or adversary is sending the input order as requested.

We have given $O(\text{polylog } n)$ space algorithms for problems that previously, in the streaming model, had no sub-linear algorithms. There are still a lot of problems in graph theory that remain to be investigated. A nice direction is to consider testing for graph minors, which could in turn yield efficient methods for testing planarity and other properties that exclude specific minors. We have some work in progress in this direction. It is also interesting to see whether all graph problems in the complexity class P can be solved in our model with $O(\text{polylog } n)$ space. Apart from the study of our specific model, we believe that the results and ideas presented in this paper could lead to improved algorithms in previously studied settings as well as yield new insights to the complexity of the problems.

References

1. Amazon elastic compute cloud (amazon ec2)
2. Aggarwal, G., Datar, M., Rajagopalan, S., Ruhl, M.: On the streaming model augmented with a sorting primitive. In: Annual IEEE Symposium on Foundations of Computer Science, pp. 540–549 (2004)
3. Alon, N., Matias, Y., Szegedy, M.: The space complexity of approximating the frequency moments. *J. Comput. Syst. Sci.* 58(1), 137–147 (1999)
4. 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)
5. Arora, S., Safra, S.: Probabilistic checking of proofs: A new characterization of np . *J. ACM* 45(1), 70–122 (1998)
6. Chakrabarti, A., Cormode, G., McGregor, A.: Robust lower bounds for communication and stream computation. In: STOC, pp. 641–650 (2008)
7. Chakrabarti, A., Jayram, T.S., Pătraşcu, M.: Tight lower bounds for selection in randomly ordered streams. In: Proc. 19th ACM/SIAM Symposium on Discrete Algorithms (SODA), pp. 720–729 (2008)
8. Demetrescu, C., Finocchi, I., Ribichini, A.: Trading off space for passes in graph streaming problems. In: SODA 2006: Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm, pp. 714–723. ACM, New York (2006)

9. Dinur, I.: The pcp theorem by gap amplification. *J. ACM* 54(3), 12 (2007)
10. Feigenbaum, J., Kannan, S., Strauss, M., Viswanathan, M.: Testing and spot-checking of data streams (extended abstract). In: *SODA 2000: Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms*, Philadelphia, PA, USA, pp. 165–174. Society for Industrial and Applied Mathematics (2000)
11. Feigenbaum, J., Kannan, S., McGregor, A., Suri, S., Zhang, J.: Graph distances in the streaming model: the value of space. In: *SODA 2005: Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, Philadelphia, PA, USA, pp. 745–754. Society for Industrial and Applied Mathematics (2005)
12. Feigenbaum, J., Kannan, S., McGregor, A., Suri, S., Zhang, J.: On graph problems in a semi-streaming model. *Theor. Comput. Sci.* 348(2), 207–216 (2005)
13. Goldreich, O.: Property testing in massive graphs, pp. 123–147 (2002)
14. Gopalan, P., Radhakrishnan, J.: Finding duplicates in a data stream. In: *SODA 2009: Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, Philadelphia, PA, USA. Society for Industrial and Applied Mathematics (to appear, 2009)
15. Guha, S., McGregor, A.: Approximate quantiles and the order of the stream. In: *PODS 2006: Proceedings of the twenty-fifth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pp. 273–279. ACM, New York (2006)
16. Guha, S., McGregor, A.: Lower bounds for quantile estimation in random-order and multi-pass streaming. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) *ICALP 2007*. LNCS, vol. 4596, pp. 704–715. Springer, Heidelberg (2007)
17. Hajnal, A., Maass, W., Turán, G.: On the communication complexity of graph properties. In: *STOC*, pp. 186–191 (1988)
18. Håstad, J., Wigderson, A.: The randomized communication complexity of set disjointness. *Theory of Computing* 3(1), 211–219 (2007)
19. Henzinger, M.R., Raghavan, P., Rajagopalan, S.: Computing on data streams, pp. 107–118 (1999)
20. Kushilevitz, E., Nisan, N.: *Communication complexity*. Cambridge University Press, New York (1997)
21. Lam, T.W., Ruzzo, W.L.: Results on communication complexity classes. *J. Comput. Syst. Sci.* 44(2), 324–342 (1992)
22. Lipton, R.J.: *Fingerprinting sets*. Cs-tr-212-89. Princeton University (1989)
23. Lipton, R.J.: Efficient checking of computations. In: *STACS*, pp. 207–215 (1990)
24. Munro, J.I., Paterson, M.: Selection and sorting with limited storage. In: *FOCS*, pp. 253–258 (1978)
25. Papadimitriou, C.H., Sipser, M.: *Communication complexity*. *J. Comput. Syst. Sci.* 28(2), 260–269 (1984)
26. Ruhl, J.M.: *Efficient algorithms for new computational models*. Ph.D thesis, Supervisor-David R. Karger (2003)
27. Yao, A.C.-C.: Some complexity questions related to distributive computing (preliminary report). In: *STOC*, pp. 209–213 (1979)

Behavioral and Logical Equivalence of Stochastic Kripke Models in General Measurable Spaces

Ernst-Erich Doberkat

Chair for Software Technology
Technische Universität Dortmund
doberkat@acm.org

Abstract. We show that logical and behavioral equivalence for stochastic Kripke models over general measurable spaces are the same. Usually, this requires some topological assumptions and includes bisimilarity; the results here indicate that a measurable structure on the state space of the Kripke model suffices. In contrast to a paper by Danos et al. we focus on the measurable structure of the factor space induced by the logic. This technique worked well in the analytic case, and it is shown to work here as well. The main contribution of the paper is methodological, since it provides a uniform framework for general measurable as well as more specialized analytic spaces.

1 Introduction

Bisimilarity and behavioral equivalence of Markov transition systems permit comparing the expressive power of these systems through a span resp. a cospan of morphisms. Given a modal logic, logical equivalence indicates that each state in one system accepts exactly the same formulas as a suitable state in the other system, in this way comparing the expressivity of the system with respect to the logic. It is well known that these ways of relating the behavior of Markov transition systems are equivalent even for very simple negation free Hennessy-Milner logics, provided the base spaces are Polish or analytic¹, see [2,3], topological assumptions and constructions entering substantially through the argumentation. So at least a partial answer to the question of relating these fundamental notions seemed to be hopeless except for the case of countable state spaces [8], but in a paper [1] that was as important as it was surprising it could be established that behavioral and logical equivalence are the same without any topological assumptions, hence for general measurable spaces. The proof technique developed in that paper is essentially *coalgebraic* in nature, emphasizing event bisimulations and introducing cocongruences. In the meantime, the analytic point of view was further developed into more general coalgebraic logics [5], refining the approach through congruences and factorizations, which, in contrast to the techniques above, might be referred to as the *universal algebra* point of view.

¹ A *Polish space* is a topological space the topology of which has a countable base, and which can be generated by a complete metric, an *analytic space* is the continuous image of a Polish space.

The present paper shows that the latter collection of proof techniques may be adapted for obtaining the essential results of [1]. It utilizes techniques which have been developed from general considerations on stochastic relations and Kripke models [5], but in contrast to [1] it uses only elementary measure-theoretic tools. Consequently, we have two fairly different approaches at our disposal. They might be interesting enough to compare. Work in progress [12] indicates that the present approach proposed here generalizes to a setting which utilizes the power of predicate liftings [9] for general measure polynomial endofunctors on the category of measurable spaces, opening up a whole new spectrum of functors for which the results above apply.

The paper is organized as follows: we briefly collect some preliminaries in Section 2, develop a strategy in Section 3, discuss the equivalence relation induced by the logic under consideration in Section 4 and prove the main technical result in Section 5. Section 6 wraps it all up by indicating some avenues to go from here. Lack of space prevents giving complete proofs. They may be found in [6].

2 Preliminaries

A measurable space (M, \mathcal{M}) consists of a set M with a σ -algebra \mathcal{M} , which is an algebra of subsets of M that is closed under countable unions (hence countable intersections and countable disjoint unions). If \mathcal{M}_0 is a family of subsets of M , then $\sigma(\mathcal{M}_0)$ is the smallest σ -algebra on M which contains \mathcal{M}_0 . Take for example as a generator \mathcal{T} all open sets in a topological space X , then $\sigma(\mathcal{T}) =: \mathcal{B}(X)$ is the σ -algebra of *Borel sets*. The elements of a σ -algebra are usually referred to as measurable sets.

An important tool is the π - λ -Theorem which makes it sometimes simpler to identify the σ -algebra generated from some family of sets.

Theorem 1 (π - λ -Theorem). *Let \mathcal{P} be a family of subsets of a set X that is closed under finite intersections. Then $\sigma(\mathcal{P})$ is the smallest set of subsets containing \mathcal{P} which is closed under complements and countable disjoint unions.*

The basic probabilistic structure is described by stochastic relations (a. k. a. sub Markov kernels).

Definition 1. $K : (M, \mathcal{M}) \rightsquigarrow (N, \mathcal{N})$ is a *stochastic relation* on the measurable spaces (M, \mathcal{M}) and (N, \mathcal{N}) iff $K(m)$ is a subprobability measure on (N, \mathcal{N}) for each $m \in M$ such that the map $m \mapsto K(m)(Q)$ is \mathcal{M} -measurable for each measurable set $Q \in \mathcal{N}$.

Denote by $\mathfrak{S}(M, \mathcal{M})$ the set of all subprobabilities on the measurable space (M, \mathcal{M}) ; this space is rendered a measurable space by endowing it with the weak- $*$ - σ -algebra \mathcal{M}^∇ ; this is the initial σ -algebra with respect to the evaluation maps $\mu \mapsto \mu(A)$ for $A \in \mathcal{M}$. Then $K : (M, \mathcal{M}) \rightsquigarrow (N, \mathcal{N})$ is a stochastic relation iff K induces a map $M \rightarrow \mathfrak{S}(N, \mathcal{N})$ which is \mathcal{M} - \mathcal{N}^∇ -measurable. \mathfrak{S} acts as an endofunctor on the category of measurable spaces, where the measurable map $f : N \rightarrow M$ is mapped to $\mathfrak{S}(f)(\mu)(B) := \mu(f^{-1}[B])$. The endofunctor is the functorial part of the Giry monad which may be perceived as the probabilistic analogue to the power set monad [74].

A *stochastic Kripke model* $\mathcal{K} = ((S, \mathcal{S}), (k_a)_{a \in \text{Act}})$ has a measurable state space (S, \mathcal{S}) and for each action $a \in \text{Act}$ a transition subprobability $k_a : (S, \mathcal{S}) \rightsquigarrow (S, \mathcal{S})$.

Given action a in state s , $k_a(s)(D)$ is the probability that the next state is a member of the measurable set $D \subseteq S$. Since $k_a(s)(S) < 1$ is admissible, the Kripke model may be in no successor state at all: mass may vanish.

This very simple Lemma which is occasionally of use.

Lemma 1. *Let $f : M \rightarrow N$ be a map; call $A \subseteq M$ f -invariant iff $a \in A$ and $f(a) = f(a')$ together imply $a' \in A$. Then $f[A_1 \cap A_2] = f[A_1] \cap f[A_2]$, whenever A_1 and A_2 are f -invariant, also $f^{-1}[f[A]] = A$ for f -invariant $A \subseteq M$. \square*

Equivalence classes are invariant w.r.t. the factor map. Let ρ be an equivalence relation on a measurable space (M, \mathcal{M}) , then the factor space M/ρ is usually endowed with the σ -algebra \mathcal{M}/ρ which is final with respect to the factor map $\eta_\rho : x \mapsto [x]_\rho$.

3 Defining the Logic \mathcal{L} and Developing a Strategy

We address in the present paper the question of behavioral and logical equivalence without topological assumptions; thus we will work in general measurable spaces, and we will show that both notions are equivalent as well. We did not include bisimilarity in this discussion for the following reason. If we want to show that two behavioral equivalent models are bisimilar, we are requested to construct a mediating model, and it is currently not clear how this can be done without constructing a semi-pullback which in turn requires at least analytic base spaces; the ground breaking paper [2] deserves to be mentioned, the results assume an analytic space, and yield a universally measurable solution, which is strictly weaker than analyticity (the latter is provided in [3]). The constructions in the present paper will be carried out for the negation free Hennessy-Milner logic $\mathcal{L} = \mathcal{L}(\text{Act}, [0, 1])$ the formulas of which are given through the grammar

$$\top \mid \phi_1 \wedge \phi_2 \mid \langle a \rangle_r \phi$$

Here $a \in \text{Act}$ is an action, and the threshold r is a real number from the unit interval. We do not require here the set Act of possible actions to be countable, and we do not restrict ourselves to rationals as the value for thresholds. Because the logic is so simple we can keep the interpreting Kripke models simple, too. Given a Kripke model $\mathcal{K} = ((S, \mathcal{S}), (k_a)_{a \in \text{Act}})$, we define for the state $s \in S$ validity of $\langle a \rangle_r \phi$ in s through $\mathcal{K}, s \models \langle a \rangle_r \phi$ iff $k_a(s)(\llbracket \phi \rrbracket_{\mathcal{K}}) \geq r$, $\llbracket \phi \rrbracket_{\mathcal{K}}$ being the set of states in which ϕ holds. Validity for the other formulas is defined as usual. Define the *theory* $Th_{\mathcal{K}}(s)$ for s as the set of all formulas which hold in state $s \in S$.

Let us briefly review the strategy for the analytic case, where we have a countable number of actions, and where the thresholds are taken from the rational numbers. Given a Kripke model \mathcal{K} over an analytic space, the logic defines an equivalence relation $r_{\mathcal{K}}$ which is countably generated. This equivalence relation is used for factoring, and because of its countable generation, we obtain as a factor space an analytic space again. This σ -algebra has a fairly rich structure. In particular the *amalgamated sum* which is introduced in Section 5 leads to an analytic space which in turn can be made the state space of a Kripke model through standard constructions. The logic influences these discussions only through the corresponding equivalence relations, witnessed by the observation that the general criterion for bisimilarity from [3] enters the discussion, this

criterion being formulated in terms of general countably generated equivalence relations (which are of considerable importance in other areas of Mathematics as well, see [10] for a comprehensive overview).

We show that it is possible to construct a cospan of Kripke models without relying on the machinery of Polish and analytic spaces. So we start from general measurable spaces, investigating the equivalence relation which is induced by the logic on the state space. Since analyticity is not available, we will not be able to observe the convenient interplay of the measurable structures induced by the logic on the state space and on the factor space, specifically we are no longer able to observe that the $r_{\mathcal{K}}$ -invariant measurable sets are exactly the inverse images of the elements of the final σ -algebra with respect to the factor map $\eta_{r_{\mathcal{K}}}$. Thus we need to construct explicitly an σ -algebra on the factor space which is closely adapted to the logic, and to derive a Kripke model from it which plays the rôle of the factor model. Similarly, the amalgamation of the equivalences on the individual models needs to be investigated more closely, the interesting properties no longer being made automatically available through analyticity. The leading idea, however, is still based on the observation that the equivalence classes induced by the logic on the state spaces of logically equivalent Kripke models are in a one-to-one correspondence.

4 The Equivalence Relation Induced by \mathcal{L}

Fix a Kripke model $\mathcal{K} = ((S, \mathcal{S}), (k_a)_{a \in \text{Act}})$ with a measurable state space (S, \mathcal{S}) , thus $k_a : (S, \mathcal{S}) \rightsquigarrow (S, \mathcal{S})$ is a stochastic relation for each action a . A *morphism* $f : \mathcal{K} \rightarrow ((T, \mathcal{T}), (\ell_a)_{a \in \text{Act}})$ between Kripke models is an \mathcal{S} - \mathcal{T} -measurable map $f : S \rightarrow T$ such that $\forall a \in \text{Act} : \ell_a \circ f = \mathfrak{S}(f) \circ k_a$ holds. This translates for each label $a \in \text{Act}$ into $\ell_a(f(s))(Q) = k_a(s)(f^{-1}[Q])$ for all measurable sets $Q \in \mathcal{T}$, $s \in S$. Note that we do not require f to be onto. Just for the record:

Lemma 2. *Let $f : \mathcal{K} \rightarrow \mathcal{L}$ be a morphism and ϕ a formula in \mathcal{L} , then $\mathcal{K}, s \models \phi \Leftrightarrow \mathcal{L}, f(s) \models \phi$ holds for each state s of \mathcal{K} , and $f^{-1}[\llbracket \phi \rrbracket_{\mathcal{L}}] = \llbracket \phi \rrbracket_{\mathcal{K}}$. \square*

The equivalence relation $r_{\mathcal{K}}$ induced by \mathcal{L} on S is defined as usual through

$$s \ r_{\mathcal{K}} \ s' \text{ iff } \forall \phi : \mathcal{K}, s \models \phi \Leftrightarrow \mathcal{K}, s' \models \phi.$$

It is noted that taking all real numbers from the unit interval as threshold values is not particularly mandatory: we can do with less. Define $\mathcal{L}^{\dagger} := \mathcal{L}(\text{Act}, \mathbb{Q} \cap [0, 1])$ as the fragment of $\mathcal{L}(\text{Act}, [0, 1])$ in which only rational numbers are admitted as indices to the modal operators, and define $r_{\mathcal{K}}^{\dagger}$ exactly as $r_{\mathcal{K}}$ but restricted to formulas from \mathcal{L}^{\dagger} . Using σ -additivity of the transition laws k_a and the fact that the rationals are dense shows that $r_{\mathcal{K}}^{\dagger} = r_{\mathcal{K}}$, see [5] Lemma 3.4.1]. In any case, two states are considered equivalent iff they cannot be separated by a formula.

Put $\mathcal{E}_{\mathcal{K}} := \{\llbracket \phi \rrbracket_{\mathcal{K}} \mid \phi \text{ is a formula}\}$, the set of all extensions of formulas, and define as σ -algebra $\mathcal{S}_{r_{\mathcal{K}}}^{\#}$ on the factor space $S/r_{\mathcal{K}}$ as the smallest σ -algebra which contains all the sets the inverse image of which are in $\mathcal{E}_{\mathcal{K}}$:

$$\mathcal{S}_{r_{\mathcal{K}}}^{\#} := \sigma(\{A \subseteq S/r_{\mathcal{K}} \mid \eta_{r_{\mathcal{K}}}^{-1}[A] \in \mathcal{E}_{\mathcal{K}}\}).$$

We analyze this construction, and then we enter a discussion of behavioral and logical equivalence.

Lemma 3. *The set $\mathcal{A} := \{\eta_{r_{\mathcal{K}}} \llbracket \phi \rrbracket_{\mathcal{K}} \mid \phi \text{ is a formula}\}$ is a generator of $\mathcal{S}_{r_{\mathcal{K}}}^{\sharp}$ which is closed under finite intersections.*

Proof. Each extension $\llbracket \phi \rrbracket_{\mathcal{K}}$ is $\eta_{r_{\mathcal{K}}}$ -invariant, and the logic \mathcal{L} is closed under conjunction, so \mathcal{A} is closed under finite intersections. Because $\llbracket \phi \rrbracket_{\mathcal{K}}$ is $\eta_{r_{\mathcal{K}}}$ -invariant, we have $\llbracket \phi \rrbracket_{\mathcal{K}} = \eta_{r_{\mathcal{K}}}^{-1} [\eta_{r_{\mathcal{K}}} \llbracket \phi \rrbracket_{\mathcal{K}}]$, thus $\sigma(\mathcal{A}) \subseteq \mathcal{S}_{r_{\mathcal{K}}}^{\sharp}$. On the other hand, if $\eta_{r_{\mathcal{K}}}^{-1} [A] \in \mathcal{E}_{\mathcal{K}}$ for some $A \subseteq S/r_{\mathcal{K}}$, then there exists a formula ϕ such that $\llbracket \phi \rrbracket_{\mathcal{K}} = \eta_{r_{\mathcal{K}}}^{-1} [A]$, hence $A = \eta_{r_{\mathcal{K}}} \llbracket \phi \rrbracket_{\mathcal{K}}$, since $\eta_{r_{\mathcal{K}}}$ is onto. \square

This has some immediate consequences.

Corollary 1. *The factor map $\eta_{r_{\mathcal{K}}} : S \rightarrow S/r_{\mathcal{K}}$ is $\mathcal{S}\text{-}\mathcal{S}_{r_{\mathcal{K}}}^{\sharp}$ -measurable. Let $S/r_{\mathcal{K}}$ be the final σ -algebra with respect to $\eta_{r_{\mathcal{K}}} : S \rightarrow S/r_{\mathcal{K}}$ and \mathcal{S} . Then $\mathcal{S}_{r_{\mathcal{K}}}^{\sharp} \subseteq \mathcal{S}/r_{\mathcal{K}}$. \square*

Comparing the construction for the general case with the one for analytic spaces, we see that in the latter case we can determine the crucial σ -algebra $\mathcal{S}_{r_{\mathcal{K}}}^{\sharp}$ through the Borel sets on the factor space. This is the distinguishing and important feature of the analytic case: The factor σ -algebra is then exactly the σ -algebra of the Borel sets on the factor space (or, very loosely speaking, factoring and forming the Borel σ -algebra commutes for the analytic case). Denote for a closer analysis by $\Sigma(\mathcal{S}, r_{\mathcal{K}})$ the σ -algebra of $\eta_{r_{\mathcal{K}}}$ -invariant measurable sets.

Proposition 1. *Assume that the set Act of actions is countable, and that*

$$\sigma(\{\llbracket \phi \rrbracket_{\mathcal{K}} \mid \phi \text{ is a formula}\}) = \Sigma(\mathcal{S}, r_{\mathcal{K}}).$$

Then $\mathcal{S}_{r_{\mathcal{K}}}^{\sharp} = \mathcal{S}/r_{\mathcal{K}}$. \square

This yields immediately

Corollary 2. *$\mathcal{S}_{r_{\mathcal{K}}}^{\sharp} = \mathcal{B}(S/r_{\mathcal{K}})$, if S is an analytic space with $\mathcal{S} = \mathcal{B}(S)$, and if the set Act of actions is countable. \square*

Now consider the Kripke model \mathcal{K} . Let $a \in \text{Act}$ be an action; the factor relation $k_{a, r_{\mathcal{K}}}$ on $(S, \mathcal{S}_{r_{\mathcal{K}}}^{\sharp})$ is defined through

$$k_{a, r_{\mathcal{K}}}([s]_{r_{\mathcal{K}}})(A) := k_a(s)(\eta_{r_{\mathcal{K}}}^{-1} [A]),$$

whenever $A \in \mathcal{S}_{r_{\mathcal{K}}}^{\sharp}$. This definition is possible since $\eta_{r_{\mathcal{K}}}^{-1} [A] \in \mathcal{S}$ for $A \in \mathcal{S}_{r_{\mathcal{K}}}^{\sharp}$ (Corollary 1). It determines in fact a stochastic relation.

Proposition 2. *$k_{a, r_{\mathcal{K}}} : (S/r_{\mathcal{K}}, \mathcal{S}_{r_{\mathcal{K}}}^{\sharp}) \rightsquigarrow (S/r_{\mathcal{K}}, \mathcal{S}_{r_{\mathcal{K}}}^{\sharp})$ is a stochastic relation.*

Proof. It is clear that $k_{a, r_{\mathcal{K}}}([s]_{r_{\mathcal{K}}})$ is a subprobability on $\mathcal{S}_{r_{\mathcal{K}}}^{\sharp}$ for each $s \in S$. The $\mathcal{S}_{r_{\mathcal{K}}}^{\sharp}$ -measurability of $v \mapsto k_{a, r_{\mathcal{K}}}(v)(A)$ is established through the π - λ -Theorem 1 and Lemma 3, capitalizing on the construction of $\mathcal{S}_{r_{\mathcal{K}}}^{\sharp}$. \square

The factor map defines a morphism between the stochastic relations k_a and $k_{a, r_{\mathcal{K}}}$, as we will see now. In fact, define the Kripke model

$$\mathcal{K}_{\mathcal{E}} := ((S/r_{\mathcal{K}}, \mathcal{S}_{r_{\mathcal{K}}}^{\sharp}), (k_{a, r_{\mathcal{K}}})_{a \in \text{Act}}),$$

then we make this observation which will be useful for the investigations of behavioral and logical equivalence.

Corollary 3. $\eta_{r_{\mathcal{K}}} : \mathcal{K} \rightarrow \mathcal{K}_{\mathcal{L}}$ is a morphism. \square

Now let $\mathcal{L} = ((T, \mathcal{T}), (\ell_a)_{a \in \text{Act}})$ be another Kripke model. Denote the equivalence relation defined by the logic \mathcal{L} on T by $r_{\mathcal{L}}$. All constructions with the σ -algebra \mathcal{S} and the equivalence relation $r_{\mathcal{K}}$ are carried out with \mathcal{T} and $r_{\mathcal{L}}$, so we may construct a σ -algebra $\mathcal{T}_{r_{\mathcal{L}}}^{\sharp}$ on $T/r_{\mathcal{L}}$, and we obtain a new Kripke model $\mathcal{L}_{\mathcal{L}} = ((T/r_{\mathcal{L}}, \mathcal{T}_{r_{\mathcal{L}}}^{\sharp}), (\ell_{a, r_{\mathcal{K}}})_{a \in \text{Act}})$ together with the morphism $\eta_{r_{\mathcal{L}}} : \mathcal{L} \rightarrow \mathcal{L}_{\mathcal{L}}$.

Models \mathcal{K} and \mathcal{L} are *behaviorally equivalent* iff there exists a cospan

$$\mathcal{K} \xrightarrow{f} \mathcal{M} \xleftarrow{g} \mathcal{L}$$

of surjective morphisms for a suitable model \mathcal{M} . Define the relation $\mathfrak{R} := \{\langle s, t \rangle \in S \times T \mid Th_{\mathcal{K}}(s) = Th_{\mathcal{L}}(t)\}$. Consequently, $s \mathfrak{R} t$ iff s and t satisfy exactly the same formulas. In particular, we know that in this case $k_a(s)(\llbracket \phi \rrbracket_{\mathcal{K}}) = \ell_a(t)(\llbracket \phi \rrbracket_{\mathcal{L}})$ holds for all formulas ϕ . Put $\mathfrak{R}_0 := \{\langle [s]_{r_{\mathcal{K}}}, [t]_{r_{\mathcal{L}}} \rangle \mid \langle s, t \rangle \in \mathfrak{R}\}$. \mathcal{K} and \mathcal{L} are said to be *logically equivalent* iff the relation \mathfrak{R} is both right and left total. This is but a simple reformulation of the usual definition of logical equivalence which states that \mathcal{K} and \mathcal{L} are logically equivalent iff given a state s in \mathcal{K} there exists a state t in \mathcal{L} such that $Th_{\mathcal{K}}(s) = Th_{\mathcal{L}}(t)$, and vice versa. Assume for the rest of this Section that the Kripke models \mathcal{K} and \mathcal{L} are logically equivalent.

Lemma 4. \mathfrak{R}_0 is the graph of a bijection $\tau : S/r_{\mathcal{K}} \rightarrow T/r_{\mathcal{L}}$; τ is $\mathcal{S}_{r_{\mathcal{K}}}^{\sharp} - \mathcal{T}_{r_{\mathcal{L}}}^{\sharp}$ -measurable. Similarly, \mathfrak{R}_0^{-1} is the graph of a bijection $\theta : S/r_{\mathcal{K}} \rightarrow T/r_{\mathcal{L}}$ which is $\mathcal{T}_{r_{\mathcal{L}}}^{\sharp} - \mathcal{S}_{r_{\mathcal{K}}}^{\sharp}$ -measurable. τ and θ are inverse to each other. \square

If \mathfrak{R} would not be a left total relation, the map τ would only be partially defined; if \mathfrak{R} would not be right total, τ would not be surjective. Consequently, this construction works only with logically equivalent Kripke models. But actually τ and θ are even richer in structure.

Lemma 5. Define τ and θ as in Lemma 4. Then $\tau : \mathcal{K}_{\mathcal{L}} \rightarrow \mathcal{L}_{\mathcal{L}}$ and $\theta : \mathcal{L}_{\mathcal{L}} \rightarrow \mathcal{K}_{\mathcal{L}}$ are morphisms. \square

5 Logical vs. Behavioral Equivalence

The construction will go through a sequence of technical steps which will be sketched now. Form the sum $S + T$ of the state space with injections $i_S : S \rightarrow S + T$ and $i_T : T \rightarrow S + T$, resp. Let $r_{\mathcal{K}} \diamond r_{\mathcal{L}}$ be the amalgamation of $r_{\mathcal{K}}$ and $r_{\mathcal{L}}$ [3]. This equivalence relation is defined for $v, v' \in S + T$ as follows

$$v r_{\mathcal{K}} \diamond r_{\mathcal{L}} v' \iff \begin{cases} s r_{\mathcal{K}} s', & v = i_S(s), v' = i_S(s'), \\ t r_{\mathcal{L}} t', & v = i_T(t), v' = i_T(t'), \\ \langle s, t \rangle \in \mathfrak{R}, & v = i_S(s), v' = i_T(t), \\ \langle t, s \rangle \in \mathfrak{R}^{-1}, & v = i_T(t), v' = i_S(s) \end{cases}$$

The construction entails $(*) [i_S(s)]_{r_{\mathcal{K}} \diamond r_{\mathcal{L}}} = i_S[[s]_{r_{\mathcal{K}}}] \cup i_T[\tau([s]_{r_{\mathcal{K}}})]$, similar for $[i_T(t)]_{r_{\mathcal{K}} \diamond r_{\mathcal{L}}}$. Thus the equivalence class of an element of $S + T$ has both a non void component from S and from T , and these components are linked through τ and θ , resp. Now define maps $I_S : S/r_{\mathcal{K}} \rightarrow (S + T)/r_{\mathcal{K}} \diamond r_{\mathcal{L}}$ and $I_T : T/r_{\mathcal{L}} \rightarrow (S + T)/r_{\mathcal{K}} \diamond r_{\mathcal{L}}$ through $I_S([s]_{r_{\mathcal{K}}}) := [i_S(s)]_{r_{\mathcal{K}} \diamond r_{\mathcal{L}}}$, $I_T([t]_{r_{\mathcal{L}}}) := [i_T(t)]_{r_{\mathcal{K}} \diamond r_{\mathcal{L}}}$, so we assign to each class in the participating spaces the class of its representative in the sum; it is clear from the characterization in $(*)$ that both maps are well defined.

Lemma 6. $I_S[\eta_{r_{\mathcal{K}}}[[\phi]_{\mathcal{K}}]] = I_T[\eta_{r_{\mathcal{L}}}[[\phi]_{\mathcal{L}}]]$ holds for each formula ϕ , and the set $\{I_S[\eta_{r_{\mathcal{K}}}[[\phi]_{\mathcal{K}}]] \mid \phi \text{ is a formula}\}$ is closed under finite intersections. \square

Define the σ -algebra \mathcal{W} on $(S + T)/r_{\mathcal{K}} \diamond r_{\mathcal{L}}$ as

$$\mathcal{W} := \sigma(\{I_S[\eta_{r_{\mathcal{K}}}[[\phi]_{\mathcal{K}}]] \mid \phi \text{ is a formula}\}),$$

then $\mathcal{W} = \sigma(\{I_T[\eta_{r_{\mathcal{L}}}[[\phi]_{\mathcal{L}}]] \mid \phi \text{ is a formula}\})$ follows from Lemma 6 and the maps I_S and I_T turn out to be measurable.

Lemma 7. $I_S : S \rightarrow (S + T)/r_{\mathcal{K}} \diamond r_{\mathcal{L}}$ is $\mathcal{S}_{r_{\mathcal{K}}}^{\#}$ - \mathcal{W} -measurable, and $I_T : T \rightarrow (S + T)/r_{\mathcal{K}} \diamond r_{\mathcal{L}}$ is $\mathcal{T}_{r_{\mathcal{L}}}^{\#}$ - \mathcal{W} -measurable. \square

This construction yields a measurable space $((S + T)/r_{\mathcal{K}} \diamond r_{\mathcal{L}}, \mathcal{W})$. It will serve as the state space for a Kripke model for which we will construct the transition law now. Before we do that, we need to make sure that the transition laws $k_{a,r_{\mathcal{K}}}$ and $\ell_{a,r_{\mathcal{L}}}$ coincide on certain crucial sets.

Lemma 8. Assume that $s \mathfrak{R} t$, then $k_{a,r_{\mathcal{K}}}([s]_{r_{\mathcal{K}}})(I_S^{-1}[C]) = \ell_{a,r_{\mathcal{L}}}([t]_{r_{\mathcal{L}}})(I_T^{-1}[C])$ for all $C \in \mathcal{W}$. \square

Now we are poised to define the transition law on the compound factor space. Put $m_a([i_S(s)]_{r_{\mathcal{K}} \diamond r_{\mathcal{L}}})(C) := k_{a,r_{\mathcal{K}}}([s]_{r_{\mathcal{K}}})(I_S^{-1}[C])$ for $C \in \mathcal{W}$. By Lemma 8 $m_a([i_T(t)]_{r_{\mathcal{K}} \diamond r_{\mathcal{L}}})(C) = \ell_{a,r_{\mathcal{L}}}([t]_{r_{\mathcal{L}}})(I_T^{-1}[C])$, provided $[i_S(s)]_{r_{\mathcal{K}} \diamond r_{\mathcal{L}}} = [i_T(t)]_{r_{\mathcal{K}} \diamond r_{\mathcal{L}}}$. This is so because the latter condition is equivalent to $\langle [s]_{r_{\mathcal{K}}}, [t]_{r_{\mathcal{L}}} \rangle \in \mathfrak{R}_0$ which in turn is equivalent to $\langle s, t \rangle \in \mathfrak{R}$. Thus we obtain for each action $a \in \text{Act}$ a map $m_a : (S + T)/r_{\mathcal{K}} \diamond r_{\mathcal{L}} \rightarrow \mathfrak{G}((S + T)/r_{\mathcal{K}} \diamond r_{\mathcal{L}}, \mathcal{W})$, and we have to make sure that it defines a Kripke model, i.e., that it is a stochastic relation. This means that we have to establish measurability.

Lemma 9. $m_a : ((S + T)/r_{\mathcal{K}} \diamond r_{\mathcal{L}}, \mathcal{W}) \rightsquigarrow ((S + T)/r_{\mathcal{K}} \diamond r_{\mathcal{L}}, \mathcal{W})$ is a stochastic relation for each action $a \in \text{Act}$. \square

Thus we may use m_a as the transition law for a Kripke model.

Corollary 4. $\mathcal{K}_{\mathcal{L}} \xrightarrow{I_S} \mathcal{M} \xleftarrow{I_T} \mathcal{L}_{\mathcal{L}}$ is a cospan of morphisms. \square

This is the main result.

Proposition 3. Logical and behavioral equivalence are the same for Kripke models over general measurable spaces for the negation free Hennessy-Milner logic \mathcal{L} .

Proof. Since morphisms preserve and reflect validity, behaviorally equivalent Kripke models are logically equivalent. Let $\mathcal{K} = ((S, \mathcal{S}), (k_a)_{a \in \text{Act}})$ and $\mathcal{L} = ((T, \mathcal{T}), (\ell_a)_{a \in \text{Act}})$ be the Kripke models under consideration. Construct the factor space $(S + T)/r_{\mathcal{K}} \diamond r_{\mathcal{L}}$ for the amalgamation $r_{\mathcal{K}} \diamond r_{\mathcal{L}}$ of the equivalence relations $r_{\mathcal{K}}$ and $r_{\mathcal{L}}$ which are constructed from logic \mathcal{L} over S resp. T together with the maps $I_S : S/r_{\mathcal{K}} \rightarrow (S + T)/r_{\mathcal{K}} \diamond r_{\mathcal{L}}$ and $I_T : T/r_{\mathcal{L}} \rightarrow (S + T)/r_{\mathcal{K}} \diamond r_{\mathcal{L}}$. Construct the σ -algebra \mathcal{W} from these data, and define the stochastic relation m_a as in Lemma 9. Put

$$\mathcal{M} := \left(((S + T)/r_{\mathcal{K}} \diamond r_{\mathcal{L}}, \mathcal{W}), (m_a)_{a \in \text{Act}} \right).$$

Then this diagram gives the desired cospan of morphisms.

$$\mathcal{K} \xrightarrow{\eta_{r_{\mathcal{K}}}} \mathcal{K}_{\mathcal{L}} \xrightarrow{I_S} \mathcal{M} \xleftarrow{I_T} \mathcal{L}_{\mathcal{L}} \xleftarrow{\eta_{r_{\mathcal{L}}}} \mathcal{L}$$

The factor maps $\eta_{r_{\mathcal{K}}}$ and $\eta_{r_{\mathcal{L}}}$ are morphisms by Corollary 3. I_S and I_T are morphisms by Corollary 4. \square

6 Conclusion and Further Work

We show that behavioral and logical equivalence can be shown to be equivalent on general measurable spaces without assuming any Polish or analytic structure. In contrast to [1] we use the machinery which has been developed in a series of papers [3, 5] for the analytic case and adapt it to the situation at hand. The core ingredient turned out to be the direct investigation of the measurable structure induced by the logic. As we have argued elsewhere, the kernel logic may be extended by additional operators to gain expressivity, without, however, making it necessary to change the algebraic core of the arguments.

This gives some hope for extending the present work to the case that we replace modal operators by predicate liftings for some measure polynomial endofunctor on the category of all measurable spaces. It is important to see how far we may be carried without topological assumptions, which appears to be one of the urgent messages from [1]; the elegant and far-reaching results obtained in [1] witness another step into this direction. Technically, the results presented here indicate somewhat surprisingly that it is promising to concentrate on the measurable structure of the spaces involved rather than on the transition probabilities proper.

Acknowledgements. The author wants to thank Vincent Danos for discussing with him some finer points from [1], and Christoph Schubert for some technical discussions.

References

1. Danos, V., Desharnais, J., Lavolette, F., Panangaden, P.: Bisimulation and cocongruence for probabilistic systems. *Information and Computation* 204(4), 503–523 (2006)
2. Desharnais, J., Edalat, A., Panangaden, P.: Bisimulation of labelled Markov-processes. *Information and Computation* 179(2), 163–193 (2002)

3. Doberkat, E.-E.: Stochastic relations: congruences, bisimulations and the Hennessy-Milner theorem. *SIAM J. Computing* 35(3), 590–626 (2006)
4. Doberkat, E.-E.: Kleisli morphisms and randomized congruences for the Giry monad. *J. Pure Appl. Alg.* 211, 638–664 (2007)
5. Doberkat, E.-E.: Stochastic coalgebraic logic. Technical Report 174, Chair for Software Technology, TU Dortmund, November 2008. Springer, Heidelberg (2009)
6. Doberkat, E.-E.: Behavioral and logical equivalence of stochastic Kripke models in general measurable spaces. Technical Report 176, Chair for Software Technology, Technische Universität Dortmund (January 2009)
7. Giry, M.: A categorical approach to probability theory. In: *Categorical Aspects of Topology and Analysis*. *Lect. Notes Math.*, vol. 915, pp. 68–85. Springer, Berlin (1981)
8. Larsen, K.G., Skou, A.: Bisimulation through probabilistic testing. *Information and Computation* 94, 1–28 (1991)
9. Pattinson, D.: Expressive logics for coalgebras via terminal sequence induction. *Notre Dame J. Formal Logic* 45(1), 19–33 (2004)
10. Schneider, S., Thomas, S.: Countable Borel equivalence relations, Athens, OH, November 2007. *Lecture Notes, Appalachian Set Theory Seminar* (2007)
11. Schubert, C.: Final coalgebras for measure-polynomial functors. Technical Report 175, Chair for Software Technology, Technische Universität Dortmund (December 2008)
12. Schubert, C.: Coalgebraic logic over measurable spaces: Behavioral and logical equivalence. Technical Report 177, Chair for Software Technology, Technische Universität Dortmund (February 2009)

Influence of Tree Topology Restrictions on the Complexity of Haplotyping with Missing Data

Michael Elberfeld, Ilka Schnoor, and Till Tantau

Institut für Theoretische Informatik
Universität zu Lübeck, 23538 Lübeck, Germany
{elberfeld,schnoor,tantau}@tcs.uni-luebeck.de

Abstract. Haplotyping, also known as haplotype phase prediction, is the problem of predicting likely haplotypes from genotype data. One fast haplotyping method is based on an evolutionary model where a perfect phylogenetic tree is sought that explains the observed data. Unfortunately, when data entries are missing as is often the case in laboratory data, the resulting incomplete perfect phylogeny haplotyping problem IPPH is NP-complete and no theoretical results are known concerning its approximability, fixed-parameter tractability, or exact algorithms for it. Even radically simplified versions, such as the restriction to phylogenetic trees consisting of just two directed paths from a given root, are still NP-complete; but here a fixed-parameter algorithm is known. We show that such drastic and ad hoc simplifications are not necessary to make IPPH fixed-parameter tractable: We present the first theoretical analysis of an algorithm, which we develop in the course of the paper, that works for arbitrary instances of IPPH. On the negative side we show that restricting the topology of perfect phylogenies does not always reduce the computational complexity: while the incomplete directed perfect phylogeny problem is well-known to be solvable in polynomial time, we show that the same problem restricted to path topologies is NP-complete.

1 Introduction

Haplotype phase prediction is an important preprocessing step in genomic association studies. In these studies two groups of people are considered, where one group has a certain disease or medical condition while the other has not, and one tries to find correlations between group membership and the genomic data of the individuals in the groups. The genomic data typically consists of information about which bases are present in an individual's DNA at so-called SNP sites (single nucleotide polymorphism sites). While the DNA sequences of different individuals are mostly identical, at SNP sites there may be variations. Low-priced methods for large-scale inference of genomic data can read out, separately for each SNP site, the bases present. At a site two bases can be present because we inherit one chromosome from our father and one from our mother. Since the bases at different sites are determined independently, we have no information on which chromosome a base belongs to. For *homozygous sites*, where the same

base is present on both chromosomes, this is not a problem, but for *heterozygous sites* this information, called the *phase* of a SNP site, is needed for accurate correlations. The idea behind *haplotype phase prediction* or just *haplotyping* is to computationally predict likely phases based on the laboratory data (which misses this information). For an individual, the genomic input data without phase information is called the *genotype* while the two predicted chromosomes are called *haplotypes*.

From a mathematical point of view haplotypes can be conveniently coded as strings over the alphabet $\{0, 1\}$, where for a given site 0 stands for one of the bases that can be observed in practice, while 1 encodes a second base that can also be observed. (The case that three bases are observed happens so seldom that it can be ignored.) A genotype g is, conceptually, a sequence of sets that arises from two haplotypes h_1 and h_2 as follows: The i th set in the sequence g is $\{h_1[i], h_2[i]\}$. However, it is customary to encode the set $\{0\}$ as 0, to encode $\{1\}$ as 1, and $\{0, 1\}$ as 2, so that a genotype is actually a string over the alphabet $\{0, 1, 2\}$. For example, the two haplotypes 0110 and 0101 give rise to (we also say *explain*) the genotype 0122; and so do 0100 and 0111.

Since different haplotype pairs can explain the same genotype and any single haplotype is equally likely *a priori*, haplotyping is not possible if only a single genotype is given. However, if a set of genotypes from a larger group of different individuals is given, certain sets of explaining haplotypes are more likely than others. For instance, a small set of explaining haplotypes is more likely than a large set since haplotypes mutate only rarely. It is customary to formalize sets of genotypes as matrices (each row is a genotype) and also sets of explaining haplotypes (each row contains a haplotype and rows $2i-1$ and $2i$ of the haplotype matrix explain exactly the genotype in row i of the genotype matrix).

One important method of haplotyping is based on the *perfect phylogeny approach* proposed by Gusfield [13]. The idea is to seek a haplotype matrix that explains the genotype matrix and whose rows (which are the haplotypes) can be arranged in a *perfect phylogenetic tree*. This means the following: A haplotype matrix B admits a *perfect phylogeny* if there exists a tree (an undirected, connected, acyclic graph) T_B such that:

1. Each column of B labels exactly one edge of T_B and each edge is labeled by at least one column.
2. Each row of B labels exactly one node of T_B .
3. For every two rows h_1 and h_2 of B and every column i , we have $h_1[i] \neq h_2[i]$ if, and only if, i lies on the path from h_1 to h_2 in T_B .

The intuition behind these properties is as follows. The nodes of the tree T_B correspond to haplotypes. The edges between the nodes correspond to mutation events: When we move from one node to another node along a single edge, the label(s) of the edge name exactly those columns in which the node labels differ. This means that when we remove an edge labeled by a column c , the two resulting components have the property that all nodes in one component have a 0 in column c and all nodes in the other component have a 1 in that column.

When a haplotype matrix B admits a perfect phylogeny T_B and, at the same time, explains a genotype matrix A , we also say that A admits a perfect phylogeny. In this case, it is useful to define a tree T_A as follows: Its topology is the same as T_B 's and so are the node labels, but the edges are labeled by the columns of A instead of the columns of B . We call T_A a perfect phylogeny for A . The formal *perfect phylogeny haplotyping* problem (PPH) is the set of all genotype matrices that admit a perfect phylogeny. Gusfield [13] showed that PPH is solvable in polynomial time.

In practice, laboratory data is never perfect and some entries may be missing in the input genotype matrices. In this case, the input matrices may contain ?-entries in addition to the 0-, 1-, and 2-entries. The objective is then to replace the missing entries by normal entries such that the resulting matrix is in PPH. This problem is known as IPPH, where the I stands for *incomplete* (in the following, the prefix I denotes that the input matrix may contain ?-entries that need to be filled up). Unfortunately, IPPH is NP-complete [20].

In order to tackle the problem, one can try to exploit properties of typical input data that may make the problem easier to solve. The first simplification is the notion of *directedness*. In real data, some genotype is typically completely known and is completely homozygous, which means that one haplotype of the sought haplotype matrix is already known. Since the roles of 0-entries and 1-entries can be exchanged individually for each column, we may assume that the known haplotype is the all-0-haplotype. This problem variant is called “directed” because the position of the all-0-haplotype in the phylogenetic tree singles out a root node and gives an orientation to the tree. The resulting problem is called IDPPH, with D standing for “directed.” It is still NP-complete [15].

A second, rather radical simplification (which is nevertheless often backed by the data) was proposed by Gramm, Nierhoff, Sharan and the third author [11]: In addition to being directed, we required that the (undirected) phylogenetic tree must form a simple path. The resulting problem was called incomplete directed perfect *path* phylogeny haplotyping. It is *still* NP-complete, but in [11] we presented a fixed-parameter algorithm for it, where the parameter is the maximum number of ?-entries per column.

A third radical simplification is to forbid heterozygous sites. This is the same as getting already phased haplotypes as input and the question is just whether they can be arranged in a perfect phylogeny. It turns out that IPP (note the missing H, since no haplotyping needs to be done) is still NP-complete [20], but the directed variant IDPP is solvable in polynomial time [2,17].

In the present paper we further the study of the computational complexity of IPPH and the above variants. We are especially interested in the following question: How do restrictions on the tree topology influence the complexity of the problem? In other words, what is the complexity of $\text{IPPH}_{\text{leafs} \leq l}$, where the explaining phylogenetic tree may have at most l leaves. As stated above, the best known result is that $\text{IDPPH}_{\text{leafs} \leq 2}$ is NP-complete, but lies in FPT.

Our Contributions. Our first main result, Theorem 2.1 presented in Section 2, is the following hardness result: $\text{IDPP}_{\text{leafs} \leq l}$ is NP-complete for every $l \geq 2$.

In sharp contrast, $\text{IDPP} \in \text{P}$. As detailed in the section on related work, past experience has indicated that restricting the topology of perfect phylogenies makes haplotyping problems easier, not harder. Theorem 2.1 shows that IDPP is a notable exception. Naturally, there are other examples of such exceptions: Finding a spanning tree for a graph is easy, finding a spanning path is hard.

Our second main contribution, presented in Section 3, is an algorithm for solving IPPH that allows a rigorous runtime analysis. In detail, we present an algorithm that on input of a number l and an incomplete $n \times m$ genotype matrix A with at most k many ?-entries per column correctly outputs: either “ $A \notin \text{IPPH}_{\text{leafs} \leq l}$ ” or a completion of A and a perfect phylogeny for this completion with at most l leaves. This algorithm runs in time $f(k, l)n^2m^{O(l)}$ and, hence, allows us to make formal statements about the fixed-parameter tractability of IPPH. First, IPPH lies in the class XP for the parameter pair (k, l) . Second and more importantly, for each fixed $l \geq 2$ the problem $\text{IPPH}_{\text{leafs} \leq l}$ is fixed-parameter tractable with respect to the number of unknown entries per column, see Theorem 3.1. This settles the central problem that we had to leave open in [11], namely whether the fixed-parameter tractability of $\text{IDPPH}_{\text{leafs} \leq 2}$ extends to the undirected case and to larger numbers of leaves. On both accounts, we answer this question affirmatively.

Due to lack of space, we omit all proofs in the present paper, they can be found in our technical report [6].

Related Work. Haplotyping methods can be split into two groups: Statistical, see [9] for a literature starting point, and combinatorial. There are two main combinatorial methods: Maximum parsimony haplotyping [4,12] and the more recent perfect phylogeny approach that was introduced by Gusfield [13] and later explored by numerous authors [1,3,5,8,16,18].

The idea of considering restricted tree topologies to speed up haplotyping is due to Gramm et al. [11] and was recently also investigated in the context of finding block partitions [10]. A different approach to deal with the NP-completeness of IPPH is due to Halperin and Karp [14]. They present a polynomial-time algorithm for IPPH that works for special instances satisfying the so-called “rich data hypothesis.” A heuristic for IPPH was proposed in [19], but no guarantees can be made concerning its runtime.

For complete data, numerous results on the complexity of PPH and its variants are known. Gusfield showed that the problem can be solved in polynomial time [13], further papers first presented simpler polynomial-time algorithms [1,8] and later even linear-time algorithms [3,5,16,18]. In [7] we have shown that PPH is hard for logarithmic space and lies in NC^2 .

The influence of restricting the tree topology on the complexity of haplotyping problems has, prior to the present paper, always been benign: In [11] it is shown that $\text{IDPPH}_{\text{leafs} \leq 2}$ has a fixed-parameter algorithm, which is not known to be the case for IPPH. In [10] it is shown that partitioning a complete genotype matrix into a minimal number of column sets such that each set admits a perfect *path* phylogeny is equivalent, in complexity theoretic terms, to finding maximal matchings; while the same problem for arbitrary perfect phylogenies is NP-hard

and even very hard to approximate. Finally, in [7] it is shown that $\text{DPPH}_{\text{leaves} \leq 2}$ lies in AC^0 , while DPPH is L-hard.

2 Hardness Result

Theorem 2.1. *IDPP_{leaves ≤ l} is NP-complete for every $l \geq 2$.*

Since $\text{IDPP} \in \text{P}$, the above theorem is a first example of a perfect *path* phylogeny problem being harder than the corresponding problem for general perfect phylogenies. Our proof is based on a reduction from the NP-complete problem MONOTONE NAE3SAT and is similar to the reduction presented in [11], which starts, however, from NAE3SAT. By starting our reduction from a (conceptually) simpler problem we are able to prove a stronger result than the one in [11].

The problem $\text{IDPP}_{\text{leaves} \leq l}$ reduces to $\text{IDPPH}_{\text{leaves} \leq l}$ via the identity mapping. It also reduces to $\text{IPP}_{\text{leaves} \leq l}$ and $\text{IPPH}_{\text{leaves} \leq l}$ by appending an all-0-haplotype. Indeed, all previously known NP-completeness results for variants of IPPH follow from Theorem 2.1, except for the NP-completeness of IPP.

3 Fixed Parameter Tractability Result

Theorem 3.1. *For each $l \geq 2$, the problem $\text{IPPH}_{\text{leaves} \leq l}$ is fixed-parameter tractable with respect to the maximal number of ?-entries per column.*

The theorem, proved in the present section, generalizes the result from [11] by Gramm, Nierhoff, Sharan and the third author that $\text{IDPPH}_{\text{leaves} \leq 2}$ is fixed-parameter tractable. The algorithm from [11] relies strongly on Gusfield’s characterization [13]: Given a genotype matrix A , a directed perfect phylogeny T for it, and any genotype g of A , the 1-entries of g label a path from the root to some node v of T and the 2-entries of g label a path containing v . Most algorithms for PPH and its variants from the literature exploit this property as follows: They first reduce the problem to the directed version DPPH and then build the phylogeny by placing columns with many 1-entries and 2-entries near to the root and columns with fewer such entries far from the root. The notions “should be placed near to the root” and “should be placed far from the root” can be quantified using Gusfield’s notion of leaf count [13].

When the data is incomplete and question marks are present, no reduction from the undirected to the directed case is known. (Indeed, IPP is NP-complete while $\text{IDPP} \in \text{P}$.) To solve the undirected problem variant $\text{IPPH}_{\text{leaves} \leq l}$ we need a replacement for the notion of leaf counts and a new characterization of genotype matrices admitting undirected perfect phylogenies. We present such a replacement, which we call the *light component size*, and also a new characterization in terms of the new notion of *mutation trees*. The characterization allows us to construct phylogenies in a stepwise fashion from the “outside” (columns far removed from the root, having a small light component size) to the “inside” (columns near to the root, having a large light component size). In each step,

we only need to remember the inner part of the partial phylogeny constructed so far, making a dynamic program feasible.

In the following two sections, we first introduce the new notion and characterization and then show how they can be used in an algorithm.

3.1 A Characterization of Undirected Perfect Phylogeny Haplotyping

A major tool in the development of efficient algorithms for the DPPH problem has been the *leaf count* of a column, which is twice the number of 1-entries plus the number of 2-entries. The name “leaf count” stems from the following observation: In a perfect phylogeny for a genotype matrix A , the number of haplotypes (which are typically attached to leaves) below the edge labeled by a column equals exactly its leaf count. This means that if two columns occur on a path from the root (recall that this is always the all-0-haplotype in a directed perfect phylogeny) to a leaf, the column with a greater leaf count is located nearer to the root.

For undirected perfect phylogenies the leaf count is no longer meaningful since there is no distinguished root node that is known in advance. To tackle this problem, we introduce the new notion of *light component sizes*. For a column c and $x \in \{0, 1, 2\}$ let $n_x(c)$ denote the number of x -entries in c .

Definition 3.2. *For a column c of a genotype matrix A its light component size and heavy component size are defined as follows:*

$$\begin{aligned} \text{lcs}(c) &:= n_2(c) + 2 \cdot \min\{n_0(c), n_1(c)\}, \\ \text{hcs}(c) &:= n_2(c) + 2 \cdot \max\{n_0(c), n_1(c)\}. \end{aligned}$$

The key observation is that when we remove an edge labeled by a column c from a perfect phylogeny T_A , then two components result and the number of node labels in one of these components will be $\text{lcs}(c)$ and we call the component the *light component*, the other will contain $\text{hcs}(c)$ labels and we call it the *heavy component*. (In case $\text{lcs}(c) = \text{hcs}(c)$, the choice is arbitrary.) The properties of T_A assure that all node labels in one component have a 0 in column c (and a 1 in the other component). Each of the $n_0(c)$ many 0-entries of c contributes two nodes labels to this component, while each 2-entry contributes one node label, which means that the number of node labels in this component is either $\text{lcs}(c)$ or $\text{hcs}(c)$. The argument is similar for the other component and for 1-entries.

We have just seen that the value in column c of all node labels in the light component is the same. Let us call this value the *light component value* $\text{lcv}(c)$. Clearly, $\text{lcv}(c) = 0$ if $n_0(c) < n_1(c)$ and $\text{lcv}(c) = 1$ if $n_0(c) > n_1(c)$. For $n_0(c) = n_1(c)$ we remarked earlier that the light component can be chosen arbitrarily; at this point we implicitly fix that choice by setting $\text{lcv}(c) = 1$. Symmetrically, we set the *heavy component value* $\text{hcv}(c) = 1 - \text{lcv}(c)$.

Our next aim is to define a quasi-ordering \preceq on columns that tells us something about how columns can possibly be arranged in a perfect phylogeny. Suppose that for two columns c and d we know that the light component of d is a

superset of the light component of c . Consider a node label h and suppose the value of h at the position of column c happens to be the light component value of c . Then we know that h must lie in the light component of c and, thus, also in the light component of d , which in turn means that at position d in h we must have the light component value of d . Phrased more succinctly: for every $i \in \{1, \dots, n\}$ we have $c[i] = \text{lc}(c) \implies d[i] = \text{lc}(d)$ and, by a similar argument, also $d[i] = \text{hc}(d) \implies c[i] = \text{hc}(c)$. Let us write $c \preceq d$ whenever these two implications hold for every i . Then $c \preceq d$ is a necessary, but not a sufficient condition for c 's light component being contained in d 's light component.

The ordering \preceq tells us something about containment of light components. We can similarly say something about columns whose light components are disjoint: then for every $i \in \{1, \dots, n\}$ we have $c[i] = \text{lc}(c) \implies d[i] = \text{hc}(d)$ and $d[i] = \text{lc}(d) \implies c[i] = \text{hc}(c)$. We write $c \perp d$ whenever these two implications hold for every i .

Our algorithm is only concerned with building a tree whose edges are labeled with columns of the input genotype matrix; the nodes of the tree are not labeled and the rows of the explaining haplotype matrix B are irrelevant to the algorithm. Since edge labels correspond to mutation events, we call the tree that is constructed by the algorithm a *mutation tree*.

Definition 3.3. *Let A be a genotype matrix. A mutation tree T for A is an undirected tree whose edges are bijectively labeled by A 's columns and which has a distinguished root node r such that the following conditions hold:*

1. Ordering condition: *For every path originating at the root with edge labels c_1, c_2, \dots, c_k we have $c_1 \succeq c_2 \succeq \dots \succeq c_k$.*
2. Compatibility condition: *For every two columns c and d that are incident to a common node v and that do not lie on the path from r to v we have $c \perp d$.*
3. Two-path condition: *For every three columns $c, d,$ and e that are incident to the same node, there is no $i \in \{1, \dots, n\}$ such that $c[i] = d[i] = e[i] = 2$.*

Lemma 3.4. *A genotype matrix A admits a perfect phylogeny with l leaves if, and only, if, there exists a mutation tree for A with l leaves.*

3.2 The Fixed-Parameter Algorithm

Our fixed-parameter algorithm for $\text{IPPH}_{\text{leaves} \leq l}$ works in two stages. The first stage is a preprocessing of the input matrix. After the preprocessing the maximal number of columns with the same light component size is bounded by a function in k and l . The basic idea is that if there are many different columns with the same light component size, they must lie on many different paths and, thus, at some point it is no longer possible to arrange them in a perfect phylogeny with only l leaves. For a detailed description of the preprocessing phase we refer to the technical report version of this paper [6].

The second stage is the main part of the algorithm. Here we test whether a preprocessed matrix A can be completed in such a way that it admits a mutation tree with at most l leaves. For the presentation of this step we need some additional terminology: Given a set of columns A or a matrix A , let $A|_{\text{lc}=i}, A|_{\text{lc} \leq i},$

and $A|_{\text{lcs}>i}$ denote the set of all columns c of A with $\text{lcs}(c) = i$, $\text{lcs}(c) \leq i$, and $\text{lcs}(c) > i$, respectively. A *completion* of a set of columns with ?-entries is obtained by replacing all ?-entries by 0-, 1-, or 2-entries. Note that a completion of a column with light component size i can have a light component size between i and $i + 2k$, where k is, as always, the number of ?-entries in the column. The *inner part* of a mutation tree T is the set $\text{inner}(T)$ of edges that are incident to the root of T .

The mutation tree construction algorithm works in iterations $i = 1, 2, \dots, n$. In iteration i it processes all completions of the set $A|_{\text{lcs}=i}$. The algorithm keeps track of what it has already found out about completions of $A|_{\text{lcs}<i}$ in previous iterations in what we call *tree records* (I, λ, U) . Such a record consists of an *inner part* I , a number $\lambda \in \{0, \dots, l\}$ of leaves, and a set of *unprocessed columns* U . The following definition formalizes the properties that tree records should have:

Definition 3.5. *Let A be an incomplete $n \times m$ genotype matrix and let $i \in \{1, \dots, n\}$. A tree record (I, λ, U) is good for A and i if there exists a completion S_i of $A|_{\text{lcs}\leq i}$ such that (a) $I = \text{inner}(T_i)$ for some mutation tree T_i for $S_i|_{\text{lcs}\leq i}$, (b) λ is the number of leafs of T_i , and (c) $U = S_i|_{\text{lcs}>i}$.*

The job of the algorithm is to compute in each iteration i the set R_i of all good tree records for A and i . Clearly, if R_n is nonempty after the last iteration, there exists a completion for A and a mutation tree with at most l leafs; and otherwise no such completion exists. Figure 1 shows the pseudo-code of the algorithm, Figure 2 shows an example of the algorithm in action.

The following two lemmas imply that the algorithm is correct and that it is a fixed-parameter algorithm for $\text{IPPH}_{\text{leafs}\leq l}$. Together, they prove Theorem 3.1.

Lemma 3.6. *After each iteration i of algorithm SOLVE-IPPH $_{\text{leafs}\leq l}$, the set R_i contains exactly the good tree records for A and i .*

Lemma 3.7. *Algorithm SOLVE-IPPH $_{\text{leafs}\leq l}$ runs in time $O(f(k)m^l n^2)$.*

Algorithm SOLVE-IPPH $_{\text{leafs}\leq l}$.

Input: An $n \times m$ genotype matrix A with at most k missing entries per column.

```

1   $A \leftarrow \text{PREPROCESS}(A)$ 
2   $R_0 \leftarrow \{(\emptyset, 0, \emptyset)\}$ 
3  for increasing light component sizes  $i \leftarrow 1, 2, \dots, n$  do
4       $R_i \leftarrow \emptyset$ 
5      for each completion  $C$  of  $A|_{\text{lcs}=i}$  do
6          for each tree record  $(I, \lambda, U) \in R_{i-1}$  do
7              for each mutation tree  $T$  for  $I \cup C|_{\text{lcs}=i} \cup U|_{\text{lcs}=i}$ 
                  with  $\lambda' - \lambda + |I|$  leafs for some  $\lambda' \leq l$ 
                  where all columns from  $I$  are incident to leafs of  $T$  do
8                   $R_i \leftarrow R_i \cup \{(\text{inner}(T), \lambda', C|_{\text{lcs}>i} \cup U|_{\text{lcs}>i})\}$ 
9  if  $R_n$  is nonempty then output " $A \in \text{IPPH}_{\text{leafs}\leq l}$ " else output " $A \notin \text{IPPH}_{\text{leafs}\leq l}$ "

```

Fig. 1. Our decision algorithm for $\text{IPPH}_{\text{leafs}\leq l}$

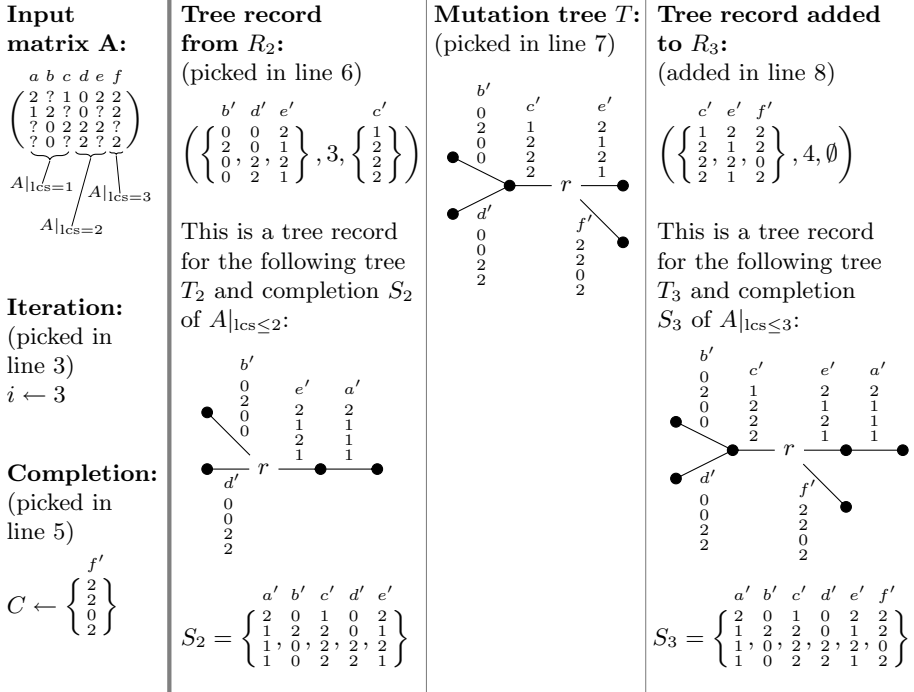


Fig. 2. Example of the third iteration of SOLVE-IPPH_{leaves ≤ l} for the indicated input matrix A. We depict a set of possible values for the loop variables for which a new tree record is added to R_3 .

4 Conclusion

Restrictions on the topologies of perfect phylogenies can greatly influence the complexity of IPPH and its variants. They can make the complexity jump from P to NP-complete (as for IDPP), but they also provide the first parameter for which a theoretical analysis is possible of an algorithm that works on arbitrary instances of the IPPH problem. Our new notions of mutation trees and light and heavy component sizes have turned out to be useful in the study of undirected perfect phylogenies; we suggest applying them to other problem versions as well.

The first main open problem is to improve the runtime of the fixed-parameter algorithm since the runtime is the range of $3^{O(k^l)}$, which is not feasible even for small values like $k = 5$ that are common in practice. The second main open question is whether IPPH is fixed-parameter tractable with respect to the maximal number of ?-entries per column.

References

1. Bafna, V., Gusfield, D., Lancia, G., Yooseph, S.: Haplotyping as perfect phylogeny: A direct approach. *J. Comput. Biol.* 10(3–4), 323–340 (2003)
2. Benham, C.J., Kannan, S., Paterson, M., Warnow, T.: Hen’s teeth and whale’s feet: Generalized characters and their compatibility. *J. Comput. Biol.* 2(4), 515–525 (1995)
3. Bonizzoni, P.: A linear-time algorithm for the perfect phylogeny haplotype problem. *Algorithmica* 48(3), 267–285 (2007)
4. Clark, A.G.: Inference of haplotypes from PCR-amplified samples of diploid populations. *J. of Mol. Biol. and Evol.* 7(2), 111–122 (1990)
5. Ding, Z., Filkov, V., Gusfield, D.: A linear-time algorithm for the perfect phylogeny haplotyping (PPH) problem. *J. Comput. Biol.* 13(2), 522–553 (2006)
6. Elberfeld, M., Schnoor, I., Tantau, T.: Influence of tree topology restrictions on the complexity of haplotyping with missing data. Tech. Rep. SIIM-TR-A-08-05, Universität zu Lübeck (2008)
7. Elberfeld, M., Tantau, T.: Computational complexity of perfect-phylogeny-related haplotyping problems. In: Ochmański, E., Tyszkiewicz, J. (eds.) MFCS 2008. LNCS, vol. 5162, pp. 299–310. Springer, Heidelberg (2008)
8. Eskin, E., Halperin, E., Karp, R.M.: Efficient reconstruction of haplotype structure via perfect phylogeny. *J. of Bioinform. and Comput. Biol.* 1(1), 1–20 (2003)
9. Excoffier, L., Slatkin, M.: Maximum-likelihood estimation of molecular haplotype frequencies in a diploid population. *Mol. Biol. and Evol.* 12(5), 921–927 (1995)
10. Gramm, J., Hartman, T., Nierhoff, T., Sharan, R., Tantau, T.: On the complexity of SNP block partitioning under the perfect phylogeny model. *Discrete Math.* (2008) (to appear), doi:010.1016/j.disc.2008.04.002
11. Gramm, J., Nierhoff, T., Sharan, R., Tantau, T.: Haplotyping with missing data via perfect path phylogenies. *Discrete and Appl. Math.* 155(6–7), 788–805 (2007)
12. Gusfield, D.: Inference of haplotypes from samples of diploid populations: Complexity and algorithms. *J. Comput. Biol.* 8(3), 305–323 (2001)
13. Gusfield, D.: Haplotyping as perfect phylogeny: Conceptual framework and efficient solutions. In: Proc. RECOMB 2002, pp. 166–175. ACM Press, New York (2002)
14. Halperin, E., Karp, R.M.: Perfect phylogeny and haplotype assignment. In: Proc. RECOMB 2002, pp. 10–19. ACM Press, New York (2004)
15. Kimmel, G., Shamir, R.: The incomplete perfect phylogeny haplotype problem. *J. Bioinform. and Comput. Biol.* 3(2), 359–384 (2005)
16. Liu, Y., Zhang, C.-Q.: A linear solution for haplotype perfect phylogeny problem. In: Proc. Int. Conf. Adv. in Bioinform. and Appl., pp. 173–184. World Scientific, Singapore (2005)
17. Pe’er, I., Pupko, T., Shamir, R., Sharan, R.: Incomplete directed perfect phylogeny. *SIAM J. Comput.* 33(3), 590–607 (2004)
18. Vijaya Satya, R., Mukherjee, A.: An optimal algorithm for perfect phylogeny haplotyping. *J. Comput. Biol.* 13(4), 897–928 (2006)
19. Vijaya Satya, R., Mukherjee, A.: The undirected incomplete perfect phylogeny problem. *IEEE/ACM T. Comput. Biol. and Bioinform.* 5(4), 618–629 (2008)
20. Steel, M.: The complexity of reconstructing trees from qualitative characters and subtrees. *J. Classif.* 9(1), 91–116 (1992)

Improved Deterministic Algorithms for Weighted Matching and Packing Problems*

Qilong Feng¹, Yang Liu², Songjian Lu², and Jianxin Wang^{1,**}

¹ School of Information Science and Engineering, Central South University, Changsha 410083, P.R. China

² Department of Computer Science and Engineering
Texas A&M University
College Station, Texas 77843-3112, USA
jxwang@mail.csu.edu.cn

Abstract. For the weighted r D-Matching problem, we present a deterministic parameterized algorithm with time complexity $O^*(4^{(r-1)k})$, improving the previous best upper bound $O^*(4^{rk})$. In particular, the algorithm can be applied to solve the unweighted 3D-Matching problem with time $O^*(16^k)$, improving the previous best result $O^*(21.26^k)$. For the weighted r -Set Packing problem, we present a deterministic parameterized algorithm with time complexity $O^*(2^{(2r-1)k})$, improving the previous best result $O^*(2^{2rk})$. The algorithm, when applied to the unweighted 3-Set Packing problem, has running time $O^*(32^k)$, improving the previous best result $O^*(43.62^k)$. Moreover, for the weighted r D-Matching and weighted r -Set Packing problems, we get a kernel of size $O(k^r)$.

1 Introduction

Matching and packing problems form an important class of NP-hard problems. In this paper, we are mainly focused on the weighted r D-Matching and weighted r -Set Packing problems, which are formally defined as follows.

Weighted r D-Matching: Given a collection $S \subseteq \mathcal{A} = A_1 \times A_2 \times \cdots \times A_r$ of r -tuples and an integer k , where A_1, A_2, \dots, A_r are pair-wise disjoint sets and each r -tuple in S has a weight value, find a subcollection S' of k r -tuples in S with the maximum weight sum such that no two r -tuples in S' have common elements, or report that no such subcollection exists.

* This work is supported by the National Natural Science Foundation of China (No. 60773111), the National Grand Fundamental Research 973 Program of China (No. 2008CB317107), the Excellent Youth Foundation of Hunan province of China (No. 06JJ10009), Program for New Century Excellent Talents in University (No. NCET-05-0683) and Program for Changjiang Scholars and Innovative Research Team in University (No. IRT0661).

** Corresponding author.

Weighted r -Set Packing: Given a collection S of r -sets (i.e., sets that contain exactly r elements) and an integer k , where each r -set in S has a weight value, find a subcollection S' of k r -sets in S with the maximum weight sum such that no two r -sets in S' have common elements, or report that no such subcollection exists.

For the weighted r D-Matching and weighted r -Set Packing problems, Downey and Fellows [1] gave the first deterministic parameterized algorithm of time complexity $O^*((rk)!(rk)^{3rk})$ [1]. Liu, Chen, and Wang [2] further reduced the above time complexity to $O^*(12.8^{rk})$. For the weighted 3D-Matching problem, Wang and Feng [3] gave a more efficient algorithm of time complexity $O^*(7.56^{3k})$. Chen et al. [4] presented a deterministic parameterized algorithm with time complexity $O^*(4^{rk})$, which is currently the best result for the weighted r D-Matching and weighted r -Set Packing problems.

We remark that there is also a very active research line on parameterized algorithms for the unweighted versions of the problems. Fellows et al. [5] gave a deterministic algorithm with running time $O^*(2^{5rk-4k} \binom{6(r-1)k+k}{rk})$. Chen et al. [6] and Jia et al. [7] presented deterministic algorithms for unweighted r D-Matching and unweighted r -Set Packing problems, with time complexity $O^*((r-1)^k ((r-1)k/e)^{k(r-2)})$. Koutis [8] gave an improved deterministic parameterized algorithm with running time $O^*(2^{O(rk)})$ for the unweighted r -Set Packing problem. For the case of $r = 3$, Liu et al. [9] presented a deterministic algorithm with time $O^*(21.26^k)$ for the unweighted 3D-Matching problem, which is currently the best result. For the unweighted 3-Set Packing problem, Liu et al. [9] gave a deterministic algorithm with time complexity $O^*(97.98^k)$, which was further reduced by Wang and Feng [10] to $O^*(43.62^k)$. This is currently the best result for the unweighted 3-Set Packing problem. It should be pointed out that all these algorithms [6,7,8,9,10] cannot be applied to solve the weighted versions. Very recently, Koutis [11] proposed a randomized algorithm of time complexity $O^*(8^k)$ for the unweighted 3D-Matching and unweighted 3-Set Packing problems. However, as remarked in [12], the algorithms in [11] are only workable for unweighted case and do not appear to extend to the weighted versions. Moreover, whether the algorithms can be extended to the r D-Matching and r -Set Packing problems for $r > 3$, and whether the randomized algorithm can be derandomized are still unknown.

In this paper, we derive deterministic parameterized algorithms of time $O^*(4^{(r-1)k})$ and $O^*(2^{(2r-1)k})$ respectively for the weighted r D-Matching and weighted r -Set Packing problems, improving the corresponding previous best result $O^*(4^{rk})$ and $O^*(2^{2rk})$. In fact, our algorithms can be applied to solve the the unweighted 3D-Matching and unweighted 3-Set Packing problems in time $O^*(16^k)$ and $O^*(32^k)$ respectively, which are much better than the corresponding previous best result $O^*(21.26^k)$ and $O^*(43.62^k)$. Moreover, a kernel of size $O(k^r)$ for the weighted r D-Matching and weighted r -Set Packing problems is presented.

We first give a brief review on the necessary background.

¹ Following the recent convention, we denote by $O^*(f(k))$ the bound $O(f(k)n^{O(1)})$.

Assume that n and k are integers such that $n > k$. Denote by Z_n the set $\{0, 1, \dots, n-1\}$. A *splitting function* over Z_n is a $\{0, 1\}$ function over Z_n . A subset W of Z_n is called a k -subset if it contains exactly k elements. Let (W_0, W_1) be a partition of the k -subset W , i.e., $W_0 \cup W_1 = W$ and $W_0 \cap W_1 = \emptyset$. We say that a splitting function f over Z_n *implements* the partition (W_0, W_1) if $f(x) = 0$ for all $x \in W_0$ and $f(y) = 1$ for all $y \in W_1$.

Definition 1. [13] *A set $\Psi_{n,k}$ of splitting functions over Z_n is an (n, k) -universal set if for every k -subset W of Z_n and any partition (W_1, W_2) of W , there is a splitting function f in $\Psi_{n,k}$ that implements (W_1, W_2) . The size of an (n, k) -universal set $\Psi_{n,k}$ is the number of splitting functions in $\Psi_{n,k}$.*

Lemma 1. [13] *There is an $O(n2^{k+12\log^2 k})$ time deterministic algorithm that constructs an (n, k) -universal set $\Psi_{n,k}$ of size bounded by $n2^{k+12\log^2 k+2}$.*

A function f on Z_n is *injective* from a subset W of Z_n if for any two different elements x and y in W , $f(x) \neq f(y)$.

By Bertrand's postulate, proved by Chebyshev in 1850 (see [15], Section 5.2), there is a prime number q such that $n \leq q < 2n$. Moreover, the smallest prime number q_0 between n and $2n$ can be constructed in time $O(n)$.

Lemma 2. [14] *Let n and k be integers, $n \geq k$, and let q_0 be the smallest prime number such that $n \leq q_0 < 2n$. For any k -subset W in Z_n , there is an integer z , $0 \leq z < q_0$, such that the function $g_{n,k,z}$ over Z_n , defined as $g_{n,k,z}(a) = (az \bmod q_0) \bmod k^2$, is injective from W .*

2 Improved Algorithm for Weighted r D-Matching

Let $S \subseteq A_1 \times \dots \times A_r$ be a collection of r -tuples. Denote by $\text{Val}^i(S)$ the set of all elements from A_i in S , $1 \leq i \leq r$, and let $\text{Val}(S) = \bigcup_{i=1}^r \text{Val}^i(S)$. Without loss of generality, we assume that $|\text{Val}^i(S)| = n$ for all i . A matching of size k is called a k -matching.

Let (S, k) be an instance of the weighted r D-Matching problem. Suppose that a k -matching of the maximum weight in S is M^* . Our improved algorithm for the weighted r D-Matching problem is based on the idea of Divide-and-Conquer. Let $\text{Val}^1(S) = \{a_1, \dots, a_n\}$. Then it is easy to see that there is an index h such that $\{a_1, \dots, a_h\}$ contains $k/2$ elements in $\text{Val}^1(M^*)$ and $\{a_{h+1}, \dots, a_n\}$ contains the other $k/2$ elements in $\text{Val}^1(M^*)$. In particular, this implies that the maximum weighted k -matching M^* can be partitioned into two $(k/2)$ -matchings M_1^* and M_2^* such that $\{a_1, \dots, a_h\}$ contains all elements in $\text{Val}^1(M_1^*)$ and $\{a_{h+1}, \dots, a_n\}$ contains all elements in $\text{Val}^1(M_2^*)$. The index h can be found by enumerating all indices from 1 to n . Therefore, finding the correct partition of the elements in $\text{Val}^1(S)$ takes only n rounds. This idea is implemented in Figure 1.

We only need to prove that when the collection S contains k -matchings, the algorithm **WRDM**(S, k) must return a k -matching of the maximum weight in S .

Theorem 1. *The algorithm WRDM in Figure 1 correctly solves the weighted rD -Matching problem in time $O^*(4^{(r-1)k})$.*

Proof. First note that the collection Q returned by the algorithm WRDM is initialized as the empty set \emptyset in step 4. Only when step 5.1 of the algorithm finds a k -matching Q_1 in S , step 5.2 of the algorithm replaces Q by a k -matching. Therefore, if the collection S has no k -matching, then the algorithm WRDM will always correctly return the empty set \emptyset .

Without loss of generality and by renaming the elements, we can assume that the set $\text{Val}^1(S)$ is the set Z_n , and that the set $\text{Val}(S) - \text{Val}^1(S)$ is the set $Z_{(r-1)n}$.

Algorithm WRDM (S, k)

Input: $S \subseteq \mathcal{A} = A_1 \times A_2 \times \dots \times A_r$ and an integer k

Output: a maximum weighted k -matching in S if such a matching exists

1. **for** $h = 1$ **to** k **do**
 - construct a $((r-1)k)^2, (r-1)h$ -universal set $\Psi_{((r-1)k)^2, (r-1)h}$;
2. let q_1 be the smallest prime number such that $n \leq q_1 < 2n$;
3. let q_2 be the smallest prime number such that $(r-1)n \leq q_2 < 2(r-1)n$;
4. $Q = \emptyset$;
5. **for** $0 \leq z_1 \leq q_1$ and $0 \leq z_2 \leq q_2$ **do**
 - 5.1 $Q_1 = \text{Matching-ext}(S, z_1, z_2, k)$;
 - 5.2 **if** $Q_1 \neq \emptyset$ and Q_1 is a k -matching with weight larger than that of Q **then** $Q = Q_1$;
6. return Q ;

Subroutine Matching-ext(S', z_1, z_2, h)

Input: a collection S' of r -tuples and an integer $h \leq k$, z_1 gives a pre-partition of $\text{Val}^1(S')$, and z_2 gives a pre-partition of $\text{Val}(S') - \text{Val}^1(S')$.

Output: an h -matching with maximum weight in S' if such a matching exists

1. **if** $h = 1$ **then** return the r -tuple with the maximum weight in S' ;
2. $Q' = \emptyset$;
3. **for** $i = 0$ **to** $k^2 - 1$ **do**
 - for** each splitting function f in $\Psi_{((r-1)k)^2, (r-1)h}$ **do**
 - 3.1 $V_0 = \{a \mid a \in \text{Val}^1(S') \text{ and } g_{n, h, z_1}(a) \leq i\}$;
 - 3.2 $V_1 = \{a \mid a \in \text{Val}^1(S') \text{ and } g_{n, h, z_1}(a) > i\}$;
 - 3.3 $W_0 = \{a \mid a \in \text{Val}(S') - \text{Val}^1(S') \text{ and } f(g_{(r-1)n, (r-1)k, z_2}(a)) = 0\}$;
 - 3.4 $W_1 = \{a \mid a \in \text{Val}(S') - \text{Val}^1(S') \text{ and } f(g_{(r-1)n, (r-1)k, z_2}(a)) = 1\}$;
 - 3.5 let S'_0 be the subcollection of r -tuples in S' that contain only elements in $V_0 \cup W_0$;
 - 3.6 let S'_1 be the subcollection of r -tuples in S' that contain only elements in $V_1 \cup W_1$;
 - 3.7 $Q'_0 = \text{Matching-ext}(S'_0, z_1, z_2, h/2)$;
 - 3.8 $Q'_1 = \text{Matching-ext}(S'_1, z_1, z_2, h/2)$;
 - 3.9 **if** $Q'_0 \neq \emptyset$, $Q'_1 \neq \emptyset$ and the sum of the weights of Q'_0 and Q'_1 is larger than that of Q' **then** $Q' = Q'_0 \cup Q'_1$;
4. return Q' ;

Fig. 1. The algorithm WRDM

In particular, for an h -matching M in S for any integer h , $\text{Val}^1(M)$ is a subset of h elements in Z_n and $\text{Val}(M) - \text{Val}^1(M)$ is a subset of $(r - 1)h$ elements in $Z_{(r-1)n}$.

We prove the following claim for the subroutine **Matching-ext**(S', z_1, z_2, h) by induction on the integer h .

Claim. Let M^* be an h -matching of the maximum weight in the collection S' , where $S' \subseteq S$ and $h \leq k$. If z_1 is an integer that makes the function g_{n,k,z_1} injective from $\text{Val}^1(M^*)$, and if z_2 is an integer that makes the function $g_{(r-1)n,(r-1)k,z_2}$ injective from $\text{Val}(M^*) - \text{Val}^1(M^*)$, then the subroutine **Matching-ext**(S', z_1, z_2, h) returns an h -matching of the maximum weight in the collection S' .

The Claim obviously holds true for the case $h = 1$ by step 1 of the subroutine (for any given z_1 and z_2). Now we consider the case $h > 1$. Recall that the function g_{n,k,z_1} is from Z_n to Z_{k^2} . Since the function g_{n,k,z_1} is injective from $\text{Val}^1(M^*)$, we can assume that g_{n,k,z_1} maps the h elements in $\text{Val}^1(M^*)$ to h different elements i_1, i_2, \dots, i_h in Z_{k^2} , where $0 \leq i_1 < i_2 < \dots < i_h \leq k^2 - 1$. Take the index $i_{h/2}$ and let M_0^* be the set of $h/2$ r -tuples in M^* such that $\forall a \in \text{Val}^1(M_0^*), g_{n,h,z_1}(a) \leq i_{h/2}$, and let M_1^* be the rest $h/2$ r -tuples in M^* such that $\forall b \in \text{Val}^1(M_1^*), g_{n,h,z_1}(b) > i_{h/2}$.

Now consider the set $\text{Val}(M^*) - \text{Val}^1(M^*)$. First of all, by our assumption, the function $g_{(r-1)n,(r-1)k,z_2}$, which is from $Z_{(r-1)n}$ to $Z_{((r-1)k)^2}$, is injective from $\text{Val}(M^*) - \text{Val}^1(M^*)$. Therefore, the function $g_{(r-1)n,(r-1)k,z_2}$ maps the set $\text{Val}(M^*) - \text{Val}^1(M^*)$ of $(r - 1)h$ elements to a set X of $(r - 1)h$ different elements in $Z_{((r-1)k)^2}$. In particular, the function $g_{(r-1)n,(r-1)k,z_2}$ maps the set $\text{Val}(M_0^*) - \text{Val}^1(M_0^*)$ of $(r - 1)h/2$ elements to a set X_0 of $(r - 1)h/2$ different elements in $Z_{((r-1)k)^2}$, and maps the set $\text{Val}(M_1^*) - \text{Val}^1(M_1^*)$ of $(r - 1)h/2$ elements to a set X_1 of $(r - 1)h/2$ different elements in $Z_{((r-1)k)^2}$. That is

$$g_{(r-1)n,(r-1)k,z} \text{ maps } \text{Val}(M_0^*) - \text{Val}^1(M_0^*) \text{ to } X_0, \tag{1}$$

$$g_{(r-1)n,(r-1)k,z} \text{ maps } \text{Val}(M_1^*) - \text{Val}^1(M_1^*) \text{ to } X_1. \tag{2}$$

Note that (X_0, X_1) makes a partition of X (i.e., $X_0 \cap X_1 = \emptyset$ and $X_0 \cup X_1 = X$). Since X is a subset of $(r - 1)h$ elements in $Z_{((r-1)k)^2}$, by the definition of the $((r - 1)k)^2, (r - 1)h$ -universal set $\Psi_{((r-1)k)^2,(r-1)h}$, there is a splitting function f_0 in $\Psi_{((r-1)k)^2,(r-1)h}$ that implements the partition (X_0, X_1) , that is $f_0(a) = 0$ for all $a \in X_0$, $f_0(b) = 1$ for all $b \in X_1$.

Now consider step 3 of the subroutine **Matching-ext**(S', z_1, z, h), when the integer $i = i_{h/2}$ is picked and the splitting function $f = f_0$ is picked. For these selections of the index $i = i_{h/2}$ and the function $f = f_0$, we can derive that $\text{Val}^1(M_0^*) \subseteq V_0$ and $\text{Val}^1(M_1^*) \subseteq V_1$. Moreover, based on the step 3.3-3.4 of the subroutine, we derive that $\text{Val}(M_0^*) - \text{Val}^1(M_0^*) \subseteq W_0$, and $\text{Val}(M_1^*) - \text{Val}^1(M_1^*) \subseteq W_1$.

It is easy to get that all elements in M_0^* are contained in $V_0 \cup W_0$, and all elements in M_1^* are contained in $V_1 \cup W_1$. By steps 3.5–3.6, the subcollection S'_0

contains the $(h/2)$ -matching M_0^* , and the subcollection S'_1 contains the $(h/2)$ -matching M_1^* . Note that M_0^* must be an $(h/2)$ -matching of the maximum weight in S'_0 – otherwise, a maximum weighted $(h/2)$ -matching in S'_0 plus the $(h/2)$ -matching M_1^* in S'_1 would form an h -matching whose weight is larger than that of M^* , contradicting the maximality of M^* . Since the function g_{n,k,z_1} is injective from $\text{Val}^1(M^*)$, the integer z_1 also makes the function g_{n,k,z_1} injective from $\text{Val}^1(M_0^*)$. Similarly, the integer z_2 makes the function $g_{(r-1)n,(r-1)k,z_2}$ injective from $\text{Val}(M_0^*) - \text{Val}^1(M_0^*)$. Therefore, by the induction hypothesis, the subroutine call **Matching-ext** $(S'_0, z_1, z_2, h/2)$ in step 3.7 will return an $(h/2)$ -matching M'_0 of the maximum weight in S'_0 , where the $(h/2)$ -matching M'_0 should have the same weight as that of M_0^* . Completely similar analysis shows that the subroutine call **Matching-ext** $(S'_1, z_1, z_2, h/2)$ in step 3.8 will return an $(h/2)$ -matching M'_1 of the maximum weight in S'_1 , where the $(h/2)$ -matching M'_1 should have the same weight as that of M_1^* . Since the collections S'_0 and S'_1 share no common elements, the union of M'_0 and M'_1 is an h -matching in S' whose weight is equal to that of the maximum weighted h -matching M^* , we conclude that after step 3.9 of the subroutine for the selections of the index $i = i_{h/2}$ and the splitting function $f = f_0$, the collection Q becomes an h -matching of the maximum weight in S' . In particular, when the subroutine **Matching-ext** (S', z_1, z_2, h) returns at step 4, it returns an h -matching of the maximum weight in S' . This completes the proof of the Claim.

For the algorithm **WRDM** (S, k) , suppose that the collection S has a k -matching \bar{M} of the maximum weight. Since the set $\text{Val}^1(\bar{M})$ is a subset of k elements in $\text{Val}^1(S) = Z_n$, by Lemma 2, there is an integer $z_1, 0 \leq z_1 \leq q_1$, such that the function g_{n,k,z_1} is injective from $\text{Val}^1(\bar{M})$. Similarly, since the set $\text{Val}(\bar{M}) - \text{Val}^1(\bar{M})$ is a subset of $(r-1)k$ elements in $\text{Val}(S) - \text{Val}^1(S) = Z_{(r-1)n}$, by Lemma 2, there is an integer $z_2, 0 \leq z_2 \leq q_2$, such that the function $g_{(r-1)n,(r-1)k,z_2}$ is injective from $\text{Val}(\bar{M}) - \text{Val}^1(\bar{M})$. When these values of z_1 and z_2 are selected in step 5 of the algorithm, by the Claim proved above, the subroutine call **Matching-ext** (S, z_1, z_2, k) in step 5.1 will return a k -matching of the maximum weight in S . Now by step 5.2 of the algorithm, the final collection Q returned by the algorithm in step 6 is a k -matching of the maximum weight in S . This completes the proof of the correctness for the algorithm **WRDM**.

Finally, we study the complexity of the algorithm **WRDM**. We first consider the complexity for the subroutine **Matching-ext** (S', z_1, z_2, h) . By Lemma 4, the $((r-1)k)^2, (r-1)h$ -universal set $\Psi_{((r-1)k)^2, (r-1)h}$ contains at most $((r-1)k)^2 2^{2(r-1)h+12 \log^2((r-1)h)+2}$ splitting functions. Therefore, the loop body, i.e., steps 3.1–3.9, of the subroutine is executed at most

$$\begin{aligned} & k^2 ((r-1)k)^2 2^{2(r-1)h+12 \log^2((r-1)h)+2} \\ & \leq 2^{(r-1)h} k^2 ((r-1)k)^2 2^{12 \log^2((r-1)k)+2} \\ & = 2^{(r-1)h} 2^{12 \log^2((r-1)k)+4 \log((r-1)k)+2} \end{aligned}$$

times (note that $r \geq 3$). Let $T(m, h)$ be the running time of the subroutine **Matching-ext** (S', z_1, z_2, h) , where $m = O(n^r)$ is the size of the collection S' . $T(m, h)$ satisfies the following recurrence relation:

$$T(m, 1) \leq cm;$$

$$T(m, h) \leq 2^{(r-1)h} 2^{12 \log^2((r-1)k) + 4 \log((r-1)k) + 2} \cdot (2T(m, h/2) + cm),$$

where c is a constant. It is not difficult to verify that there are constants c_1 and c_2 such that $T(m, h) \leq c_1 4^{(r-1)h + c_2 \log^3(r-1)k} m$. This shows that the running time of the subroutine **Matching-ext**(S', z_1, z_2, h) is bounded by $O(4^{(r-1)h + O(\log^3(r-1)k)} m)$.

Since the prime number q_1 is bounded by $2n$, and the prime number q_2 is bounded by $2(r-1)n$, from the analysis for the complexity of the subroutine **Matching-ext**(S', z_1, z_2, h), now it is straightforward to conclude that the running time of the algorithm **WRDM**(S, k) is bounded by $O(4^{(r-1)k + O(\log^3(r-1)k)} m^3) = O^*(4^{(r-1)k})$, where $m = O(n^r)$ is the size of the input collection S . This completes the proof of the theorem. \square

We point out that the algorithm **WRDM**(S, k) can be used directly to solve the unweighted r D-Matching problem in time $O^*(4^{(r-1)k})$, which improves the previous best deterministic algorithm of running time $O^*(4^{rk})$ for the problem [4]. Particularly, for the unweighted 3D-Matching problem, we can get a deterministic algorithm of running time $O^*(16^k)$, improving the previous best result $O^*(21.26^k)$ [9].

3 Improved Algorithm for Weighted r -Set Packing

A k -packing is a packing of size k . Let (S, k) be an instance of the weighted r -Set Packing problem, and assume $\text{Val}(S)$ is the set of elements occurring in S , $\text{Val}(S) = n$. Given an instance (S, k) , assume that $\text{Val}(S) = \{1, 2, \dots, n\}$. Given a set s , the element with smallest value in s is the *pivot* of s . Then let $\text{Pivot}(S) = [a_1, \dots, a_l]$ be those elements which are pivots and $a_i < a_j$ if $i < j$.

Lemma 3. *Given an instance (S, k) , let P be the k -packing with maximum weight in S . Suppose s_1, \dots, s_k are sets in P and are sorted in increasing order by their pivots. Let a_j be the pivot of s_i , then a_1, \dots, a_j are not in sets s_{j+1}, \dots, s_k for $1 \leq i \leq k$.*

Proof. Since s_1, \dots, s_k are sorted in increasing order by their pivots, a_{j+1}, \dots, a_l are larger than a_1, \dots, a_j . By definition, any element in s_p are larger than the pivot a_p for $j + 1 \leq p \leq k$. So a_1, \dots, a_j can not be in s_{j+1}, \dots, s_k . \square

Given an instance (S, k) , let P be a k -packing with maximum weight in S . Suppose s_1, \dots, s_k are k sets in P and all sets are sorted in increasing order by their pivots. It is obvious that $\text{Pivot}(S)$ can be found in polynomial time. Also we can find some j such that a_1, \dots, a_j are not in $s_{k/2+1}, \dots, s_k$ according to lemma 3. That j can be found by trying $1, 2, \dots, j$ sequentially. Then we divide $\text{Val}(S) - \{a_1, \dots, a_j\}$ into two disjoint parts W_0 and W_1 such that $\text{Val}(s_1, \dots, s_{k/2}) \subseteq W_0 \cup \{a_1, \dots, a_j\}$, and $\text{Val}(s_{k/2+1}, \dots, s_k) \subseteq W_1$. The idea is implemented in the algorithm **WRSP** given in figure 2.

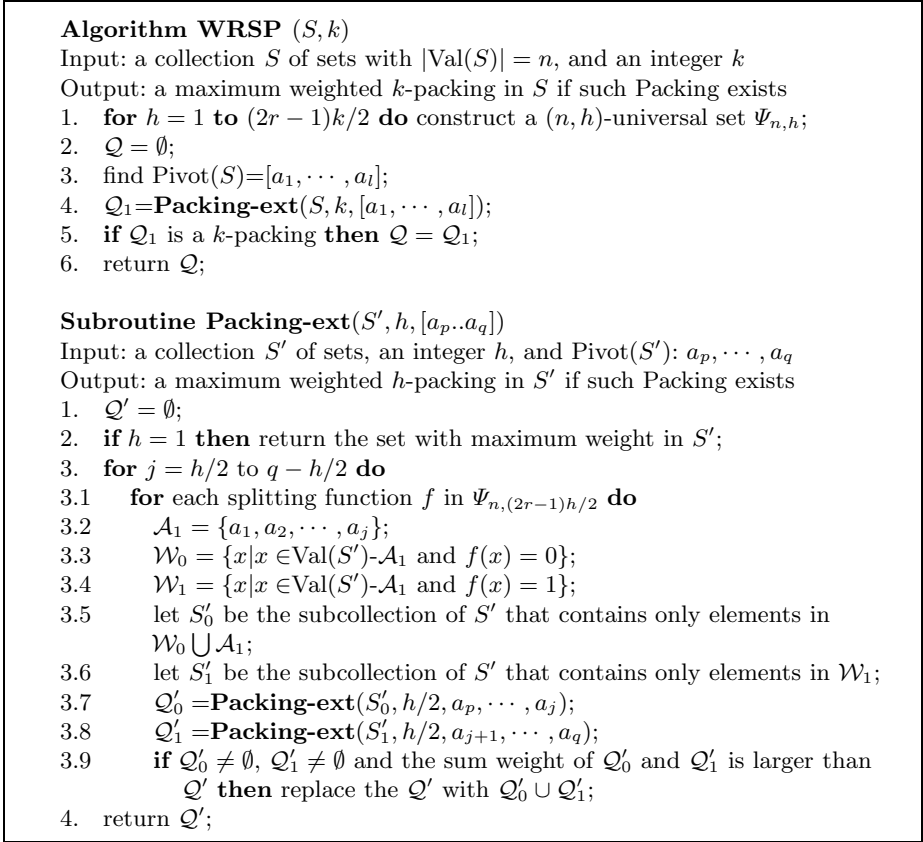


Fig. 2. The algorithm WRSP

Theorem 2. *The algorithm WRSP correctly solves the weighted r -Set Packing problem in time $O(m^2 \log^k 2^{(2r-1)k+O(\log^3((2r-1)k/2))})$, where m is the size of S .*

The proof of Theorem 2 is similar to Theorem 1, which is neglected here.

Now the time complexity is related to m , which is the size of S . However, m can be replaced by $O(k^{O(r)})$. We just apply lemma 2 to have multiple problems of size bounded by $O((rk)^{2r})$.

Given an instance (S, k) of the weighted r -Set Packing problem, let P^* be a k -packing with maximum weight in S . By lemma 2, there must exist an integer z_3 such that makes the function g_{n,rk,z_3} mapping from S to $Z_{(rk)^2}$ injective from $\text{Val}(P^*)$. Given two sets $s_1 = \{x_1, \dots, x_r\}$ and $s_2 = \{y_2, \dots, y_r\}$ in S , s_1 and s_2 are *conflicted* if x_i and y_i are mapped to the same element in $Z_{(rk)^2}$ by $g(n, rk, z_3)$. Given integer z_3 , we can reorganize the sets of S in the following way: (1) If a set s in S contains two or more elements mapping to the same element of $Z_{(rk)^2}$, delete s from S . (2) If two sets s_1 and s_2 are conflicted, delete the

set with smaller weight, or delete any one if they have the same weight. Let S' be the remaining sets after the above process. Since z_3 is an integer that makes g_{n,rk,z_3} injective from $\text{Val}(P^*)$, there must exist a k -packing P' in S' having the same weight as P^* by our operations. Therefore, any k -packing with maximum weight in S' is also a k -packing with maximum weight in S . We note that the size of the S' is bounded by $(rk)^{2r}$, since the number of elements in S' is $(rk)^2$.

Lemma 4. *Given an instance (S, k) of the weighted r -Set Packing problem, let P^* be a k -packing with maximum weight in S . The instance (S, k) can be reduced to at most $2n$ new instances (S', k) such that (1) the size of S' is bounded by $(rk)^{2r}$, and (2) at least one new instance contains a k -packing P' of the same weight as P^* .*

Given an instance (S, k) of the weighted r -Set Packing problem, we first use Lemma 4 to reduce the instance (S, k) to many new instances (S', k) . Then we apply our algorithm **WRSP** (S', k) . If none of the new instances find a k -packing, return \emptyset . Otherwise, return the k -packing of the maximum weight.

Corollary 1. *The weighted r -Set Packing problem can be solved in time $O(2n(rk)^{2r \log k} 2^{(2r-1)k + O(\log^3((2r-1)k/2))}) = O^*(2^{(2r-1)k})$.*

We point out that the algorithm **WRSP** (S, k) can be directly applied to solve the unweighted r -Set Packing problem in time $O^*(2^{(2r-1)k})$, and greatly improves the previous best result $O^*(2^{2rk})$ [4]. Particularly, for unweighted 3-Set Packing problem, we can have an algorithm of time $O^*(32^k)$, which greatly improves the previous best result $O^*(43.62^k)$ [3].

Remark: For the weighted r -Set Packing and weighted r D-Matching problems, a kernel of size $O(k^r)$ can be obtained to replace the m in Theorem 2. Because of the page limitation, the process of getting that kernel is neglected here and can be found in the full paper.

4 Conclusions

In this paper, we study improved algorithms for the weighted r D-Matching and weighted r -Set Packing problems. We present an improved algorithm with time complexity $O^*(4^{(r-1)k})$ for the weighted r D-Matching problem, which greatly improves the previous best result $O^*(4^{rk})$. For the weighted r -Set Packing problem, a more efficient parameterized algorithm in time $O^*(2^{(2r-1)k})$ is presented, which greatly improves the current best result $O^*(2^{2rk})$. Moreover, the proposed algorithms can be applied to solve the unweighted 3D-Matching and unweighted 3-Set Packing problems with time complexity $O^*(16^k)$ and $O^*(32^k)$ respectively, greatly improving the corresponding previous best results. At last, a kernel of size $O(k^r)$ is obtained for the weighted r -Set Packing and weighted r D-Matching problems.

References

1. Downey, R., Fellows, M.: *Parameterized Complexity*. Springer, New York (1999)
2. Liu, Y., Chen, J., Wang, J.: On efficient FPT algorithms for weighted matching and packing problems. In: *Proc. 4th Ann. Conference on Theory and Applications of Models of Computation (TAMC 2007)*. LNCS, vol. 4484, pp. 575–586 (2007)
3. Wang, J., Feng, Q.: Improved parameterized algorithms for weighted 3-set packing. In: Hu, X., Wang, J. (eds.) *COCOON 2008*. LNCS, vol. 5092, pp. 130–139. Springer, Heidelberg (2008)
4. Chen, J., Kneis, J., Lu, S., Mölle, D., Richter, S., Rossmanith, P., Sze, S., Zhang, F.: Randomized divide-and-conquer: improved path, matching, and packing algorithms. *SIAM Journal on Computing* (to appear)
5. Fellows, M., Knauer, C., Nishimura, N., Ragde, P., Rosamond, F., Stege, U., Thilikos, D., Whitesides, S.: Faster fixed-parameter tractable algorithms for matching and packing problems. In: Albers, S., Radzik, T. (eds.) *ESA 2004*. LNCS, vol. 3221, pp. 311–322. Springer, Heidelberg (2004)
6. Chen, J., Friesen, D., Jia, W., Kanj, I.: Using nondeterminism to design efficient deterministic algorithms. *Algorithmica* 40, 83–97 (2004)
7. Jia, W., Zhang, C., Chen, J.: An efficient parameterized algorithm for m -set packing. *Journal of Algorithms* 50, 106–117 (2004)
8. Koutis, I.: A faster parameterized algorithm for set packing. *Information Processing Letters* 94, 7–9 (2005)
9. Liu, Y., Lu, S., Chen, J., Sze, S.H.: Greedy localization and color-coding: improved matching and packing algorithms. In: Bodlaender, H.L., Langston, M.A. (eds.) *IWPEC 2006*. LNCS, vol. 4169, pp. 84–95. Springer, Heidelberg (2006)
10. Wang, J., Feng, Q.: An $O^*(3.52^{3k})$ parameterized algorithm for 3-set packing. In: Agrawal, M., Du, D.-Z., Duan, Z., Li, A. (eds.) *TAMC 2008*. LNCS, vol. 4978, pp. 82–93. Springer, Heidelberg (2008)
11. Koutis, I.: Faster algebraic algorithms for path and packing problems. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfssdóttir, A., Walukiewicz, I. (eds.) *ICALP 2008, Part I*. LNCS, vol. 5125, pp. 575–586. Springer, Heidelberg (2008)
12. Williams, R.: Finding paths of length k in $O^*(2^k)$ time, arXiv:0807.3026v2 [cs.DS] (2008)
13. Naor, M., Schulman, L., Srinivasan, A.: Splitters and near-optimal derandomization. In: *Proc. 39th Annual Symposium on Foundations of Computer Science (FOCS 1995)*, pp. 182–190 (1995)
14. Fredman, M., Komlos, J., Szemerédi, E.: Storing a sparse table with $O(1)$ worst case access time. *Journal of the ACM* 31, 538–544 (1984)
15. Shoup, V.: *A Computational Introduction to Number Theory and Algebra*, 2nd edn. Cambridge University Press, New York (2008)

Parameterized Complexity of Coloring Problems: Treewidth versus Vertex Cover

(Extended Abstract)

Jiří Fiala¹, Petr A. Golovach^{2,*}, and Jan Kratochvíl¹

¹ Institute for Theoretical Computer Science** and Department of Applied Mathematics, Charles University, Prague, Czech Republic
`{fiala,honza}@kam.mff.cuni.cz`

² Institutt for informatikk, Universitetet i Bergen, Norway
`petr.golovach@ii.uib.no`

Abstract. We compare the fixed parameter complexity of various variants of coloring problems (including LIST COLORING, PRECOLORING EXTENSION, EQUITABLE COLORING, $L(p, 1)$ -LABELING and CHANNEL ASSIGNMENT) when parameterized by treewidth and by vertex cover number. In most (but not all) cases we conclude that parametrization by the vertex cover number provides a significant drop in the complexity of the problems.

1 Introduction

An important aspect in the parameterized complexity theory is the choice of the parameter for a problem. One of the best investigated structural parameters for graph problems is the treewidth of the input graph (see e.g. surveys [4,6]). While many problems are FPT when parameterized by the treewidth, there are problems which are NP-hard even for graphs of small fixed treewidth (or even for trees). Also, there are problems which can be solved in polynomial time for graphs of bounded treewidth, but the exponent of the polynomial depends on the width (i.e., they are in XP if parameterized by the treewidth). Some of these problems are known to be W-hard, when parameterized by treewidth, which contributes to the fine structure of the FPT hierarchy. New results in this direction on FPT-complexity of variants of graph coloring and domination problems were recently obtained in [13,9].

For problems which are difficult for graphs of bounded treewidth, it is interesting to consider different structural parameterizations that impose stronger restrictions. Fellows et al. proposed to study parametrization by the vertex cover number and they applied it to graph layout problems in [12]. The goal of this paper is to pursue the road opened in [12] and apply this angle of view to several variants of graph coloring problems, including problems coming from the area of Frequency Assignment.

* Supported by Norwegian Research Council.

** Supported by the Ministry of Education of the Czech Republic as project 1M0545.

The *vertex cover number* of a graph G (denoted by $\mathbf{vc}(G)$) is the minimum size of a set W of vertices of G such that $I = V(G) \setminus W$ is an independent set. For such a partition, the star $K_{1,|I|}$ with the central bag W and leaf-bags $W \cup \{x\}$ for $x \in I$ is a tree-decomposition of G of width W . Hence $\mathbf{tw}(G) \leq \mathbf{vc}(G)$ and parametrization by the vertex cover number indeed has a chance to make problems easier. In most (but not all) coloring and labeling cases considered in this paper, we conclude that parametrization by vertex cover number does make the problem more tractable.

Recall that a (*proper*) *vertex coloring* of a graph is an assignment c of colors to the vertices of the graph such that for any two adjacent vertices u and v , $c(u) \neq c(v)$. Many variants of graph coloring have been intensively studied. We will consider LIST COLORING (where the input graph comes equipped with lists of admissible colors for the vertices, and the question is if there is a proper coloring such that each vertex is colored by a color from its list), PRECOLORING EXTENSION (where some vertices of the input graph are precolored and the other ones are free to be colored by any color; this is a special version of LIST COLORING when each list has either one element or contains all colors), and EQUITABLE COLORING (in which case it is asked that any two color classes differ by at most one in size). All these variants of graph coloring are NP-hard in the general case [21,3] and it is natural to explore their FPT-complexity under different parameterizations as well [5].

Distance constrained labeling of graphs is a concept generalizing graph coloring that stems from the Frequency Assignment Problem. Here the colors (we prefer to call them *labels*) are nonnegative integers and requirements are posed on the difference of labels assigned to vertices that are close to each other [22,7]. In particular, given numbers p and q , an $L(p, q)$ -*labeling* of span k of a graph is a labeling of its vertices by integers from $\{0, 1, \dots, k\}$ such that the labels of adjacent vertices differ by at least p , and the labels of vertices at distance two in the graph differ by at least q . This concept was intensively studied both for the practical motivation and for interesting theoretical properties. E.g., $L(2, 1)$ -LABELING is polynomial time solvable for trees [8] but NP-complete for graphs of treewidth two [14], and for every $p > q > 1$, p and q relatively prime, $L(p, q)$ -LABELING becomes NP-complete already for trees [15].

In this paper, we concentrate on the case $q = 1$; in this case it is simply required that labels assigned to vertices at distance two are distinct. Note also that in the case $p = q = 1$, $L(1, 1)$ -LABELING coincides with coloring the second distance power of the input graph (just beware of the offset 1 between the span of a labeling and the number of colors in a coloring), also previously intensively studied [2,11]. Also because of this meaningful correlation, we treat the case $p = q = 1$ in more detail and consider LIST $L(1, 1)$ -LABELING and $L(1, 1)$ -PRELABELING EXTENSION problems.

Finally, we consider the CHANNEL ASSIGNMENT problem whose input is a graph equipped with integer weights on its edges, and the task is to assign nonnegative integers to its vertices so that the difference of labels assigned to a pair of adjacent vertices is greater or equal than the weight of the corresponding

edge, while minimizing the span of the assignment (i.e., the largest label used). This formulation also stems from the Frequency Assignment Problem, and e.g., the $L(p, q)$ -LABELING problem with input graph G coincides with CHANNEL ASSIGNMENT for the second distance power of G and weights having only two values — p and q . However, note that the transition from G to its second power does not preserve bounded treewidth, so the FPT-complexity results do not follow from one another. Yet CHANNEL ASSIGNMENT is known NP-hard for graphs of treewidth at most three [20] (note only that in this case the size of the input is measured as $n + \log w$ where n is the number of vertices and w the maximum weight of an edge).

A comparison of known and new results on the fixed parameter complexity of the above mentioned problems for parametrization by treewidth versus parametrization by vertex cover number is summarized in Table 1.

Table 1. Complexity of coloring and labeling problems parameterized by treewidth and vertex cover, the results of this paper being denoted by [*]. In the last four rows, k is the parameter (treewidth or vertex cover number).

	Treewidth	Vertex cover
LIST COLORING	W[1]-hard [13]	W[1]-hard [12],[*]
PRECOLORING EXTENSION	W[1]-hard [13]	FPT[*]
EQUITABLE COLORING	W[1]-hard [13]	FPT[*]
$L(0, 1)$ -LABELING	W[1]-hard[*]	FPT[*]
$L(1, 1)$ -LABELING	W[1]-hard[*]	FPT[*]
$L(2, 1)$ -LABELING	NP-c for $k \geq 2$ [14]	FPT[*]
LIST $L(1, 1)$ -LABELING	NP-c for $k \geq 2$ [*]	NP-c for $k \geq 4$ [*]
$L(1, 1)$ -PRELABELING EXTENSION	NP-c for $k \geq 2$ [*]	FPT[*]
CHANNEL ASSIGNMENT	NP-c for $k \geq 3$ [20]	in XP[*]

In this extended abstract we only present samples of proofs of our results. The reader is referred to the full version of the paper for the omitted ones. We also expect that the reader is familiar with the theory of tree decompositions and parameterized complexity. We refer to the book of Downey and Fellows [10] for an excellent exposition of this concept. We consider finite undirected graphs and use standard graph theory notation and terminology.

2 Complexity of Coloring Problems

It has been mentioned without proof in the conclusion of [12] that LIST COLORING remains W[1]-hard when parameterized by the vertex cover number. We note that the problem remains hard even for a special class of *split graphs*. (A graph G is a split graph if its vertex set can be partitioned into a clique and an independent set.)

Theorem 1. *The LIST COLORING problem is W[1]-hard for split graphs with the size of the maximum clique being the parameter.*

Since the size of the maximum clique of a split graph differs from its vertex cover number by no more than one, it also follows (perhaps somewhat surprisingly), that LIST COLORING of split graphs is $W[1]$ -hard when parameterized by the vertex cover number. In contrast with the closely related PRECOLORING EXTENSION:

Theorem 2. *The PRECOLORING EXTENSION problem is FPT when parameterized by the vertex cover number.*

Proof. Suppose that W is a vertex cover of G , and let $|W| = k$. Let $I = I_1, \dots, I_s$ be the partition of the independent set $I = V(G) \setminus W$ such that any two vertices of I belong to the same I_i if and only if they have the same set of neighbors in W . Clearly, $s \leq 2^k - 1$, provided G is connected. Let X be the set of non-precolored vertices of I and let $c_U : V(G) \setminus X \rightarrow \{1, \dots, r\}$ be the given precoloring.

We reduce the problem to the list coloring problem for the subgraph H of G induced by $W \cup X$. For each vertex $v \in X$, we let $L(v) := \{1, \dots, r\}$. If $w \in W$ is precolored, then we let $L(w) := \{c_U(w)\}$, otherwise $L(w) := \{1, \dots, r\} \setminus c_U(N(w_j))$, i.e., we exclude the colors of precolored neighbors of w_j . Clearly, G allows a precoloring extension with at most r colors if and only if H has a feasible list coloring.

Denote by F the subgraph of H induced by W . We distinguish two cases:

If $r > k$ then any feasible list coloring of F can be extended to a list coloring of H by the greedy algorithm. If $|L(w)| \geq k$ for some $w \in W$, then we can remove this vertex since it is always possible to extend any list coloring of the obtained graph to the list coloring of F . The existence of a list coloring of the remaining vertices $w \in W$ with $|L(w)| < k$ can be resolved in $O(k^{k-1})$ time.

If $r \leq k$ then $|L(w)| \leq k$ for any vertex $w \in W$. We consider all colorings of W , and their extensions to H by the greedy algorithm. Since W has at most k^k colorings, the running time is $O(k^{k+1}n)$.

Since H can be constructed in time $O(r(n+m))$ where $n = |V(G)|$ and $m = |E(G)|$, the total running time of the algorithm is $O(k^{k+1}n + r(n+m))$.

The third variant of graph coloring we want to explore is EQUITABLE COLORING. When showing that this problem becomes easy when parameterized by the vertex cover number, we utilize the approach used in [12]. The main idea is to reduce our problem to the integer linear programming problem which is FPT when parameterized by the number of variables. Formally, we use the following problem:

p-VARIABLE INTEGER LINEAR PROGRAMMING FEASIBILITY

Input: A $p \times q$ matrix A with integer elements, an integer vector $b \in \mathbb{Z}^q$.

Parameter: p .

Question: Is there a vector $x \in \mathbb{Z}^p$ such that $A \cdot x \leq b$?

It was proved by Lenstra [19] that this problem is FPT, and this algorithmic result was improved afterward by different authors (see, e. g., a survey [1]):

Theorem 3 ([19,18,16]). *The p -VARIABLE INTEGER LINEAR PROGRAMMING FEASIBILITY problem can be solved using $O(p^{2.5p+o(p)} \cdot L)$ arithmetic operations and space polynomial in L , where L is the number of bits of the input.*

Theorem 4. *The EQUITABLE COLORING problem is FPT when parameterized by the vertex cover number.*

Proof. Let W be a vertex cover of size k of a graph G on n vertices, and let again $\{I_1, \dots, I_s\}$ be the partition of the remaining vertices according to their neighborhoods.

Let r be the required number of colors. Set $t = \lfloor \frac{n}{r} \rfloor$. Any equitable coloring of G contains $a = n - rt$ color classes of cardinality $t + 1$ and $b = r - a$ color classes of cardinality t . We distinguish two cases:

If $r \leq k$, then for each proper coloring V_1, \dots, V_r of W we construct a system of linear integer inequalities with sr variables $x_{i,j}$, $i \in \{1, \dots, s\}$ and $j \in \{1, \dots, r\}$, where $x_{i,j}$ is the number of vertices of color j in the set I_i :

$$\begin{cases} x_{i,j} \geq 0, \\ x_{i,j} = 0, & \text{if color } j \text{ is used in } N(I_i), \\ x_{1,j} + \dots + x_{s,j} = t + 1 - |W \cap V_j|, & \text{if } j \in \{1, \dots, a\}, \\ x_{1,j} + \dots + x_{s,j} = t - |W \cap V_j|, & \text{if } j \in \{a + 1, \dots, r\}. \end{cases}$$

It can be easily seen that this problem has an integer solution if and only if there is an equitable coloring of G which extends the starting coloring of W . Since W has at most k^k colorings and the number of variables is at most $k(2^k - 1)$, the EQUITABLE COLORING problem can be solved in FPT-time.

If $r > k$ then we impose the following assumptions on the desired equitable coloring: Vertices of W are colored by colors $\{1, \dots, k\}$ and there exists an integer l between $\max\{0, k - b\}$ and $\min\{k, a\}$ such that the color classes V_1, \dots, V_l contain $t + 1$ vertices and the color classes V_{l+1}, \dots, V_k only t vertices.

By permuting the names of colors, every equitable coloring of G gives rise to a coloring satisfying the above conditions. On the other hand, if a partial coloring of G with k colors exists, such that all vertices of W are colored and the conditions are satisfied, then it can be extended to an equitable coloring of G : Any color from the set $\{k + 1, \dots, r\}$ can be used for coloring of an arbitrary vertex of I .

Hence we consider all at most k^k colorings of W by colors $1, \dots, k$, and for every l such that $\max\{0, k - b\} \leq l \leq \min\{k, a\}$, we construct a system of linear integer inequalities with sk variables $x_{i,j}$, $i \in \{1, \dots, s\}$ and $j \in \{1, \dots, k\}$, where $x_{i,j}$ is the number of vertices of color j in the set I_i :

$$\begin{cases} x_{i,j} \geq 0, \\ x_{i,j} = 0, & \text{if color } j \text{ used in } N(I_i), \\ x_{1,j} + \dots + x_{s,j} = t + 1 - |W \cap V_j|, & \text{if } j \in \{1, \dots, l\}, \\ x_{1,j} + \dots + x_{s,j} = t - |W \cap V_j|, & \text{if } j \in \{l + 1, \dots, k\}. \end{cases}$$

Since the number of variables is bounded by $k(2^k - 1)$, we again conclude that the problem is solvable in FPT time.

3 Complexity of the $L(p, 1)$ -LABELING Problems

In this section we consider the $L(p, 1)$ -LABELING problems for $p = 0, 1$. It was proved in [14] that the $L(2, 1)$ -LABELING problem is NP-complete even for graphs of treewidth two (and this result can be extended for any fixed $p \geq 2$). On the other hand, it was shown in [23] that the $L(1, 1)$ -LABELING problem can be solved by a dynamic programming algorithm in time $O(\Delta^{2^{8(t+1)+1}} \cdot n) + O(n^3)$, for n -vertex graphs of treewidth at most t with maximum degree Δ , and the same holds for $L(0, 1)$ -LABELING. We show here that it is impossible to solve these problems in FPT-time unless $\text{FPT} = \text{W}[1]$.

Theorem 5. *The $L(0, 1)$ -LABELING and $L(1, 1)$ -LABELING problems are $\text{W}[1]$ -hard when parameterized by the treewidth.*

Proof. As a sample of a hardness proof, we show the result for $L(1, 1)$ -LABELING, the proof for $L(0, 1)$ -LABELING is analogous. We reduce from the $\text{EQUITABLE COLORING}$ problem. It was proved in [13] that it is $\text{W}[1]$ -hard when parameterized both by the treewidth and r .

Let us start with auxiliary constructions. Suppose that $l \leq \lambda - 2$ is a positive integer. We define the graph $F(l)$ with vertices $a_1, \dots, a_l, b_1, \dots, b_l, f, g$ and $c_1, \dots, c_{\lambda-l-2}$ as follows. Each vertex a_i is joined by an edge with f , each vertex b_i is joined with g , and f and g are joined with each other and with all vertices c_j . We call the vertices a_i the A -roots of $F(l)$, and the vertices b_i the B -roots. The vertex f is called the F -vertex, vertex g is called the G -vertex, and vertices c_j are called the C -vertices. We need the following property of $F(l)$.

Lemma 1. *For any $L(1, 1)$ -labeling of $F(l)$ of the span λ , the set of labels used on the A -roots is the same as the set of labels used on the B -roots, and all A -roots (B -roots) are labeled by different labels. Also any labeling of A -roots by different integers from the set $\{1, \dots, \lambda\}$ can be extended to an $L(1, 1)$ -labeling of $F(l)$.*

Now we describe the reduction. Let G be a graph on n vertices u_1, \dots, u_n with m edges, for which an equitable coloring by r colors is questioned. Assume without loss of generality that r divides n , and let $l = \frac{n}{r}$. Define $\lambda = n + m + 1$. Let $(\{X_i \mid i \in V(T)\}, T)$ be a tree decomposition of G of width $t = \text{tw}(G)$. We assume that T has maximum degree three, and for any two adjacent nodes i and j of T , $|X_i \setminus X_j| + |X_j \setminus X_i| \leq 1$ (this may increase the size of T linearly). Choose some walk $P = i_1 \dots i_s$ in T which contains all nodes and visits any node at most three times. Let e_1, \dots, e_m be edges of G enumerated in the order in which they occur in bags X_{i_1}, \dots, X_{i_m} . It is assumed also that $|X_{i_1}| = 1$ (it is possible to choose a leaf of T as the starting point of the walk).

For every $i \in \{1, \dots, r\}$, m copies of $F(l)$ are constructed and joined consecutively by gluing B -roots of each copy with A -roots of the next copy (see Fig. 1). Denote by A_i A -roots of the first copy, and by $B_{i,j}$ ($C_{i,j}$, $f_{i,j}$ and $g_{i,j}$ correspondingly) B -roots (C -vertices, F -vertices and G -vertices correspondingly) of the j -th copy. We proceed by adding a copy of $F(n)$ with A -roots v_1, \dots, v_n and with B -roots united with vertices of $A_1 \cup \dots \cup A_r$. Denote by f_0, g_0 and

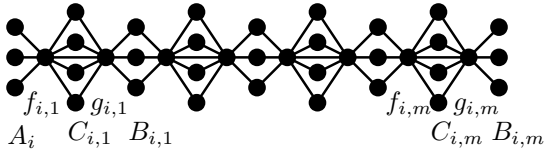


Fig. 1. Construction of chains of graphs $F(l)$

C_0 F-vertices, G-vertices and C-vertices of $F(n)$ correspondingly. For each edge $e_j = u_p u_q$, a vertex w_j is introduced and joined by edges with v_p , v_q , and with $l - 1$ vertices of sets $B_{i,j}$ for $i \in \{1, \dots, r\}$. Denote the obtained graph by H (see Fig. 2). The proof is concluded by the following lemma (whose proof can be found in the full version of the paper).

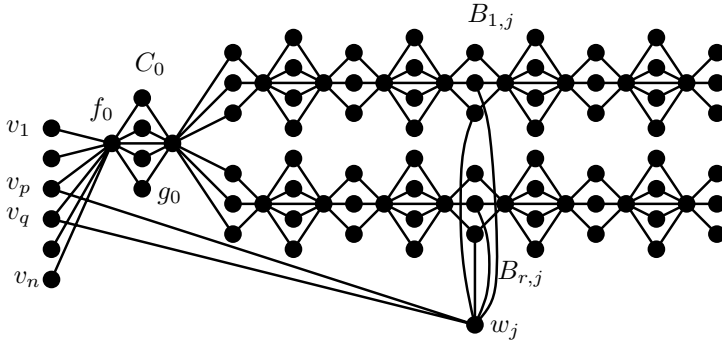


Fig. 2. Construction of H

Lemma 2. *The graph G has an equitable coloring by r colors if and only if H has an $L(1, 1)$ -labeling of span λ . Moreover, $\text{tw}(H) \leq (2r + 2)t + 3r + 2$.*

In the case of $L(p, 1)$ -LABELING problems, the decrease in complexity is most visible (note that for $p > 1$ even from NP-hardness to FPT):

Theorem 6. *For every p , the $L(p, 1)$ -LABELING problem is FPT when parameterized by the vertex cover number.*

Proof. As in the case of the *EQUITABLE COLORING* problem we use reductions to systems of integer linear inequalities. We only note here that the case $p = 0, 1$ is considerably simpler than the case $p > 1$, since we have to be careful about the linear ordering of the label space in the latter case.

4 Labeling as Coloring of the Distance Power

We have already mentioned that the special case of $L(p, q)$ -LABELING for $p = q = 1$ coincides with the coloring of the second distance power of the input

graph. As such, it has attracted attention of many graph theorists, and we also want to reserve some extra space to refining our results from the previous section. In particular, we will pay closer attention to the LIST and PRELABELING variants of the problem. We prefer to stay in the *labeling* setting because the FPT result holds for general p . (The hardness results are new and interesting just for $p = 1$, since it was known that the LIST $L(p, 1)$ -LABELING and $L(p, 1)$ -PRELABELING EXTENSION problems for $p \geq 2$ are NP-complete for graphs of treewidth two [17,14].)

Theorem 7. *The LIST $L(1, 1)$ -LABELING problem is NP-complete for graphs of treewidth at most two, as well as for graphs of vertex cover number at most four, even if all lists have at most three elements.*

Somewhat surprisingly, the complexity of $L(1, 1)$ -PRELABELING EXTENSION differs when parameterized by treewidth or vertex cover number:

Theorem 8. *The $L(1, 1)$ -PRELABELING EXTENSION problem is NP-complete for graphs of treewidth at most two.*

Theorem 9. *For every p , the $L(p, 1)$ -PRELABELING EXTENSION problem is in the class FPT when parameterized by the vertex cover number.*

5 Complexity of the CHANNEL ASSIGNMENT Problem

Recall that for the CHANNEL ASSIGNMENT problem, the span of the assignment - as a part of the input - is measured in binary encoding. Since CHANNEL ASSIGNMENT is known to be NP-complete for graphs of treewidth 3, the following theorem proves a drop in complexity under parametrization by the vertex cover number. However, it does not settle its FPT status, and we thus leave this question as an open problem.

Theorem 10. *For every k , the CHANNEL ASSIGNMENT problem is solvable in polynomial time for graphs of vertex cover number bounded by k .*

Proof. We will actually show that the minimum span of a feasible labeling can be computed in polynomial time, if the input graph has bounded vertex cover number. Towards this end suppose that G comes equipped with a weight function $w : E(G) \rightarrow Z^+$ with all weights at most w_{max} . Let $V(G) = W \cup I$ be a partition into a vertex cover W of size k and an independent set I .

Suppose G, w allows a channel assignment of span λ , and consider a (hypothetical) assignment $f : V(G) \rightarrow \{0, 1, \dots, \lambda\}$ which minimizes the sum $\sum_{x \in V(G)} f(x)$. Construct an auxiliary directed graph \tilde{G} with vertex set $W \cup X$ for some $X \subset I$ as follows: For every $u \in W$, follow one of the following rules (if more than one are applicable, choose an arbitrary one)

1. if there is an $x \in I$ such that $f(x) = 0$, $xu \in E(G)$ and $f(u) = w(xu)$, then add one such x to X and the arc ux to $E(\tilde{G})$,

2. if there is a $v \in W$ such that $uv \in E(G)$ and $f(u) = f(v) + w(uv)$, then add one such arc uv to $E(\tilde{G})$,
3. if there are $x \in I, v \in W$, such that $vx, xv \in E(G)$ and $f(x) = f(v) + w(vx)$ and $f(u) = f(x) + w(xu)$, then add one such vertex x to X , and the arcs ux, xv to $E(\tilde{G})$.

This auxiliary graph is a directed forest, all sinks are labeled 0, and all other vertices have outdegree 1 (if some $u \in W$ had outdegree 0, then reassigning $f'(u) = f(u) - 1$ would yield a valid assignment with a smaller sum of labels). Let us call such a directed graph a *scenario*. Since each vertex of W which is not a sink can be adjacent to $n - k$ sinks from I or can be adjacent to at most $k - 1$ vertices of W or can be connected by directed paths of length two to a vertex in W in at most $(n - k)(k - 1)$ ways, the number of possible scenarios is at most $(k(n - k + 1))^k = O(k^k n^k)$.

For each scenario, we check if it extends to a valid channel assignment and what would be the minimum span of such an extension. For a particular scenario with vertex set $W \cup X$, the labels of the vertices $W \cup X$ are uniquely determined by the scenario. We first check all $O(k^2)$ edges between the vertices of $W \cup X$, and then we attend to the vertices of $I \setminus X$. Let u_1, u_2, \dots, u_k be an ordering of W determined by the scenario such that $f(u_1) \leq f(u_2) \leq \dots \leq f(u_k)$. For each vertex $z \in I \setminus X$, we check whether it fits in some interval $[f(u_i) .. f(u_{i+1})]$. This can be done in time linear in $(\log w_{max} + \log n)k$ by checking if $\max_{j \leq i}(f(u_j) + w(u_j z)) \leq \min_{j \geq i+1}(f(u_j) - w(u_j z))$. If none of these intervals is available for $f(z)$, and neither is the interval $[0 .. f(u_1)]$, we have to label z by $f(z) = \min_{1 \leq j \leq k}(f(u_j) + w(u_j z))$. Finally we compute the maximum of all labels used to get the span. In this way we compute the minimum possible span of a channel assignment in time $O(k^{k+2} n^{k+1} (\log w_{max} + \log n))$.

References

1. Aardal, K., Weismantel, R., Wolsey, L.A.: Non-standard approaches to integer programming. *Discrete Appl. Math.* 123, 5–74 (2002); Workshop on Discrete Optimization, DO 1999, Piscataway, NJ (1999)
2. Agnarsson, G., Halldórsson, M.M.: Coloring powers of planar graphs. *SIAM J. Discrete Math.* 16, 651–662 (2003) (electronic)
3. Alon, N.: Restricted colorings of graphs, in *Surveys in combinatorics, 1993* (Keele). London Math. Soc. Lecture Note Ser., vol. 187, pp. 1–33. Cambridge Univ. Press, Cambridge (1993)
4. Bodlaender, H.L.: Treewidth: Characterizations, applications, and computations. In: Fomin, F.V. (ed.) *WG 2006*. LNCS, vol. 4271, pp. 1–14. Springer, Heidelberg (2006)
5. Bodlaender, H.L., Fomin, F.V.: Equitable colorings of bounded treewidth graphs. *Theoret. Comput. Sci.* 349, 22–30 (2005)
6. Bodlaender, H.L., Koster, A.M.C.A.: Combinatorial optimization on graphs of bounded treewidth. *Comput. J.* 51, 255–269 (2008)
7. Calamoneri, T.: The $l(h, k)$ -labelling problem: A survey and annotated bibliography. *Comput. J.* 49, 585–608 (2006)

8. Chang, G.J., Kuo, D.: The $L(2, 1)$ -labeling problem on graphs. *SIAM J. Discrete Math.* 9, 309–316 (1996)
9. Dom, M., Lokshantov, D., Saurabh, S., Villanger, Y.: Capacitated domination and covering: A parameterized perspective. In: Grohe, M., Niedermeier, R. (eds.) *IWPEC 2008*. LNCS, vol. 5018, pp. 78–90. Springer, Heidelberg (2008)
10. Downey, R.G., Fellows, M.R.: *Parameterized complexity*. Monographs in Computer Science. Springer, New York (1999)
11. Dvořák, Z., Král, D., Nejedlý, P., Škrekovski, R.: Coloring squares of planar graphs with girth six. *European J. Combin.* 29, 838–849 (2008)
12. Fellows, M., Lokshantov, D., Misra, N., Rosamond, F.A., Saurabh, S.: Graph layout problems parameterized by vertex cover. In: *ISAAC (2008)*
13. Fellows, M.R., Fomin, F.V., Lokshantov, D., Rosamond, F.A., Saurabh, S., Szeider, S., Thomassen, C.: On the complexity of some colorful problems parameterized by treewidth. In: Dress, A.W.M., Xu, Y., Zhu, B. (eds.) *COCOA*. LNCS, vol. 4616, pp. 366–377. Springer, Heidelberg (2007)
14. Fiala, J., Golovach, P.A., Kratochvíl, J.: Distance constrained labelings of graphs of bounded treewidth. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) *ICALP 2005*. LNCS, vol. 3580, pp. 360–372. Springer, Heidelberg (2005)
15. Fiala, J., Golovach, P.A., Kratochvíl, J.: Computational complexity of the distance constrained labeling problem for trees (extended abstract). In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfssdóttir, A., Walukiewicz, I. (eds.) *ICALP 2008, Part I*. LNCS, vol. 5125, pp. 294–305. Springer, Heidelberg (2008)
16. Frank, A., Tardos, É.: An application of simultaneous Diophantine approximation in combinatorial optimization. *Combinatorica* 7, 49–65 (1987)
17. Golovach, P.A.: Systems of pairs of q -distant representatives, and graph colorings. *Zap. Nauchn. Sem. S.-Peterburg. Otdel. Mat. Inst. Steklov. (POMI)* 293, 5–25, 181 (2002)
18. Kannan, R.: Minkowski’s convex body theorem and integer programming. *Math. Oper. Res.* 12, 415–440 (1987)
19. Lenstra Jr., H.W.: Integer programming with a fixed number of variables. *Math. Oper. Res.* 8, 538–548 (1983)
20. McDiarmid, C., Reed, B.: Channel assignment on graphs of bounded treewidth. *Discrete Math.* 273, 183–192 (2003); *EuroComb 2001* (Barcelona)
21. Tuza, Z.: Graph colorings with local constraints—a survey. *Discuss. Math. Graph Theory* 17, 161–228 (1997)
22. Yeh, R.K.: A survey on labeling graphs with a condition at distance two. *Discrete Math.* 306, 1217–1231 (2006)
23. Zhou, X., Kanari, Y., Nishizeki, T.: Generalized vertex-coloring of partial k -trees. *IEICE Trans. Fundamentals of Electronics, Communication and Computer Sciences E83-A*, 671–678 (2000)

Discovering Almost Any Hidden Motif from Multiple Sequences in Polynomial Time with Low Sample Complexity and High Success Probability

Bin Fu¹, Ming-Yang Kao², and Lusheng Wang³

¹ Dept. of Computer Science, University of Texas – Pan American
TX 78539, USA

`binfu@cs.panam.edu`

² Department of Electrical Engineering and Computer Science,
Northwestern University, Evanston, IL 60208, USA

`kao@northwestern.edu`

³ Department of Computer Science, The City University of Hong Kong,
Kowloon, Hong Kong

`lwang@cs.cityu.edu.hk`

Abstract. We study a natural probabilistic model for motif discovery that has been used to experimentally test the effectiveness of motif discovery programs. In this model, there are k background sequences, and each character in a background sequence is a random character from an alphabet Σ . A motif $G = g_1g_2 \dots g_m$ is a string of m characters. Each background sequence is implanted a probabilistically generated approximate copy of G . For a probabilistically generated approximate copy $b_1b_2 \dots b_m$ of G , every character is probabilistically generated such that the probability for $b_i \neq g_i$ is at most α . It has been conjectured that multiple background sequences can help with finding faint motifs G .

In this paper, we develop an efficient algorithm that can discover a hidden motif from a set of sequences for any alphabet Σ with $|\Sigma| \geq 2$ and is applicable to DNA motif discovery. We prove that for $\alpha < \frac{1}{4}(1 - \frac{1}{|\Sigma|})$ and any constant $x \geq 8$, there exist positive constants c_0, ϵ, δ_1 and δ_2 such that if the length ρ of motif G is at least $\delta_1 \log n$, and there are $k \geq c_0 \log n$ input sequences, then in $O(n^2 + kn)$ time this algorithm finds the motif with probability at least $1 - \frac{1}{2^x}$ for every $G \in \Sigma^\rho - \Psi_{\rho, h, \epsilon}(\Sigma)$, where ρ is the length of the motif, h is a parameter with $\rho \geq 4h \geq \delta_2 \log n$, and $\Psi_{\rho, h, \epsilon}(\Sigma)$ is a small subset of at most $2^{-\Theta(\epsilon^2 h)}$ fraction of the sequences in Σ^ρ . The constants c_0, ϵ, δ_1 and δ_2 do not depend on x when x is a parameter of order $O(\log n)$. Our algorithm can take any number k sequences as input.

1 Introduction

Motif discovery is an important problem in computational biology and computer science. It has applications to coding theory [3,5], locating binding sites

and conserved regions in unaligned sequences [19,11,7,18], genetic drug target identification [10], designing genetic probes [10], and universal PCR primer design [14,2,17,10].

This paper focuses on the application of motif discovery to finding conserved regions in a set of given DNA, RNA, or protein sequences. Such conserved regions may represent common biological functions or structures. Many performance measures have been proposed for motif discovery. Let C be a subset of 0-1 sequences of length n . The *covering radius* of C is the smallest integer r such that each vector in $\{0,1\}^n$ is at a distance at most r from C . The decision problem associated with the covering radius for a set of binary sequences is NP-complete [3]. Another similar problem called closest string problem was also proved to be NP-hard [3,10]. Some approximation algorithms have also been proposed. Li et al. [13] gave an approximation scheme for the closest string and substring problems. The related consensus patterns problem is that give n sequences s_1, \dots, s_n , find for a region of length L in each s_i , and a *median* string s of length L so that the total Hamming distance from s to these regions is minimized. Approximation algorithms for the consensus patterns problem were reported in [12]. Furthermore, a number of heuristics and programs have been developed [16,8,9,20,11].

In many applications, motifs are faint and may not be apparent when two sequences are compared alone but may become clearer when more sequences are compared together [6]. It has been conjectured that comparing more sequences can help with identifying faint motifs. In this paper, we give an analytical proof for this conjecture by providing an algorithm that can find almost any hidden motif from multiple sequences in polynomial time with low sample complexity and high success probability.

We study a natural probabilistic model for motif discovery. In this model, there are k background sequences and each character in the background sequence is a random character from an alphabet Σ . A motif $G = g_1g_2 \dots g_m$ is a string of m characters. Each background sequence is implanted a probabilistically generated approximate copy of G . For a probabilistically generated approximate copy $b_1b_2 \dots b_m$ of G , every character is probabilistically generated such that the probability for $b_i \neq g_i$ (mutation) is at most α . This model was first proposed in [16] and has been widely used in experimentally testing motif discovery programs [8,9,20,11]. We note that a mutation in our model converts a character g_i in the motif into a different character b_i without probability restriction. This means that a character g_i in the motif may not become any character b_i in $\Sigma - \{g_i\}$ with equal probability.

We design an algorithm that for a reasonably large k can discover the implanted motif with high probability. Specially, we prove that for $\alpha < \frac{1}{4}(1 - \frac{1}{|\Sigma|})$ and any constant $x \geq 8$, there exist positive constants c_0, ϵ, δ_1 and δ_2 such that if the length of the hidden motif is at least $\delta_1 \log n$ and there are $k \geq c_0 \log n$ input sequences, then in $O(n^2 + kn)$ time this algorithm finds the motif with probability at least $1 - \frac{1}{2^x}$ for every $G \in \Sigma^\rho - \Psi_{\rho,h,\epsilon}(\Sigma)$, where Σ is the alphabet with $|\Sigma| \geq 2$, ρ is the length of the motif, h is a parameter with $\rho \geq 4h \geq \delta_2 \log n$,

n is the longest length of any input sequence, and $\Psi_{\rho,h,\epsilon}(\Sigma)$ is a small subset of a $2^{-\Theta(\epsilon^2 h)}$ fraction of the sequences of Σ^ρ . If x is considered as a parameter of order $O(\log n)$, then c_0, δ_1 and δ_2 do not depend on x .

Our algorithm can take any number k sequences as input. We give an analysis of how the motif is recovered with probability depending on k . The more sequences we have, the more likely the algorithm recovers the motif.

Our algorithm is a deterministic algorithm that has a provable high success probability to return the exact correct hidden motif. The randomness in the input sequences is the only source of randomness for the algorithm. The algorithm needs the motif to be long enough but does not need to know the length of the motif. In a related work [4], we need that for the alphabet has large constant size.

2 The Sequence Model and Some Preliminaries

For a set A , $|A|$ denotes the number of elements in A . Σ is an alphabet with $|\Sigma| = t \geq 2$. For an integer $n \geq 0$, Σ^n is the set of sequences of length n with characters from Σ . For a sequence $S = a_1 a_2 \cdots a_n$, $S[i]$ denotes the character a_i , $S[i, j]$ denotes the substring $a_i \cdots a_j$ for $1 \leq i \leq j \leq n$, and $|S|$ denotes the length of the sequence S . We use \emptyset to represent the empty sequence, which has length 0.

Let $G = g_1 g_2 \cdots g_m$ be a fixed sequence of m characters. G is the hidden motif to be discovered by our algorithm. A $\Theta_\alpha(n, G)$ -sequence has the form $S = a_1 \cdots a_{n_1} b_1 \cdots b_m a_{n_1+1} \cdots a_{n_2}$, where $n_2 + m \leq n$, each a_i has probability $\frac{1}{t}$ to be equal to π for each $\pi \in \Sigma$, and b_i has probability at most α not equal to g_i for $1 \leq i \leq m$, where $m = |G|$. $\aleph(S)$ denotes the *motif region* $b_1 \cdots b_m$ of S . A mutation converts a character g_i in the motif into an arbitrary different character b_i without probability restriction. This allows a character g_i in the motif to change into any character b_i in $\Sigma - \{g_i\}$ with even different probability.

For two sequences $S_1 = a_1 \cdots a_m$ and $S_2 = b_1 \cdots b_m$ of the same length, let $\text{diff}(S_1, S_2) = \frac{| \{i | a_i \neq b_i (i=1, \dots, m)\} |}{m}$, i.e., the ratio of difference between the two sequences.

Theorem 1 ([15]). *Let X_1, \dots, X_n be n independent random 0-1 variables, where X_i takes 1 with probability p_i . Let $X = \sum_{i=1}^n X_i$, and $\mu = E[X]$. Then for any $\delta > 0$, 1. $\Pr(X < (1 - \delta)\mu) < e^{-\frac{1}{2}\mu\delta^2}$, and 2. $\Pr(X > (1 + \delta)\mu) < \left[\frac{e^\delta}{(1+\delta)^{(1+\delta)}} \right]^\mu$.*

Corollary 1 ([13]). *Let X_1, \dots, X_n be n independent random 0-1 variables and $X = \sum_{i=1}^n X_i$. Then 1. If X_i takes 1 with probability at most p , then for any $\frac{1}{3} > \epsilon > 0$, $\Pr(X > pn + \epsilon n) < e^{-\frac{1}{3}n\epsilon^2}$; and 2. If X_i takes 1 with probability at least p , then for any $\frac{1}{3} > \epsilon > 0$, $\Pr(X < pn - \epsilon n) < e^{-\frac{1}{2}n\epsilon^2}$.*

3 An Overview of the Main Algorithm

The main algorithm Discover-Motif has two phases. Phase 1 consists of a subroutine called Align-Sequences. Phase 2 consists of a subroutine called Recover-Motif. We outline these subroutines below.

In Phase 1, we select a pair of $\Theta_\alpha(n, G)$ -sequences S_1, S_2 and find two equal length substrings U and V of S_1 and S_2 respectively such that U and V are of length at least $c_0 \log n$ for some constant c_0 and U and V only have a small difference. The substring U of S_1 is extracted.

In Phase 2, we use U to search for similar patterns in other input sequences and align the sequences so that their copies of the motif will start at the same column. A new set of $\Theta_\alpha(n, G)$ -sequences S_3, S_4, \dots, S_k will be used for these input sequences. The subsequence U extracted in the first phase is used to match a substring X_i of S_i for $i = 2, \dots, k$. Assume that X_2, \dots, X_k are derived from matching U to all sequences S_2, S_2, \dots, S_k . We align S_1, \dots, S_k by putting them into k different rows so that all X_i ($i = 2, \dots, k$) stay in the same column region with U in S_1 . This is shown in the transformation from Figure 1 to Figure 2.

For each column in the motif region, the i -th character $G[i]$ of G can be recovered by voting among the characters of S_1, \dots, S_k in the same column. In other words, $G[i]$ is the character that appears more than the other characters in the same column. We will prove in Section 4 that with high probability, each $G[i]$ can be recovered correctly.

On the other hand, if a column is not in the region of the motif, we can prove that with high probability, all characters appear with similar frequencies in the column.

In the following, Definition 1 characterizes the motifs which are difficult for our algorithm to discover. Lemma 1 shows that such difficult motifs are relatively few. Lemma 1 is based on the fact that for a random sequence, two equal length subsequences are not similar to each other if they are long enough.

Definition 1. Let Σ be an alphabet with at least 2 characters. Let h and m be integers with $h \leq m$. Define $\Phi_{m,h,\epsilon}(\Sigma)$ be the set of all sequences S in Σ^m such that $\text{diff}(S[i, i + h - 1], S[j, j + h - 1]) \leq (1 - \frac{1}{|\Sigma|}) - \epsilon$ for some two $i \neq j$ with $1 \leq i < j \leq m - h + 1$. Define $\Psi_{m,h,\epsilon}(\Sigma) = \cup_{u=h}^m \Phi_{m,u,\epsilon}(\Sigma)$. We note that $\text{diff}(\cdot)$ is defined in Section 2.

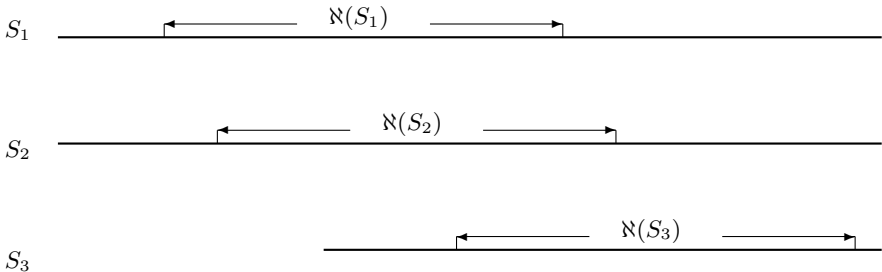


Fig. 1. The motif regions of S_1, S_2 and S_3 are not aligned

Lemma 1. For every constant $\epsilon > 0$, $\frac{|\Psi_{\rho,h,\epsilon}(\Sigma)|}{|\Sigma^\rho|} \leq \rho^2 \frac{c^h}{1-c}$, where $c = e^{-\frac{2}{3}} < 1$.

Proof. Assume that S is a random sequence of Σ^ρ . Let $u \geq h$. We consider $S[i, i + u - 1]$ and $S[j, j + u - 1]$. The probability that the t -th characters of these two subsequences are not the same is $1 - \frac{1}{|\Sigma|}$. By Corollary 1 with probability at most $e^{-\frac{\epsilon^2 u}{3}}$, $\text{diff}(S[i, i + u - 1], S[j, j + u - 1]) \leq (1 - \frac{1}{|\Sigma|}) - \epsilon$. Therefore, with probability at most $\rho^2 e^{-\frac{\epsilon^2 u}{3}}$, there exist i and j with $1 \leq i < j \leq \rho - u + 1$ such that $\text{diff}(S[i, i + u - 1], S[j, j + u - 1]) \leq (1 - \frac{1}{|\Sigma|}) - \epsilon$.

Therefore, with probability at most $\sum_{u=h}^\rho \rho^2 e^{-\frac{\epsilon^2 u}{3}} \leq \rho^2 \frac{c^h}{1-c}$, there are integers i, j , and u with $1 \leq i < j \leq \rho - u + 1$ and $h \leq u \leq \rho$ such that $\text{diff}(S[i, i + u - 1], S[j, j + u - 1]) \leq (1 - \frac{1}{|\Sigma|}) - \epsilon$.

For an intuitive understanding of Lemma 1, note that $\Psi_{\rho,h,\epsilon}(\Sigma)$ is a subset of Σ^ρ . If y and ϵ are constants, we can select constant c such that $\rho^2 \frac{c^h}{1-c} < \frac{1}{2y}$ with $h \geq c \log \rho$. Therefore, $\Psi_{\rho,h,\epsilon}(\Sigma)$ is a small portion of sequences in Σ^ρ when, for example, $y \geq 10$.

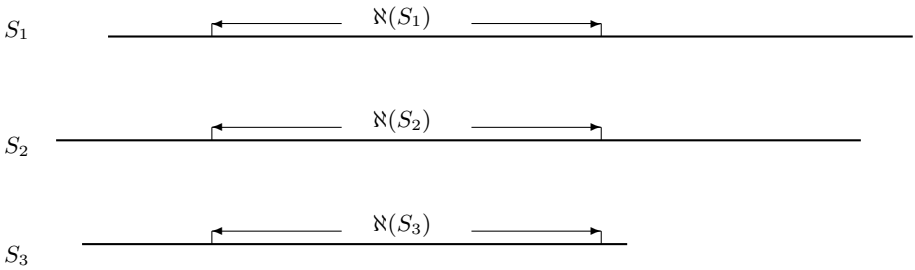


Fig. 2. S_1, S_2 and S_3 with their motifs in the same column region

4 Main Theorem and Main Algorithm Discover-Motif

Definition 2. Assume $\sigma_1 > 0$. Given k $\Theta_\alpha(n, G)$ -sequences S_1, \dots, S_k , an alignment puts them into k rows such that each sequence S_i is arranged in $|S_i|$ consecutive positions at row i for the $|S_i|$ characters of S_i . An alignment for S_1, \dots, S_k is a σ_1 -error alignment for S_1, \dots, S_k if at least $(1 - \sigma_1)k$ sequences have their $\aleph(S_i)$ in the same columns. We often use an array A of k rows to hold an alignment of k sequences.

Our main algorithm Discover-Motif consists two sub-routines. The first sub-routine Align-Sequences aligns the input sequences so that most copies of the motif are in the same column regions. The second sub-routine Recover-Motif recovers the motif based on the output of the first sub-routine. The performance of Discover-Motif, Align-Sequences, and Recover-Motif is described in the following theorem.

Theorem 2. Assume that α with $\alpha < \frac{1}{4}(1 - \frac{1}{|\Sigma|})$ and x are positive constants. Then, there exist constants $\sigma_1, \delta_1, \delta_2, \epsilon > 0$ such that given an input of $\Theta_\alpha(n, G)$ -sequences S_1, \dots, S_k with $G \in \Sigma^\rho - \Psi_{\rho, h, \epsilon}(\Sigma)$ and $\rho \geq 4h \geq \delta_2 \log n$, Discover-Motif which consists sub-routines Align-Sequences and Recover-Motif satisfies the following two conditions for all sufficiently large n (the maximum length of the input sequences):

1. With probability at most $e^{-\Omega(h)} + e^{-\Omega(k)}$, Align-Sequences fails to return a σ_1 -error alignment for S_1, \dots, S_k . Align-Sequences runs in $O(n^2 + kn \log n)$ worst-case time.
2. Given a σ_1 -error alignment for S_1, \dots, S_k , Recover-Motif returns $g'_1 \dots g'_s$ such that with probability at most $e^{-\Omega(k)}$, $s \neq |G|$ and for each $i = 1, \dots, s$, with probability at most $e^{-\Omega(k)}$, each $g'_i \neq G[i]$. Recover-Motif runs in time $O(kn) + O(n \log n)$.

Proof. The algorithms Discover-Motif, Align-Sequences, and Recover-Motif are detailed in Sections 4.3, 4.1, and 4.2, respectively.

The proof of the theorem has two stages. We first set up some constants and parameters as follows. Afterwards, the statements i and ii of the theorem will be proved by a series of lemmas.

Let $\beta = 0.5$, and $c = e^{-\frac{c^2}{3}}$.

Select constant ϵ such that

$$4\alpha + 5\epsilon < 1 - \frac{1}{|\Sigma|}. \tag{1}$$

Select constants (as large as possible) $\sigma_0 = \sigma_2 > 0$, $\sigma_4 > 0$, and $\sigma_1 > 0$ such that

$$(1 - (\alpha + \sigma_1 + \sigma_2)) > \frac{1}{|\Sigma|} + \sigma_1 + \sigma_2, \text{ and} \tag{2}$$

$$(\sigma_4 + \sigma_0) \leq \sigma_1. \tag{3}$$

Let constants $\sigma_3 > 0$ and $c_1 = \frac{24}{\epsilon^2}$. Let $h \geq \frac{c_1 \log n}{1 - 2\sigma_3}$ and $z = \frac{h}{4}$. From the above selections, we have that $c^z \leq n^{-2}e^{-2\sigma_3 h}$. It is easy to see that

$$\frac{c^z}{1 - c} + n^2 c^z + 4n^2 c^z < e^{-\sigma_3 h} \text{ for all sufficiently large } n. \tag{4}$$

Let constant $\sigma_4 > 0$ be selected so that $n \frac{2c^z}{1 - c} + n \frac{2c^z}{1 - c} \leq \sigma_4$ for all sufficiently large n .

It is easy to see the existence of those constants ϵ, c_0 and c_1 to satisfy inequalities (II) to (4). The existence of ϵ for inequality (II) is due to the condition $\alpha < \frac{1}{4}(1 - \frac{1}{|\Sigma|})$ of the theorem.

Let $\mu_0 = 1 - (\alpha + \sigma_1 + \sigma_2)$, and $\mu_1 = (1 - (\sigma_1 + \sigma_2))$.

4.1 Phase 1 of Algorithm Discover-Motif — Align-Sequences

Align-Sequences uses a subroutine Find-a-Piece-of-Motif(S, S') to find a similar subsequence of length at least $4z = 4 \lceil c_1 \log n \rceil$ between S_1 and S_2 . The subsequence returned by Find-a-Piece-of-Motif(S, S') will then be used to align the motif regions among input sequences S_1, S_2, \dots, S_k as shown in Figure 2.

Find-a-Piece-of-Motif(S, S')

Input: two $\Theta_\alpha(n, G)$ -sequences S and S' ;

Steps:

let guess-motif-length = $|S|$;

repeat

 jump-step = guess-motif-length/4;

$i = 1$;

 repeat

 let $U = U_1U_2 = S[i, i + 4z - 1]$ with $|U_1| = |U_2| = |U|/2 = 2z$;

 let $t = 1$;

 repeat

 let $V = V_1V_2 = S'[t, t + 4z - 1]$ with $|V_1| = |V_2| = |V|/2 = 2z$;

 if $\text{diff}[U_1, V_1] \leq 2(\alpha + \epsilon)$ and $\text{diff}[U_2, V_2] \leq 2(\alpha + \epsilon)$

 then return U and stop;

 let $t = t + 1$;

 until $t > |S'| - 4z + 1$

$i = i + \text{jump-step}$;

 until $i > |S| - 4z + 1$;

 guess-motif-length = guess-motif-length/2;

until jump-step $< 2z$;

return “failure”;

End of function Find-a-Piece-of-Motif

Algorithm Align-Sequences(S_1, \dots, S_k) calls subroutine Find-a-Piece-of-Motif(S_1, S_2) to obtain a similar region U of length $4z$ between S_1 and S_2 . We will show that the subsequence U is most likely from the motif region of S_1 . Align-Sequences then aligns all the sequences S_1, S_2, \dots, S_k by putting those subsequences of S_2, S_3, \dots, S_k that are similar to U into the same column regions as illustrated in Figure 2.

Align-Sequences(S_1, \dots, S_k)

Input: k $\Theta_\alpha(n, G)$ -sequences S_1, \dots, S_k for some unknown motif G to be detected.

Steps:

let $U = \text{Find-a-Piece-of-Motif}(S_1, S_2)$;

partition U into $U = U'_1U'_2U'_3U'_4$ with $|U'_1| = |U'_2| = |U'_3| = |U'_4| = z$;

for each S_r with $2 \leq r \leq k$

 find a subsequence $Y_r = X_1X_2X_3X_4$ of S_r such that $\text{diff}[U'_i, X_i] \leq 2\alpha + \epsilon$

 for $i = 1, 2, 3, 4$;

let A be a $k \times 3n$ array (with k rows and $3n$ columns);

put S_1 into the middle of the first row of A (i.e., let S_1 be arranged at positions $A[1][n+1], A[1][n+2], \dots, A[1][n+|S_1|]$);
 for each S_i with $2 \leq i \leq k$
 put S_i in row i so that $U[1]$ and $Y_i[1]$ are in the same column (which implies $U[j]$ and $Y_i[j]$ are also in the same column for $j = 1, 2, \dots, |U|$);
 return A ;
 End of function Align-Sequences

4.2 Phase 2 of Algorithm Discover-Motif — Recover-Motif

Recover-Motif will detect the columns that contain the motif from a σ_1 -error alignment. Each character of the motif is recovered by voting on the characters in the same column. Each column that does not contain motif character does not have a character that appears as frequently as in the motif region.

Let $\text{Maj}(A, j)$ denote the character that appears the largest number of times in the column j of A . For a character $a \in \Sigma$, $\text{Occur}(a, j, A)$ denotes the number of times that a appears in the column j of A .

The subroutine Recover-Motif uses a subroutine called Detect-Motif-Boundary (A) that finds a pair of column indices j_L and j_R such that most copies of the motif in the input sequences will be located from column $j_L + 1$ to column $j_R - 1$. We will show that if the alignment has a suitably small error, then Detect-Motif-Boundary returns the boundaries of the motif region with high probability if the number of the input sequences is reasonably large.

Detect-Motif-Boundary(A)

Input: a $k \times 3n$ matrix A that holds k aligned sequences S_1, \dots, S_k ;

Steps:

Case 1: $2k > h$.

Select the leftmost column j_L in A such that $\text{Occur}(\text{Maj}(A, j_L + 1), j_L + 1, A) \geq \mu_0 k$;

Select the rightmost column j_R in A such that $\text{Occur}(\text{Maj}(A, j_R - 1), j_R - 1, A) \geq \mu_0 k$;

Case 2: $2k \leq h$.

Select the leftmost column j_L in A such that

(L1) $\text{diff}(A[1][j_L, j_L + i], A[2][j_L, j_L + i]) \leq 2\alpha + \sigma_2$ for all $i = 2^j$

(j is an integer ≥ 0) with $k \leq i \leq h$, and

(L2) $\text{Occur}(\text{Maj}(A, j_L + i), j_L + i, A) \geq \mu_0 k$ for every i with $i = 2^j \leq k$ for some non-negative integer $j \geq 0$;

Select the rightmost column j_R in A such that

(R1) $\text{diff}(A[1][j_R, j_R - i], A[2][j_R, j_R - i]) \leq 2\alpha + \sigma_2$ for all $i = 2^j$

(j is an integer ≥ 0) with $1 \leq i \leq \min(k, h)$, and

(R2) $\text{Occur}(\text{Maj}(A, j_R - i), j_R - i, A) \geq \mu_0 k$ for every i with $i = 2^j \leq k$ for some non-negative integer $j \geq 0$;

return (j_L, j_R) ;

End of function Detect-Motif-Boundary

Recover-Motif(A) recovers the motif by voting on the character in each column of A in the motif region, which is between the boundaries returned from the function Detect-Motif-Boundary.

Recover-Motif(A)

Input: $k \times 3n$ matrix A that holds k aligned sequences S_1, \dots, S_k ;
 let $(j_L, j_R) = \text{Detect-Motif-Boundary}(A)$;
 let g'_s be the character that appears the largest number of times in column $j_L + s$ of A for $0 < s < j_R - j_L$;
 return $g'_1 \cdots g'_h$ as the motif G ;
 End of function Recover-Motif

4.3 Algorithm Discover-Motif

Now we can describe our main algorithm Discover-Motif by using the subroutines already described in Sections 4.1 and 4.2.

Discover-Motif (S_1, \dots, S_k)

Input: k $\Theta_\alpha(n, G)$ -sequences S_1, \dots, S_k for some unknown motif G .
 Steps: let $A = \text{Align-Sequences}(S_1, \dots, S_k)$;
 return Recover-Motif(A);
 End of Discover-Motif

The proof about the correctness of the algorithm is too long to be included.

Theorem 3. *Assume that $\alpha < \frac{1}{4}(1 - \frac{1}{|\Sigma|})$ and x are constants. Then there exist constants $\delta_1, \delta_2, \epsilon > 0$ such that given an input of $\Theta_\alpha(n, G)$ -sequence sequences S_1, \dots, S_k with $k > c_0 \log n$, Discover-Motif returns the motif G with probability at least $1 - \frac{1}{2^x}$ for all $G \in \Sigma^\rho - \Psi_{\rho, h, \epsilon}(\Sigma)$ and runs in $O(n^2)$ time, where ρ is the length of the motif G , $\rho \geq \delta_1 \log n$, and $\rho \geq 4h \geq \delta_2 \log n$.*

Proof. By Theorem 2, there exist constants $\delta_1, \delta_2, \epsilon > 0$ such that Align-Sequences and Recover-Motif fail each with probability at most $\frac{1}{2^{x+1}}$.

Acknowledgements. We thank Miklós Csürös and Manan Sanghi for helpful discussions. Bin Fu is supported in part by National Science Foundation Early Career Award 0845376. Ming-Yang Kao is supported in part by National Science Foundation Grant CNS-0627751. Lusheng Wang is fully supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. CityU 1196/03E).

References

1. Chin, F., Leung, H.: Voting algorithms for discovering long motifs. In: Proceedings of the 3rd Asia-Pacific Bioinformatics Conference, pp. 261–272 (2005)
2. Dopazo, J., Rodríguez, A., Sáiz, J.C., Sobrino, F.: Design of primers for PCR amplification of highly variable genomes. Computer Applications in the Biosciences 9, 123–125 (1993)

3. Frances, M., Litman, A.: On covering problems of codes. *Theoretical Computer Science* 30, 113–119 (1997)
4. Fu, B., Kao, M.-Y., Wang, L.: Efficient algorithms for model-based motif discovery from multiple sequences. In: Agrawal, M., Du, D.-Z., Duan, Z., Li, A. (eds.) TAMC 2008. LNCS, vol. 4978, pp. 234–245. Springer, Heidelberg (2008)
5. Gąsieniec, L., Jansson, J., Lingas, A.: Efficient approximation algorithms for the Hamming center problem. In: Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. S905–S906 (1999)
6. Gusfield, D.: *Algorithms on Strings, Trees, and Sequences*. Cambridge University Press, Cambridge (1997)
7. Hertz, G., Stormo, G.: Identification of consensus patterns in unaligned DNA and protein sequences: a large-deviation statistical basis for penalizing gaps. In: Proceedings of the 3rd International Conference on Bioinformatics and Genome Research, pp. 201–216 (1995)
8. Keich, U., Pevzner, P.: Finding motifs in the twilight zone. *Bioinformatics* 18, 1374–1381 (2002)
9. Keich, U., Pevzner, P.: Subtle motifs: defining the limits of motif finding algorithms. *Bioinformatics* 18, 1382–1390 (2002)
10. Lanctot, J.K., Li, M., Ma, B., Wang, L., Zhang, L.: Distinguishing string selection problems. In: Proceedings of the 10th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 633–642 (1999)
11. Lawrence, C., Reilly, A.: An expectation maximization (EM) algorithm for the identification and characterization of common sites in unaligned biopolymer sequences. *Proteins* 7, 41–51 (1990)
12. Li, M., Ma, B., Wang, L.: Finding similar regions in many strings. In: Proceedings of the 31st Annual ACM Symposium on Theory of Computing, pp. 473–482 (1999)
13. Li, M., Ma, B., Wang, L.: On the closest string and substring problems. *Journal of the ACM* 49(2), 157–171 (2002)
14. Lucas, K., Busch, M., Mossinger, S., Thompson, J.: An improved microcomputer program for finding gene- or gene family-specific oligonucleotides suitable as primers for polymerase chain reactions or as probes. *Computer Applications in the Biosciences* 7, 525–529 (1991)
15. Motwani, R., Raghavan, P.: *Randomized Algorithms*. Cambridge University Press, Cambridge (2000)
16. Pevzner, P., Sze, S.: Combinatorial approaches to finding subtle signals in DNA sequences. In: Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology, pp. 269–278 (2000)
17. Proutski, V., Holme, E.C.: Primer master: a new program for the design and analysis of PCR primers. *Computer Applications in the Biosciences* 12, 253–255 (1996)
18. Stormo, G.: Consensus patterns in DNA. In: Doolittle, R.F. (ed.) *Molecular evolution: computer analysis of protein and nucleic acid sequences*. *Methods in Enzymology*, vol. 183, pp. 211–221 (1990)
19. Stormo, G., Hartzell III, G.: Identifying protein-binding sites from unaligned DNA fragments. In: Proceedings of the National Academy of Sciences of the United States of America, vol. 88, pp. 5699–5703 (1991)
20. Wang, L., Dong, L.: Randomized algorithms for motif detection. *Journal of Bioinformatics and Computational Biology* 3(5), 1039–1052 (2005)

A Complete Characterisation of the Linear Clique-Width of Path Powers^{*}

Pinar Heggernes¹, Daniel Meister¹, and Charis Papadopoulos²

¹ Department of Informatics, University of Bergen, Norway
pinar.heggernes@ii.uib.no, daniel.meister@ii.uib.no

² Department of Mathematics, University of Ioannina, Greece
charis@cs.uoi.gr

Abstract. A k -path power is the k -power graph of a simple path of arbitrary length. Path powers form a non-trivial subclass of proper interval graphs. Their clique-width is not bounded by a constant, and no polynomial-time algorithm is known for computing their clique-width or linear clique-width. We show that k -path powers above a certain size have linear clique-width exactly $k + 2$, providing the first complete characterisation of the linear clique-width of a graph class of unbounded clique-width. Our characterisation results in a simple linear-time algorithm for computing the linear clique-width of all path powers.

1 Introduction

Clique-width is a graph parameter that describes the structure of a graph and its behaviour with respect to hard problems [6]. Many NP-hard graph problems become solvable in polynomial time on graphs whose clique-width is bounded by a constant [21, 27]. If the problem, in addition, is expressible in a certain type of monadic second order logic, it becomes fixed parameter tractable when parameterised by clique-width [7]. Clique-width can be viewed as a generalisation of the more widely studied parameter treewidth, since there are graphs of bounded clique-width but unbounded treewidth (e.g., complete graphs), whereas graphs of bounded treewidth have bounded clique-width [9]. As pathwidth is a restriction on treewidth, *linear clique-width* is a restriction on clique-width, and hence graphs of bounded clique-width might have unbounded linear clique-width (e.g., cographs [16]). Both clique-width and linear clique-width are NP-hard to compute [11]. These two closely related graph parameters have received much attention recently, and the interest in them is increasing [4, 7, 9, 13, 8, 11, 10, 23, 24, 25, 2, 5, 16, 3, 11, 14, 15, 22, 20, 17, 12].

In this paper, we give a complete characterisation of the linear clique-width of path powers, which form a subclass of proper interval graphs. Hereditary subclasses of proper interval graphs have bounded clique-width [22], however path powers are not hereditary, and they have unbounded clique-width [13] and thus unbounded linear clique-width. This is the first graph class of unbounded clique-width whose linear clique-width is hereby completely characterised. More

^{*} This work is supported by the Research Council of Norway.

precisely, we show that k -path powers above a certain size have linear clique-width exactly $k + 2$. A k -path power is the k -power graph of a simple path. We also characterise the linear clique-width of smaller k -path powers. Our characterisation results in a simple linear-time algorithm for computing the linear clique-width of path powers, making this the first graph class on which clique-width or linear clique-width is unbounded, and linear clique-width can be computed in polynomial time. In addition, we give a characterisation of the linear clique-width of path powers through forbidden induced subgraphs. The main difficulty to overcome in obtaining these results has been to prove a tight lower bound on the linear clique-width of path powers.

To review related results, we can mention that graphs of clique-width at most 2 [9] and at most 3 [4] can be recognised in polynomial time. Also graphs of linear clique-width at most 2 [14] and at most 3 [20] can be recognised in polynomial time. Several graph classes have been studied with respect to whether or not their clique-width is bounded by a constant [1, 2, 3, 10, 13, 17, 22, 23, 24]. For specific graph classes of unbounded clique-width and thus unbounded linear clique-width, little is known on the computation of their clique-width or linear clique-width. So far the only result that computes either of these parameters exactly is given by Golubic and Rotics [13], who show that a $k \times k$ grid has clique-width $k + 1$. (Notice that for fixed k , there are infinitely many k -path powers, but only one $k \times k$ grid.) Other than this, mainly some upper [11, 17] and lower [13, 5] bounds have been given some of which are mentioned below. Typical for lower bounds is that they are not tight, and therefore they do not lead to exact computation of the clique-width or the linear clique-width efficiently. For lower bounds, Golubic and Rotics gave lower bounds on the clique-width of some subclasses of proper interval graphs and permutation graphs [13], and Corniel and Rotics showed an exponential gap between clique-width and treewidth [5].

Specifically for path powers, the results of Gurski and Wanke on the linear clique-width of power graphs imply that the linear clique-width of a k -path power is at most $(k + 1)^2$ [17]. Fellows et al. showed that the linear clique-width of a graph is bounded by its pathwidth plus 2 [11], which gives $k + 2$ as an upper bound on the linear clique-width of k -path powers. For lower bounds, Golubic and Rotics showed that the clique-width and thus the linear clique-width of a k -path power on $(k + 1)^2$ vertices is *at least* $k + 1$ [13]. The authors conjecture that the clique-width of k -path powers on $(k + 1)^2$ vertices is exactly $k + 2$ [13]. This conjecture is still open. The same upper and lower bounds are still the best known bounds also on the linear clique-width of k -path powers on $(k + 1)^2$ vertices. In this paper, we prove the conjecture to be true for linear clique-width.

The results that we present in this paper contribute to better understanding of linear clique-width and clique-width. The knowledge on these graph parameters is still limited, and there is no general intuition on what makes a graph structurally more complicated (larger clique-width) than other graphs. To prove the lower bound $k + 2$ on the above mentioned k -path powers (in Section 5), the technique we apply is through identifying *maximal* k -path powers of linear clique-width *at most* $k + 1$ (in Section 4).

2 Basic Definitions, Notation, and Linear Clique-Width

We consider undirected finite graphs with no loops or multiple edges. For a graph $G = (V, E)$, we denote its vertex and edge set by $V(G) = V$ and $E(G) = E$, respectively. Two vertices u and v of G are called *adjacent* if $uv \in E$; if $uv \notin E$ then u and v are *non-adjacent*. A *path* in G is a sequence of vertices (v_1, v_2, \dots, v_l) such that $v_i v_{i+1} \in E$ for $1 \leq i \leq l-1$. For a vertex set $S \subseteq V$, the *subgraph of G induced by S* is denoted by $G[S]$. Moreover, we denote by $G-v$ the graph $G[V \setminus \{v\}]$. The *neighbourhood* of a vertex x in G is $N_G(x) = \{v \mid xv \in E\}$ and its *degree* is $|N_G(x)|$. For two vertices x and y , if another vertex z is adjacent to exactly one of them then we say that z *distinguishes* x and y .

Let G and H be two vertex-disjoint graphs. The *disjoint union* of G and H is the graph with vertex set $V(G) \cup V(H)$ and edge set $E(G) \cup E(H)$. The notion of clique-width was first introduced in [6]. The *clique-width* of a graph G is the minimum number of labels needed to construct G using the following four operations: create new graph on a single vertex with label i , disjoint union, change all labels i to j , add all edges between vertices with label i and vertices with label j where $i \neq j$. The *linear clique-width* of a graph, denoted by $\text{lcwd}(G)$, is defined by the above operations with the restriction that at least one of the operands of the disjoint union operation must be a graph on a single vertex [16, 25]. This results in a linear structure, and linear clique-width can be viewed as a graph layout problem [15, 19].

A *layout* for a graph $G = (V, E)$ is a linear ordering of its vertices, usually defined as a bijective mapping from the set $\{1, \dots, |V|\}$ to V . For $A \subseteq V$, a *group* in A is a maximal set of vertices with the same neighbourhood in $V \setminus A$. Note that two groups in A are either equal or disjoint, implying that the group relation defines a partition of A . By $\nu_G(A)$, we denote the number of groups in A . Let β be a layout for G . Let x be a vertex of G and let p be the position of x in β , i.e., $p = \beta^{-1}(x)$. The *set of vertices to the left of x with respect to β* is $\{\beta(1), \dots, \beta(p-1)\}$ and denoted as $L_\beta(x)$, and the *set of vertices to the right of x with respect to β* is $\{\beta(p+1), \dots, \beta(|V|)\}$ and denoted as $R_\beta(x)$. We write $L_\beta[x]$ and $R_\beta[x]$ if x is included. Function ad_β is a $\{0, 1\}$ -valued function on the vertex set of G with respect to β . Given a vertex x of G , if one of the following conditions is satisfied then $\text{ad}_\beta(x) = 1$; if none of the conditions is satisfied then $\text{ad}_\beta(x) = 0$:

- (1) all (other) vertices in the group in $L_\beta[x]$ that contains x are neighbours of x
- (2) $\{x\}$ is not a group in $L_\beta[x]$, and there are a non-neighbour y of x in the group of $L_\beta[x]$ containing x and a neighbour z of x in $L_\beta(x)$ such that y and z are non-adjacent

The *groupwidth of a graph G with respect to a layout β for G* , denoted as $\text{gw}(G, \beta)$, is the smallest number k such that $\nu_G(L_\beta(x)) + \text{ad}_\beta(x) \leq k$ for all $x \in V(G)$. The *groupwidth* of a graph G , denoted as $\text{gw}(G)$, is the smallest number k such that there is a layout β for G satisfying $\text{gw}(G, \beta) \leq k$. Based on the function ad , the following result was proved in [19]. Analogous results already existed [15, 25] using different formulations of function ad .

Theorem 1 ([15, 25, 19]). *For every graph G , $lcwd(G) = gw(G)$.*

For a given graph G , the k -power graph of G is the graph that has the same vertex set as G such that two vertices are adjacent if and only if the distance (length of a shortest path) between them is at most k in G . For a given $l \geq 1$, P_l is the graph with vertex set $\{x_1, x_2, \dots, x_l\}$ and edge set $\{x_1x_2, x_2x_3, \dots, x_{l-1}x_l\}$. A k -path power is a graph that is the k -power graph of P_l for some l . Notice that the k -power graph of P_l for any $k \geq l - 1$ is a complete graph. Observe that for a k -path power that is not complete, a largest clique contains exactly $k + 1$ vertices. A path power is a k -path power for some k . For a path power, a vertex of smallest degree is called *endvertex*. A path power that is not complete has exactly two endvertices, that are non-adjacent.

Lemma 1. *Let P be a path power and let β be a layout for P . If $ad_\beta(x) = 0$ for a vertex x of P then x is an endvertex of P .*

3 Groups in Induced Subgraphs of Path Powers

The linear clique-width bounds that we present in this paper are all proved by applying Theorem 1. The main technique is to count groups in subgraphs. As a main tool, we use a representation of path powers that arranges vertices into rows and columns of a 2-dimensional array. Let G be a graph. A *bubble model* for G is a 2-dimensional structure $\mathcal{B} = \langle B_{i,j} \rangle_{1 \leq j \leq s, 1 \leq i \leq r_j}$ such that the following conditions are satisfied:

- for $1 \leq j \leq k$ and $1 \leq i \leq r_j$, $B_{i,j}$ is a (possibly empty) set of vertices of G
- the sets $B_{1,1}, \dots, B_{r_k,k}$ are pairwise disjoint and cover $V(G)$
- two vertices u, v of G are adjacent if and only if there are $1 \leq j \leq j' \leq s$ and $1 \leq i \leq r_j$ and $1 \leq i' \leq r_{j'}$ such that $u, v \in B_{i,j} \cup B_{i',j'}$ and (a) $j = j'$ or (b) $j + 1 = j'$ and $i > i'$.

A similar structure is given by Golumbic and Rotics [13]. The sets $B_{i,j}$ are called *bubbles*. If every bubble $B_{i,j}$ contains exactly one vertex, we also write $\langle b_{i,j} \rangle_{1 \leq j \leq s, 1 \leq i \leq r_j}$. A graph is a proper interval graph if and only if it has a bubble model [18]. For $1 \leq j \leq s$, we combine the sets $B_{1,j}, \dots, B_{r_j,j}$ to the j th column, also denoted as \mathcal{B}_j . We say that \mathcal{B} is a *bubble model on s columns and b rows* if $s = a$ and $r_1 = \dots = r_{s-1} = \max\{r_1, \dots, r_s\} = b$.

Theorem 2. *Let $k \geq 1$. A graph G is a k -path power if and only if there is $s \geq 1$ such that G has a bubble model on s columns and $k + 1$ rows and all bubbles contain exactly one vertex.*

Proof. Due to space restrictions, we only prove one implication. Let G be the k -power graph of P_l . We rename the vertices of the path as follows. For $1 \leq i \leq l$, let $b_{b,a} =_{\text{def}} x_i$ where a and b are such that $i = a(k + 1) + b$ and $1 \leq b \leq k + 1$. Let s be smallest such that $l \leq s(k + 1)$, and let $r_1 =_{\text{def}} \dots =_{\text{def}} r_{s-1} =_{\text{def}} k + 1$ and $r_s =_{\text{def}} n - (s - 1)(k + 1)$. Let $\mathcal{B} =_{\text{def}} \langle b_{i,j} \rangle_{1 \leq j \leq s, 1 \leq i \leq r_j}$. Let x_i and $x_{i'}$

be vertices of G with $i < i'$. Let a, a', b, b' be such that $1 \leq b, b' \leq k + 1$ and $i = a(k + 1) + b$ and $i' = a'(k + 1) + b'$. Clearly, if $a = a'$ then $b' - b \leq k$ and therefore $i' - i \leq k$. If $a < a'$ then $i' - i \leq k$ if and only if $b > b'$. Hence, \mathcal{B} is a bubble model for G . And by construction, \mathcal{B} is a bubble model on $k + 1$ rows and all bubbles contain exactly one vertex.

We call the bubble model of a path power that is constructed in the proof of Theorem 2 *canonical*. Observe that the proof of Theorem 2 gives a simple linear-time algorithm for constructing a canonical bubble model for a given path power.

In our lower bound proofs, the main task is to determine the number of groups. Let G be a graph with bubble model $\mathcal{B} = \langle B_{i,j} \rangle_{1 \leq j \leq s, 1 \leq i \leq r_j}$. Let $A \subseteq V(G)$ and let $1 \leq \hat{j} \leq s$. The \hat{j} -boundary of $\mathcal{B}[A]$ is the set $\Phi_{\hat{j}}(\mathcal{B}[A])$ of pairs (i, t_i) that satisfy one of the following conditions:

- $t_i = \hat{j}$ and $i < r_{\hat{j}}$ and $B_{i,t_i} \subseteq A$ and $B_{i',\hat{j}} \not\subseteq A$ for all $i < i' \leq r_{\hat{j}}$
- $t_i < \hat{j}$ and $1 \leq i \leq \min\{r_{t_i}, \dots, r_{\hat{j}}\}$ and $B_{i,t_i} \subseteq A$ and $B_{i,j} \not\subseteq A$ for all $t_i < j \leq \hat{j}$.

Lemma 2. *Let G be a graph with bubble model $\mathcal{B} = \langle B_{i,j} \rangle_{1 \leq j \leq s, 1 \leq i \leq r_j}$ on $s \geq 2$ columns and $l \geq 2$ rows. Let $A \subseteq V(G)$ and let $1 \leq \hat{j} \leq s$. The bubbles in $\Phi_{\hat{j}}(\mathcal{B}[A])$ appear in pairwise different groups in A .*

4 Maximal k -Path Powers of Linear Clique-Width $k + 1$

In the next section we will show that the linear clique-width of a k -path power containing $k(k + 1) + 2$ vertices is at least $k + 2$. In fact they will turn out to be the smallest k -path powers of maximum linear clique-width. This result is achieved by showing that a k -path power containing $k(k + 1) + 1$ vertices has layouts of groupwidth at most $k + 1$ of only very restricted type. This is exactly what we prove in this section, through a series of results. More precisely, we concentrate on the beginning of a possible layout of groupwidth at most $k + 1$, identify the earliest point where the maximum group number is reached, and we show that the two vertices on either side of this point are uniquely defined, hence the restriction on the layout. This restriction in the layouts is used in the next section to show that it is not possible to extend such a k -path power by even a single vertex without increasing the linear clique-width.

The main result of this section is given in Lemmas 7 and 8. To make the statements of the results shorter, we avoid repeating the following definitions. Throughout this section, let

- P be a k -path power on $k(k + 1) + 1$ vertices, with $k \geq 3$,
- β be a layout for P such that $\text{gw}(P, \beta) \leq k + 1$,
- $\mathcal{B} = \langle b_{i,j} \rangle_{1 \leq j \leq k+1, 1 \leq i \leq r_j}$ be a canonical bubble model for P (Theorem 2) such that $b_{1,1} \prec_{\beta} b_{1,k+1}$.

Note that β indeed exists, which is shown later (Lemma 10). Note also that $r_1 = \dots = r_k = k + 1$. For $A \subseteq V(P)$ and $1 \leq j \leq k$, we say that column \mathcal{B}_j

is full with respect to A if $b_{1,j}, \dots, b_{k+1,j} \in A$. Let x_f be the leftmost vertex of P with respect to β such that there is an index j_f between 1 and k with \mathcal{B}_{j_f} full with respect to $L_\beta[x_f]$. By the choice of x_f , j_f is uniquely defined. Let $L_f^- =_{\text{def}} L_\beta(x_f)$ and $L_f =_{\text{def}} L_\beta[x_f]$. Denote by $x_{f+1}, x_{f+2}, x_{f+3}$ the three vertices that follow x_f in β .

By Lemma 3 and the fact that non-complete path powers have at most two endvertices, there are at most two vertices for which function ad can have value 0. We can be even more specific.

Lemma 3. *If $\text{ad}_\beta(b_{1,1}) = 0$ then $b_{1,1}$ and $b_{1,2}$ are in the same group in $L_\beta[b_{1,1}]$. If $\text{ad}_\beta(b_{1,k+1}) = 0$ then $b_{1,k+1}$ and $b_{1,k}$ are in the same group in $L_\beta[b_{1,k+1}]$.*

Lemma 4. *Let $u \in V(P)$. Let K be a group in $L_\beta[u]$. Let $b_{i,j}$ and $b_{i',j'}$ be two vertices in K . Then, $N_P(b_{i,j}) \Delta N_P(b_{i',j'}) \subseteq L_\beta[u]$. In particular, if $|j - j'| \geq 2$ then \mathcal{B}_j and $\mathcal{B}_{j'}$ are full with respect to $L_\beta[u]$.*

From Lemma 4 it follows that a group in L_f can contain vertices from only the same column or from two consecutive columns, since exactly one column is full with respect to L_f .

Lemma 5. *There is no $1 \leq j \leq k$ such that $b_{1,j}, \dots, b_{k+1,j} \notin L_f$.*

Let $A \subseteq V(P)$. For every $1 \leq j \leq k$, we denote by $g_j(A)$ the number of groups in A that contain a vertex from column \mathcal{B}_j but not from any of the columns $\mathcal{B}_{j+1}, \dots, \mathcal{B}_{k+1}$. Note that if there is at most one column that is full with respect to A then it suffices to forbid vertices from \mathcal{B}_{j+1} due to Lemma 4.

Lemma 6. *Let $u \in L_f^-$ be such that for every $1 \leq j \leq k$, there is $1 \leq i \leq k + 1$ with $b_{i,j} \in L_\beta[u]$. Then, $g_1(L_\beta[u]), \dots, g_{k-1}(L_\beta[u]) \geq 1$.*

Lemma 7. *The vertices $b_{1,k}$ and $b_{1,k+1}$ are not in L_f .*

Proof. By definition of x_f , we know that $b_{1,k+1} \neq x_f$. It can be shown that $b_{1,k+1} \notin L_f$. Suppose for a contradiction that $b_{1,k} \in L_f$. Suppose that $j_f = k$. Let $1 \leq i \leq k + 1$ be such that $x_f = b_{i,k}$. We consider L_f^- . With Lemma 6 and $b_{1,k+1} \notin L_f^-$, $g_1(L_f^-), \dots, g_k(L_f^-) \geq 1$. If there is $1 \leq j \leq k$ such that $g_j(L_f^-) \geq 2$ then $\nu_P(L_f^-) \geq k + 1$, and since $b_{i,k}$ is not endvertex of P , $\text{gw}(P, \beta) > k + 1$. Therefore, $g_1(L_f^-) = \dots = g_k(L_f^-) = 1$. If $i \geq 2$ then L_f^- contains $b_{1,k}$ and another vertex from \mathcal{B}_k . Since they are distinguished by $b_{1,k+1}$, this gives $g_k(L_f^-) \geq 2$, which is a contradiction. Thus, $i = 1$, i.e., $x_f = b_{1,k}$. In particular, $b_{1,k} \notin L_f^-$. As an auxiliary result, the following can be shown by induction. For every $2 \leq j \leq k$:

- (1) $b_{k+2-j,j}, \dots, b_{k+1,j} \in L_f^-$ and $b_{1,j} \notin L_f^-$
- (2) the vertices from \mathcal{B}_j in L_f^- form a group in L_f^- .

Observe that $b_{1,k}$ is in a singleton group in L_f and every group in L_f^- is a group in L_f . Thus, $\nu_P(L_f) = k + 1$ and $g_1(L_f) = \dots = g_{k-1}(L_f) = 1$ and $g_k(L_f) = 2$.

Then, $\text{ad}_\beta(x_{f+1}) = 0$, i.e., x_{f+1} is an endvertex of P . Since $x_{f+1} = b_{1,1}$ yields a contradiction, $x_{f+1} = b_{1,k+1}$, and therefore, $b_{1,1} \in L_f$. Furthermore, $b_{1,1}$ is the only vertex from \mathcal{B}_1 in L_f . We show that $\nu_{P-b_{1,k+1}}(L_f) = k + 1$. Consider $\mathcal{B}[L_f \setminus \{b_{1,1}\}]$. Let \mathcal{B}' be defined as in the proof of Lemma 5. We apply Lemma 2 to $\mathcal{B}'[L_f \setminus \{b_{1,1}\}]$ and its $(k + 1)$ -boundary: there are (at least) k boundary vertices. Suppose that $b_{1,1}$ is in the same group as a boundary vertex in L_f . Since no other vertex from \mathcal{B}_1 is in L_f , $b_{1,1}$ can be in group only with $b_{1,2}$. This, however, contradicts $b_{1,2} \notin L_f$ due to the auxiliary result. Hence, $b_{1,1}$ is not in the same group as any vertex from the boundary, and therefore $\nu_{P-b_{1,k+1}}(L_f) = k + 1$. We obtain $\nu_P(L_\beta[x_{f+1}]) = k + 1$, and since x_{f+2} is not an endvertex of P , we conclude a contradiction to the groupwidth assumption for β , and therefore $j_f < k$.

It can be shown that there is no $2 \leq i \leq k + 1$ with $b_{i,k} \in L_f^-$. The k -boundary of $\mathcal{B}[L_f]$ contains k vertices, and due to Lemma 2, they are in pairwise different groups in L_f . Since no vertex in columns $\mathcal{B}_1, \dots, \mathcal{B}_{k-1}$ is adjacent to $b_{k+1,k}$, which is not contained in L_f , $b_{1,k}$ is vertex in a singleton group, and therefore $\nu_P(L_f) \geq k + 1$ and $\text{ad}_\beta(x_{f+1}) = 0$. According to Lemma 1, x_{f+1} is endvertex of P . If $x_{f+1} = b_{1,1}$ then $j_f > 1$, and no vertex from \mathcal{B}_1 is in the same group as a vertex from another column in L_f because of $b_{1,1}$. Then, the above arguments show that L_f has at least $k + 2 > k + 1$ groups, which is a contradiction to the groupwidth assumption for β . Thus, $x_{f+1} = b_{1,k+1}$. Then, $\nu_P(L_\beta[x_{f+1}]) \geq k + 1$, since no vertex in L_f is adjacent to $b_{1,k+1}$. However, x_{f+2} is no endvertex of P , which yields $\nu_P(L_\beta(x_{f+2})) + \text{ad}_\beta(x_{f+2}) > k + 1$, a contradiction to the groupwidth assumption for β . Hence, the assumption $b_{1,k} \in L_f$ is false, and we conclude the lemma.

Lemma 8. *The following holds for layout β :*

- $\nu_P(L_f) = k + 1$ and $x_f = b_{1,2}$ and $x_{f+1} = b_{1,1}$
- $b_{3,1}, \dots, b_{k+1,1} \in L_f$ and $b_{2,k}, b_{3,k} \in L_f$
- the vertices from \mathcal{B}_1 in L_f are in the same group and the vertices from \mathcal{B}_k in L_f are in the same group in L_f .

5 The Linear Clique-Width of Path Powers

In this section, we are finally ready to give a complete characterisation of the linear clique-width of path powers of all sizes. We start with the previously mentioned lower bound.

Lemma 9. *Let G be a k -path power on $k(k + 1) + 2$ vertices, with $k \geq 1$. Then, $\text{lcwd}(G) \geq k + 2$.*

Proof. For $k = 1$, G is a 1-path power on four vertices, i.e., $G = P_4$. It holds that $\text{lcwd}(P_4) = 3$. For $k = 2$, G is a 2-path power on eight vertices. It can be checked that $\text{lcwd}(G) = 4$. So, let $k \geq 3$. Suppose for a contradiction that there is a layout β for G such that $\text{gw}(G, \beta) \leq k + 1$. Let a be an endvertex of G . Then,

$G-a$ is a k -path power on $k(k+1)+1$ vertices. Let β' be obtained from β by deleting a . Then, $\text{gw}(G-a, \beta') \leq k+1$, and the results of Section 4 can be applied to $G-a$ and β' . Let $\mathcal{B} = \langle b_{i,j} \mid 1 \leq j \leq k+1, 1 \leq i \leq r_i \rangle$ be a canonical bubble model for $G-a$ such that $b_{1,1} \prec_{\beta'} b_{1,k+1}$. Let x_f and L_f and L_f^- for $G-a$ and β' be defined as in Section 4. Due to Lemma 8, $b_{k,1}, b_{k+1,1} \in L_f$, and $b_{2,k}, b_{3,k} \in L_f$, and $b_{k,1}$ and $b_{k+1,1}$ are in the same group in L_f , and $b_{2,k}$ and $b_{3,k}$ are in the same group in L_f . Furthermore, $b_{1,1}, b_{1,2} \notin L_f^-$ and $b_{1,k}, b_{1,k+1} \notin L_f$ (Lemma 7). By the choice of a as an endvertex of G , a is adjacent to $b_{k,1}$ and non-adjacent to $b_{k+1,1}$ or a is adjacent to $b_{3,k}$ and non-adjacent to $b_{2,k}$. If $x_f \prec_{\beta} a$ then a distinguishes $b_{k,1}$ and $b_{k+1,1}$ in the former case, and $b_{2,k}$ and $b_{3,k}$ in the latter case. With $\nu_{G-a}(L_f) = k+1$ due to Lemma 8, it follows that $\nu_G(L_f) \geq k+2$, which is a contradiction to our assumption. Hence, $a \prec_{\beta} x_f$. Since $\nu_{G-a}(L_f) = k+1$ and $\text{ad}_{\beta'}(x_f) = 1$, $\nu_{G-a}(L_f^-) = k$. Note also that $\text{ad}_{\beta}(x_f) = 1$ due to Lemmata 11 and 8, so that $\nu_G(L_{\beta}(x_f)) = k$ by our assumptions. Remember that there is a vertex for every column of \mathcal{B} that is not in L_f^- . If the neighbours of a are in \mathcal{B}_1 then a is vertex in a singleton group in $L_{\beta}(x_f)$, particularly because of $b_{1,1}, b_{1,2} \notin L_{\beta}(x_f)$. If the neighbours of a are in \mathcal{B}_k and \mathcal{B}_{k+1} then a is vertex in a singleton group in $L_{\beta}(x_f)$, particularly because of $b_{1,k}, b_{1,k+1} \notin L_{\beta}(x_f)$. Hence, $\nu_G(L_{\beta}(x_f)) > k$, which yields a contradiction to our assumption together with $x_f = b_{1,2}$ and $\text{ad}_{\beta}(b_{1,2}) = 1$. Therefore, $\text{gw}(G) \geq k+2$.

Now we give the upper bounds. It is known that $\text{lcwd}(G) \leq \text{pw}(G) + 2$ for G an arbitrary graph [11], where $\text{pw}(G)$ is the pathwidth of G . For path powers, the pathwidth is equal to the maximum clique size minus 1, which implies the first statement of the next result. For path powers on few vertices, the second statement gives an even better bound.

Lemma 10

- 1) Let G be a k -path power, with $k \geq 1$. Then, $\text{lcwd}(G) \leq k+2$ ([11]).
- 2) Let G be a k -path power on $l(k+1)+1$ vertices, with $2 \leq l \leq k$. Then, $\text{lcwd}(G) \leq l+1$.

Proof. It remains to prove the second statement. Let $\mathcal{B} = \langle b_{i,j} \mid 1 \leq j \leq l+1, 1 \leq i \leq r_j \rangle$ be a canonical bubble model for G . Let $\beta = \langle b_{k+1,l}, \dots, b_{k+1,1}, b_{k,l}, \dots, b_{2,1}, b_{1,2}, b_{1,1}, b_{1,3}, \dots, b_{1,l+1} \rangle$, i.e., the vertices in \mathcal{B} appear in β row by row, starting from the bottom row, and within a row, from right to left, except for the first row. We show that $\text{gw}(G, \beta) \leq l+1$. Let $x = b_{i,j}$ be a vertex of G . If $i \geq 2$ then $\nu_G(L_{\beta}[x]) \leq l$. To see this, observe that $b_{i+1,j'}, \dots, b_{k+1,j'} \in L_{\beta}[x]$ and $b_{1,j'}, \dots, b_{i,j'} \notin L_{\beta}[x]$ for all $j' < j$ and $b_{i,j'}, \dots, b_{k+1,j'} \in L_{\beta}[x]$ and $b_{1,j'}, \dots, b_{i-1,j'} \notin L_{\beta}[x]$ for all $j' \geq j$. Hence, the vertices of every column that are in $L_{\beta}[x]$ are in the same group. Since there are l columns in \mathcal{B} with vertices in $L_{\beta}[x]$, the claim holds. Now, let $i = 1$. It holds that $b_{2,j'}, \dots, b_{k+1,j'} \in L_{\beta}[x]$ for all $1 \leq j' \leq l$. If $x = b_{1,2}$ then $L_{\beta}[x]$ has exactly $l+1$ groups, since $b_{1,2}$ is not in the same group as any other vertex. It holds that $\text{ad}_{\beta}(b_{1,1}) = 0$, which is easy to check with the definition of function ad . Thus, $\nu_G(L_{\beta}(b_{1,1})) + \text{ad}_{\beta}(b_{1,1}) = l+1+0 \leq l+1$. For $j \geq 3$, $b_{1,1}, \dots, b_{k+1,1}, b_{1,2}, \dots, b_{k+1,j-2}, b_{1,j-1}$ are in the same group, and in $L_{\beta}[x]$, and $\nu_G(L_{\beta}[x]) \leq l$. We conclude that $\text{gw}(G, \beta) \leq l+1$.

With the lower and upper linear clique-width bounds, we are ready to give the complete characterisation.

Theorem 3. *Let G be a k -path power on n vertices, with $k \geq 1$ and $n \geq k + 2$.*

- *If $n \geq k(k + 1) + 2$ then $\text{lcwd}(G) = k + 2$.*
- *If $k + 2 \leq n \leq k(k + 1) + 1$ then $\text{lcwd}(G) = \lceil \frac{n-1}{k+1} \rceil + 1$.*

Note that k -path powers on at most $k + 1$ vertices are complete graphs and therefore have linear clique-width at most 2.

Corollary 1. *Let $k \geq 1$ and let G be a path power on at least two vertices. Then, $\text{lcwd}(G) \leq k + 1$ if and only if G does not contain the k -path power on $k(k + 1) + 2$ vertices as induced subgraph.*

With the characterisation in Corollary 1, we can construct a simple algorithm that computes the linear clique-width of path powers.

Theorem 4. *There is a linear-time algorithm that computes the linear clique-width of path powers.*

Proof. Let G be a path power. A canonical bubble model for G can be computed in linear time. Applying Corollary 1, $\text{lcwd}(G) = l + 1$ where l is the smallest number such that G does not contain an l -path power on $l(l + 1) + 2$ vertices as induced subgraph. This number is easy to determine from the computed bubble model.

6 Concluding Remark

It would be very interesting to see if our results can be extended to clique-width of path powers. We suspect that for large enough path powers, their clique-width is equal to their linear clique-width. We leave the resolution of this as an open problem and a future research direction.

Acknowledgements. We are grateful to anonymous referees for valuable remarks.

References

1. Boliac, R., Lozin, V.: On the Clique-Width of Graphs in Hereditary Classes. In: Bose, P., Morin, P. (eds.) ISAAC 2002. LNCS, vol. 2518, pp. 44–54. Springer, Heidelberg (2002)
2. Brandstädt, A., Dragan, F., Le, H.-O., Mosca, R.: New Graph Classes of Bounded Clique-Width. *Theory of Computing Systems* 38, 623–645 (2005)
3. Brandstädt, A., Engelfriet, J., Le, H.-O., Lozin, V.: Clique-Width for 4-Vertex Forbidden Subgraphs. *Theory of Computing Systems* 39, 561–590 (2006)
4. Corneil, D.G., Habib, M., Lanlignel, J.-M., Reed, B.A., Rotics, U.: Polynomial time recognition of clique-width ≤ 3 graphs. In: Gonnet, G.H., Viola, A. (eds.) LATIN 2000. LNCS, vol. 1776, pp. 126–134. Springer, Heidelberg (2000)
5. Corneil, D.G., Rotics, U.: On the Relationship between Clique-width and Treewidth. *SIAM Journal on Computing* 34, 825–847 (2005)

6. Courcelle, B., Engelfriet, J., Rozenberg, G.: Handle-rewriting hypergraph grammars. *Journal of Computer and System Sciences* 46, 218–270 (1993)
7. Courcelle, B., Makowsky, J.A., Rotics, U.: Linear time solvable optimization problems on graphs of bounded clique-width. *Theory of Computing Systems* 33, 125–150 (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, 23–52 (2001)
9. Courcelle, B., Olariu, S.: Upper bounds to the clique width of graphs. *Discrete Applied Mathematics* 101, 77–114 (2000)
10. Espelage, W., Gurski, F., Wanke, E.: Deciding clique-width for graphs of bounded tree-width. *J. Graph Algorithms Appl.* 7, 141–180 (2003)
11. Fellows, M.R., Rosamond, F.A., Rotics, U., Szeider, S.: Clique-width Minimization is NP-hard. In: *STOC 2006*, pp. 354–362 (2006)
12. Fomin, F., Golovach, P., Lokshtanov, D., Saurabh, S.: Clique-width: On the Price of Generality. In: *SODA 2009*, pp. 825–834 (2009)
13. Golombic, M.C., Rotics, U.: On the Clique-Width of Some Perfect Graph Classes. *International Journal of Foundations of Computer Science* 11, 423–443 (2000)
14. Gurski, F.: Characterizations for co-graphs defined by restricted NLC-width or clique-width operations. *Discrete Mathematics* 306, 271–277 (2006)
15. Gurski, F.: Linear layouts measuring neighbourhoods in graphs. *Discrete Mathematics* 306, 1637–1650 (2006)
16. Gurski, F., Wanke, E.: On the relationship between NLC-width and linear NLC-width. *Theoretical Computer Science* 347, 76–89 (2005)
17. Gurski, F., Wanke, E.: The NLC-width and clique-width for powers of graphs of bounded tree-width. *Discrete Applied Mathematics* 157, 583–595 (2009)
18. Heggernes, P., Meister, D., Papadopoulos, C.: A new representation of proper interval graphs with an application to clique-width. *Electronic Notes in Discrete Mathematics* 32, 27–34 (2009)
19. Heggernes, P., Meister, D., Papadopoulos, C.: Graphs of small bounded linear clique-width. Tech. rep. 362 in Informatics, University of Bergen (2007)
20. Heggernes, P., Meister, D., Papadopoulos, C.: Graphs of Linear Clique-Width at Most 3. In: Agrawal, M., Du, D.-Z., Duan, Z., Li, A. (eds.) *TAMC 2008*. LNCS, vol. 4978, pp. 330–341. Springer, Heidelberg (2008)
21. Kobler, D., Rotics, U.: Edge dominating set and colorings on graphs with fixed clique-width. *Discrete Applied Mathematics* 126, 197–221 (2003)
22. Lozin, V.: From tree-width to clique-width: excluding a unit interval graph. In: *ISAAC 2008*. LNCS, vol. 5369, pp. 872–883. Springer, Heidelberg (2008)
23. Lozin, V., Rautenbach, D.: Chordal bipartite graphs of bounded tree- and clique-width. *Discrete Mathematics* 283, 151–158 (2004)
24. Lozin, V., Rautenbach, D.: On the Band-, Tree-, and Clique-Width of Graphs with Bounded Vertex Degree. *SIAM Journal on Discrete Mathematics* 18, 195–206 (2004)
25. Lozin, V., Rautenbach, D.: The relative clique-width of a graph. *Journal of Combinatorial Theory, Series B* 97, 846–858 (2007)
26. Makowsky, J.A., Rotics, U.: On the clique-width of graphs with few P_4 s. *International Journal of Foundations of Computer Science* 10, 329–348 (1999)
27. Makowsky, J.A., Rotics, U., Averbouch, I., Godlin, B.: Computing Graph Polynomials on Graphs of Bounded Clique-Width. In: Fomin, F.V. (ed.) *WG 2006*. LNCS, vol. 4271, pp. 191–204. Springer, Heidelberg (2006)

Preserving Privacy versus Data Retention

Markus Hinkelmann and Andreas Jakoby

Institut für Theoretische Informatik, Universität zu Lübeck, Germany
{hinkelma, jakoby}@tcs.uni-luebeck.de

Abstract. The retention of communication data has recently attracted much public interest, mostly because of the possibility of its misuse. In this paper, we present protocols that address the privacy concerns of the communication partners. Our data retention protocols store streams of encrypted data items, some of which may be flagged as critical (representing misbehavior). The frequent occurrence of critical data items justifies the self-decryption of all recently stored data items, critical or not. Our first protocol allows the party gathering the retained data to decrypt all data items collected within, say, the last half year whenever the number of critical data items reaches some threshold within, say, the last month. The protocol ensures that the senders of data remain anonymous but may reveal that different critical data items came from the same sender. Our second, computationally more complex scheme obscures this affiliation of critical data with high probability.

1 Introduction

Recently, governments all over the world have increased their surveillance efforts. In 2006 the European Union adopted directive 2006/24/EC [10], on “the retention of data generated or processed in connection with the provision of publicly available electronic communications services or of public communications networks”. Member states have to implement the directive by 2009 as national law. By then, communication service providers must retain data that identify the source and the destination of communication, its type, date and duration for at least six months. Additionally, information about the location of mobile communication equipment has to be recorded. Officials want to use retained data to detect and investigate critical activities.

Certainly, these issues mean a conflict between investigational interests and preserving the sphere of personal privacy. A simple solution may be as follows: The communication providers encrypt the retained data of an user. If some suspicious facts justify a judicial order to open the stored data, the private decryption key is disclosed to the officials. Thus, the provider has always access to the retained data, at least to decryption keys of the users. The goal of this paper is to present protocols that implement such encryption and decryption processes and feature advanced properties. Our protocols allow the self-decryption of retained data if a threshold of critical activities is passed. Encrypted data that are stored before a prescribed period cannot be decrypted. We also care about the

anonymity of users, i.e. the stored data are only related to the encrypted identity of a user. Furthermore, the provider is not responsible to store any user related data except for data needed for the encryption and classification of the actual message. I.e. the provider is not responsible to retain any information about the users' behaviors. For enhanced data privacy third parties (the providers) should not be allowed to store private data longer than necessary. This includes the knowledge about the number of critical activities of a user, too.

Additionally, we propose techniques that data can only be associated with user if it has been retained in the prescribed period and that critical activities cannot be traced for a longer period of time. Up to our knowledge our protocols are the first that ensure this kind of privacy.

Related Work: With the introduction of the Internet data retention, surveillance and privacy have drawn a lot of attention in the fields of sociology and computer science. Marx [17] identified four conditions under that data retention raises ethical concerns: Collecting data involves physical or psychological harm, produce invalid results, crosses a personal boundary without notice, or violates trust. Having access to the data the temptation to misusing them is great. Blanchette and Johnson [4] argue that the important value of social forgetfulness is slipping away since the introduction of electronic data retention. Cryptography provides some hope to counter this threat, e.g. by introducing digital pseudonyms [7,8]. Nevertheless, electronic wiretapping means an architected security breach and it is necessary to limit its use to appropriate scenarios [16]. Several papers deal with the technical implementation of data retention [1,23,19]. But to our knowledge no scheme proposes solutions for an increased level of privacy in data retention.

Our scheme has the feature that the retained data automatically allow their decryption if a threshold of misbehavior is reached. Hence, secret sharing will be one important tool. Shamir [22] and Blakley [3] independently introduced secret sharing schemes. Several schemes using general access structures have been presented (e.g. [2,13]). Since original shares are as large as the secret, one might ask to reduce the size of the shares. Czimraz [9] showed that this is impossible for every access structure. Using an information dispersal algorithm [20] Krawczyk [15] proposes a scheme to reduce the share size.

Using Shamir shares our protocols allow that a secret can be decrypted only if a threshold is reached within a determined period of time. If the messages are too old they become useless for the decryption. Rivest et al. [21] introduced the notion of time-release crypto. They propose to use computational puzzles as time-locks to schedule the first point in time when it is possible to decrypt data. In a similar way timed commitments and signatures are implemented by Boneh and Naor [6]. Haber et al. [11] presented protocols for cryptographical timestamping that ensure the privacy of the data and the integrity of the timestamp. As second technique, we use pseudorandom number generators (PRNG) to create keys and identification information. Blum and Micali [5], and Yao [24] introduced the notion for cryptographically robust PRNG. In Sections 3 and 4 we present basic protocols. We discuss these protocols and identify a new problem type:

the history of messages. Our main scheme, presented in section 5, obscures the message history with high probability (w.h.p.).

2 Preliminaries

Let X be a discrete random variable that takes values from the set of real numbers or strings over the alphabet Σ . If X is uniformly distributed, we also write $X \in_R \Sigma^*$. Using a function symbol f we write $\Pr[f(X) = y]$ for $\sum_{x: f(x)=y} \Pr[X = x]$.

Pseudorandom Number Generators: Pseudorandom number generators (PRNG) are functions having special properties which make them very suitable for cryptography. If the input (the so called *seed*) of a PRNG is unknown, the output is indistinguishable from random strings for computationally bounded adversaries. On the other hand, PRNGs are deterministic functions. Thus, if the seed is known, we are able to reproduce the output of a PRNG.

Definition 1. Let $h : \mathbb{N} \rightarrow \mathbb{N}$ be a polynomial stretch function with $h(\ell) > \ell$ for all ℓ . Let S and Y be uniformly distributed random variables taking values in Σ^ℓ and $\Sigma^{h(\ell)}$, respectively. We call a function $G : \Sigma^\ell \rightarrow \Sigma^{h(\ell)}$ a PRNG if for all probabilistic polynomial-time bounded algorithms \mathcal{A} , for all polynomials p and for all sufficiently large ℓ it holds that $|\Pr[\mathcal{A}(Y) = 1] - \Pr[\mathcal{A}(G(S)) = 1]| < \frac{1}{p(\ell)}$.

The following proposition describes how to stretch the output of a PRNG. It appears in [12] and is due to an observation made by Goldreich and Micali. For a string $w = w_1 \dots w_\ell$ and $1 \leq a \leq b \leq \ell$ let $w_{\{a, \dots, b\}}$ be the substring $w_a \dots w_b$. The operator \circ denotes the concatenation of strings. Let $G : \{0, 1\}^\ell \rightarrow \{0, 1\}^{\ell+1}$ be a PRNG. We define $G^{(1)}(S) = G(S)$, and inductively, for all $i \geq 1$, $G^{(i+1)}(S) = G(G^{(i)}(S)_{\{1, \dots, \ell\}} \circ G^{(i)}(S)_{\{\ell+1, \dots, \ell+i\}})$. Then, for every polynomial q and sufficiently large ℓ it holds that $G^{(q(\ell))} : \{0, 1\}^\ell \rightarrow \{0, 1\}^{\ell+q(\ell)}$ is a PRNG.

Thus, for every polynomial h the function $\widehat{G} = G^{(h(\ell))}$ is a PRNG. As seen above, we inductively define the PRNGs $\widehat{G}^{(1)}(S) = \widehat{G}(S)$ and $\widehat{G}^{(i+1)}(S) = \widehat{G}(\widehat{G}^{(i)}(S)_{\{1, \dots, \ell\}} \circ \widehat{G}^{(i)}(S)_{\{\ell+1, \dots, \ell+i \cdot h(\ell)\}})$ for $i \leq q(\ell)$.

Definition 2. Let $S \in_R \{0, 1\}^\ell$ be a random string and $i \in \mathbb{N}$. We define the seed generating function $\text{seed}^{(i)} : \{0, 1\}^\ell \rightarrow \{0, 1\}^\ell$ and the pseudorandom string function $\text{rand}^{(i)} : \{0, 1\}^\ell \rightarrow \{0, 1\}^{h(\ell)}$ such that $\text{seed}^{(0)}(S) = S$, and for $i \geq 1$ it holds that $\text{seed}^{(i)}(S) = \widehat{G}^{(i)}(S)_{\{1, \dots, \ell\}}$ and $\text{rand}^{(i-1)}(S) = \widehat{G}^{(i)}(S)_{\{\ell+1, \dots, \ell+h(\ell)\}}$.

We observe that for $i \in \mathbb{N}$ we can use $\text{seed}^{(i)}(S)$ to calculate the values $\text{seed}^{(i+1)}(S)$ and $\text{rand}^{(i)}(S)$ by $\widehat{G}(\text{seed}^{(i)}(S)) = \text{seed}^{(i+1)}(S) \circ \text{rand}^{(i)}(S)$. Starting with a seed S we can generate sequences $\{\text{seed}^{(i)}(S)\}_{i=0}^\tau$ and $\{\text{rand}^{(i)}(S)\}_{i=0}^\tau$ efficiently. For a time step t we will use $\text{rand}^{(t)}(S)$ to encrypt some data. Note that if τ gets large, the probability to distinguish between random strings and pseudorandom strings increases. To work against this we may increase ℓ . But to inhibit this threat we can additionally substitute such a seed $\text{seed}^{(t)}(S)$ by a new random string from

time to time. In the following we will focus only on the way how we use the two sequences. Although we can efficiently compute $\text{seed}^{(j)}(S)$ and $\text{rand}^{(j)}(S)$ for $j \geq i$ on input $\text{seed}^{(i)}(S)$, no probabilistic polynomial time algorithm on input $\text{seed}^{(i)}(S)$ or $\text{rand}^{(i)}(S)$ is able to deduce more than negligible information about the strings $\text{seed}^{(k)}(S)$ and $\text{rand}^{(k-1)}(S)$ for $k < i$ due to the cryptographical robustness of the PRNG G .

Shamir's Secret Sharing: Let \mathcal{F} be a field with more than n elements. In [22] Shamir presented a threshold scheme to divide some data $D \in \mathcal{F}$ into n shares D_1, \dots, D_n such that D can only be reconstructed if one knows at least k shares D_i . I.e. let p be a random polynomial over the field \mathcal{F} of degree $k - 1$ with $p(0) = D$. For $1 \leq i \leq n$ we can choose $D_i := p(i)$. Having access to k shares we can compute the polynomial p by Lagrange interpolation. If we have obtained less than k shares, then for each value D' we can generate a polynomial p' with $p'(0) = D'$ that is consistent with the obtained shares. Thus, this scheme provides information-theoretical privacy.

Anonymity: Let $\text{subj}(M)$ be the decrypted *subject* of interest about a message M . For example, this may be the information about its sender, receiver, or contents. We assume that $\text{subj}(M)$ is encrypted in the message. Since our protocols are randomized we regard $\text{subj}(M)$ as a random variable. The set $SU = \{\text{subj}(M') \mid M' \text{ is a possible message}\}$ describes the set of all possible subjects of interest from all messages.

Definition 3. Let I be a uniformly distributed random variable taking values from SU . For a message M we call the information about a message $\text{subj}(M)$ anonymous for a party A if for all probabilistic polynomial-time bounded algorithms \mathcal{A} , for every polynomial p , for all communication strings c and any content r of the random string of A , for all sufficiently large security parameters k for encryption of $\text{subj}(M)$ it holds that $|\Pr[\mathcal{A}(\text{subj}(M), M, c, r) = 1] - \Pr[\mathcal{A}(I, M, c, r) = 1]| \leq \frac{1}{p(k)}$.

In the following we usually encrypt the subject of a message using a bitwise XOR (\oplus) with a pseudorandom string. Then, we regard the security parameter k as the length of the seed for the PRNG.

3 Basic Structure and Types of Messages

We divide the participating parties that into three groups.

1. Users, who e.g. surf in the Internet or send emails. They want to use these services privately. We also refer to users as senders or receivers.
2. Communication service providers offer and control access to the system. Providers are corporations that want to maximize their profit and minimize their costs and responsibilities.
3. The officials (government, police, ...) ensure that other parties respect the law. In the context of data retention we call them gathering party.

To control the senders some governments have already prescribed data retention. I. e. providers are responsible for collection and storage of information about the communication of the senders. If the officials lawfully demand the retained data for a certain user, the providers have to disclose them. This approach of data retention rises severe concerns about privacy and massively increasing storage costs of the providers [25]. In the following we present protocols that ensures the privacy of the senders, liberates the providers from storing the retained data items, and allows officials to inspect all data items from a predetermined period if a party has recently committed too many critical actions.

A critical interaction might occur if a user sends an email to a party that is already a subject of investigation. Depending on the type of interaction we distinguish between critical and non-critical interactions and, thus, between critical and non-critical messages. We assume that the gathering party supplies the provider with a blacklist of critical actions. Whenever a sender interacts with the system the provider classifies this action as critical or non-critical. The provider prepares an encrypted data retention message for this interaction and sends the message to the gathering party. We will investigate the problem of permitting a gathering party the decryption of recorded messages only if it has received enough critical messages within a predetermined period of time.

A message M can be described as a four tuple $M = \langle \text{time}, \text{id}, \text{share}, \text{load} \rangle$. where $\text{time}(M)$ denotes the initiating time (the time the message was sent), $\text{id}(M)$ denotes some kind of message ID or sender pseudonym, and $\text{share}(M)$ denotes the shares corresponding to that message. $\text{load}(M)$ consists of further information associated with M – we assume that this part of a message includes the subject $\text{subj}(M)$ of the message. If M is non-critical we choose $\text{id}(M) = \text{share}(M) = 0$. The sequence of messages belonging to the same sender may reveal information about his (critical) activities if the sender can be identified. We denote the sequence of messages belonging to the same sender as the history of the sender. Given a message M we also call the sequence of messages M' belonging to the sender of M with $\text{time}(M) > \text{time}(M')$ the history of M .

Encryption of the Load: We assume there are n different senders. The identity of a sender is a unique code word of a binary blockcode \mathcal{I} with Hamming distance δ and more than n code words. If a sender \mathcal{I}_i commits an activity, the gathering party wants to retain data corresponding to that activity. Depending on the type of the activity, critical or not, the provider prepares a message M , critical or not, including the encrypted retained data (the encrypted subject) in $\text{load}(M)$. Then, the provider sends the message to the gathering party. The gathering party stores all retained data in a pool of messages. If we have obtained a decryption key from critical messages, we must identify the messages that can be decrypted with this key. Therefore, we introduce an indicator string R , a pseudo-random seed L , and the (encrypted) fingerprint $\text{fp}_{R,L}(\mathcal{I}_i)$ that corresponds to the identity of the sender. The implementation of the fingerprint is based on Naor's commitment scheme [18]. We define $\text{load}(M) = \langle R, \text{fp}_{R,L}(\mathcal{I}_i), \text{enc}_K(\text{subj}(M)) \rangle$. Let t be the time step when M is sent. For the random string $S \in \{0,1\}^\ell$ we generate the keys K and $L \in \{0,1\}^\ell$ from $\text{seed}^{(t)}(S)$ by $\text{rand}^{(t)}(S) = K \circ L$.

Let $G' : \{0, 1\}^\ell \rightarrow \{0, 1\}^{|\text{subj}(M)|}$ be a PRNG. Then, we define the encryption function as $\text{enc}_K(\text{subj}(M)) = G'(K) \oplus \text{subj}(M)$.

Let $m \in \mathbb{N}$ be the security parameter for the fingerprint and $G'' : \{0, 1\}^\ell \rightarrow \{0, 1\}^{m \cdot |\mathcal{I}_i|}$ be a PRNG. Let $G''(L) = B_1 \circ \dots \circ B_{|\mathcal{I}_i|}$ and $R = R_1 \circ \dots \circ R_{|\mathcal{I}_i|}$ with $B_i, R_i \in \{0, 1\}^m$ and R_i is a random string containing at least one 1. Then, we define $\text{fp}_{R,L}(\mathcal{I}_i) = \tilde{B}_1 \circ \dots \circ \tilde{B}_{|\mathcal{I}_i|}$ as follows: If the j th bit b_j of \mathcal{I}_i is 0 then we choose $\tilde{B}_j := B_j$ and if $b_j = 1$ choose $\tilde{B}_j := B_j \oplus R_j$. Having access to the message M and the keys K and L by expanding $\text{seed}^{(t)}(S)$, we can decrypt \mathcal{I}_i and check whether it matches a fingerprint $\text{fp}_{R,L}(\mathcal{I}_i)$. Let K', L' denote two keys generated by $\text{seed}^{(t)}(S')$ with $S \neq S'$, then, $\text{fp}_{K,L}(\mathcal{I}_i)$ should be different to $\text{fp}_{K',L'}(\mathcal{I}_j)$ for every different identity $\mathcal{I}_j \in \mathcal{I}$. If the unwanted case happens, i.e. we associate the message M to the wrong identity \mathcal{I}_j , we say that a collision occurs. Assume that with probability $(1 - q)$ all strings K and $G'(K)$, L , and $G''(L)$, respectively, are different for all messages. If for the security parameter it holds that $m \geq \ell + 1$, then the collision probability is at most $q + (1 - q) \cdot 2^{-(\delta - 1)\ell}$. We can ensure that q is very small by increasing the seed length. Next, we analyze the information about \mathcal{I}_i and $\text{subj}(M)$ that can be deduced from $\text{load}(M)$.

Lemma 4. *For any probabilistic polynomial algorithm \mathcal{A} , all polynomials p and for all sufficiently large ℓ it holds that $\Pr[\mathcal{A}(\text{load}(M)) = \mathcal{I}_i] < \frac{1}{n} + \frac{1}{p(\ell)}$ and $\Pr[\mathcal{A}(\text{load}(M)) = \text{subj}(M)] < \frac{1}{|\mathcal{S}\mathcal{U}|} + \frac{1}{p(\ell)}$.*

If we do not have any information about the decryption keys K and L then the advantage to guess \mathcal{I}_i and $\text{subj}(M)$ for a message M is negligible.

Allocation and Generation of the Keys: We propose that critical messages themselves contribute to obtaining the decryption key if a sender commits too many misbehaviors within a specific period of time. Let $\mathcal{T}_i \subset \mathbb{N}$ with $i \in \mathbb{N}$ denote the i th period. Let Π_t be the set of all periods i such that $t \in \mathcal{T}_i$ and let $t_{\min}(\mathcal{T}_i) = \min_{t \in \mathcal{T}_i} t$. To implement the wanted behavior of self decryption we will assign a Shamir shares [22] of the key to each message: For each sender and each period \mathcal{T}_i we generate a random polynomial p of degree $d - 1$ over a field \mathcal{F} that is sufficiently large such that $p(0) = \text{seed}^{(t_{\min}(\mathcal{T}_i))}(S)$. If we send a critical message M in time step $t \in \mathcal{T}_i$, then we attach the share $p(t - t_{\min}(\mathcal{T}_i) + 1)$ to M . Thus, if we receive d messages within \mathcal{T}_i , we can reconstruct $\text{seed}^{(t_{\min}(\mathcal{T}_i))}(S)$ by Lagrange interpolation. Afterwards, we can generate the sequence of $\text{seed}^{(t)}(S)$ and $\text{rand}^{(t)}(S)$ for all $t \in \mathcal{T}_i$. According to the used encryption of the load of a message we can identify those (critical and non-critical) messages where we can correctly decrypt the fingerprint $\text{fp}_{R,L}(\mathcal{I}_i)$. For each of these identified messages M we can also decrypt $\text{subj}(M)$.

4 A Threshold Scheme for Critical Data

In this section we present a scheme to construct critical messages.

Scheme Initialization: The gathering party supplies the provider with a black-list of critical activities. Then, for each user i the provider performs the

following steps: The provider generates an initial seed S and a unique random number u . For each period \mathcal{T}_j we choose a random polynomial p_j with $p_j(0) = \text{seed}^{(t_{\min}(\mathcal{T}_j))}(S)$.

Sending a critical message: Assume that user i performs a critical activity at round t . The provider identifies it by his blacklist. Using the user-specific seed S and random number u the provider generates the message M with $\text{time}(M) = t$, $\text{id}(M) = u$, $\text{share}(M) = \langle p_{j_1}(x_{j_1}), p_{j_2}(x_{j_2}), \dots, p_{j_\ell}(x_{j_\ell}) \rangle$ where $\Pi_t = \{j_1, \dots, j_\ell\}$ and $x_j = t - t_{\min}(\mathcal{T}_j) + 1$, as well as $\text{load}(M) = \langle R, \text{fp}_{R,L}(\mathcal{I}_i), \text{enc}_K(\text{subj}(M)) \rangle$. Then, the provider sends M to the gathering party.

Identification and Decryption: If the gathering party has received d critical messages with the same id u within \mathcal{T}_j it reconstructs $\text{seed}^{(t_{\min}(\mathcal{T}_j))}(S)$ by Lagrange interpolation. This reconstruction can be done efficiently if the critical messages are sorted according to their $\text{id}(M)$. Afterwards, it can generate all subsequent values $\text{seed}^{(t)}(S)$ and $\text{rand}^{(t)}(S)$. Using these values the gathering party is able to identify all (critical and non-critical) messages where it can correctly decrypt the fingerprint $\text{fp}_{R,L}(\mathcal{I}_i)$. For each of these identified messages M the gathering party decrypts $\text{subj}(M)$.

Let $\mathcal{IU}_{\text{unident}}$ be the set of all identities that the gathering party has not been able to identify, i.e., for any $\mathcal{I}_i \in \mathcal{IU}_{\text{unident}}$ and all periods \mathcal{T}_j the gathering party has not received d critical messages associated with \mathcal{I}_i within period \mathcal{T}_j .

Theorem 5. *Let M be a message that is associated with $\mathcal{I}_i \in \mathcal{IU}_{\text{unident}}$. Then, M is anonymous to the gathering party with respect to $\mathcal{IU}_{\text{unident}}$.*

If we restrict ourselves to use only one period $\mathcal{T}_i = \mathcal{T}_0$, then we can also use the protocols proposed by Jarecki and Shmatikov [14] since they may only encrypt a constant number of messages of a tag (user). If the messages can be decrypted, all messages with the same tag can be decrypted. A PRNG as key generator allows us to encrypt a polynomial number of messages with the same tag and also prevents the decryption of messages with the same tag that were encrypted long ago.

Privacy of the Message History: Now, we are going to analyze the situation where the gathering party has received d or more critical messages with the same id u . Let t' be the earliest time step such that we can recover a seed $\text{seed}^{(t')}(S)$ from these messages. Then, we can decrypt the identity \mathcal{I}_i and all messages from the corresponding sender that are initiated at time step $t \geq t'$. In addition, we are able to identify the complete history of critical messages since all critical message have the same id u . Therefore, we can also determine partial knowledge about the history of the identified sender. Recall that even for an identified user one of our goals is to ensure the anonymity of messages initiated at steps $t < t'$. We can guarantee this for non-critical messages since we cannot compute $\text{seed}^{(t)}(S)$ for $t < t'$ by construction. But the history of critical messages is still disclosed. One may use, for instance, the following approach to obscure the history: We allow that every user can use a fixed number α of different ids. Hence, it is possible that a user can choose one of his IDs to be used in a

message. If we assume that each id is only valid for a fixed period of time and if the user performs only a small number of critical activities he can hide the history that is associated with a specific id. Hence, in the worst case, the sender might be able to send $(2\alpha + 1)(d - 1)$ critical messages within Δ time steps such that he cannot be identified. However, in most cases it is desirable that a sender of critical messages is identified whenever the threshold d of critical messages is reached.

5 A Protocol for Obscuring the History

In the previous section we have presented a protocol that allows an observer to gain some knowledge on the history of the parties. This knowledge includes the appearance of critical data even if the content of the critical data remains decrypted. In the following protocol we will change the way how critical messages are generated. This allows us to mix and thereby obscure the histories of the critical messages. This protocol does not change the generation of non-critical messages. More precise, we will replace the *polynomial* or *pseudo sender ID* of every critical message by an ambiguous randomly chosen message ID $\text{id}_{\text{act}} \in_R \{1, \dots, N\}$, i.e. by an ID that may appear for several messages of several senders. To connect consecutive messages of the same sender we will include the message ID $\text{id}_{\text{pre}} \in \{1, \dots, N\}$ of the preceding message (or a randomly chosen message ID if the actual message is the first message of the sender) in the actual message. Hence, we modify the structure of a message M as follows: $M = \langle \text{time}, \text{id}_{\text{act}}, \text{id}_{\text{pre}}, \text{share}, \text{load} \rangle$. Recall, that we assume that the IDs are chosen randomly and are not unique, i.e. within a certain period of time several messages with the same ID will be used with a non negligible probability.

Let $\mathcal{M}_{[t]}$ denote the set of all messages collected by the gathering party until step t . We can draw a *message graph* $G_{[t]} := (\mathcal{M}_{[t]}, E_{[t]})$ where for $M_1, M_2 \in \mathcal{M}_{[t]}$ it holds that $(M_1, M_2) \in E_{[t]}$ iff $\text{time}(M_1) > \text{time}(M_2)$ and $\text{id}_{\text{pre}}(M_1) = \text{id}_{\text{act}}(M_2)$. A directed path from a source in $G_{[t]}$ to a sink denotes the possible sequence of all messages of a sender. We have to describe an algorithm that detects a correct sequence if the threshold of critical messages is reached. Analogously to the identification mechanism, we add the encrypted sender ID to the load of each critical message and we assume that, as in the previous protocols, $\text{share}(M)$ gives us a share of the seed of a pseudorandom number generator. Having the desired number of d consecutive critical messages M_1, \dots, M_d we can compute a corresponding seed and by using this seed we can determine (decrypt) a value for the sender ID of every message id_i on this sequence. If all values id_i are equal, then we assume that these messages were initiated by the sender with ID id_i . Note that if such a sequence is initiated by one sender the ID of this sender will be detected. On the other hand, following our analysis to identify the sender of non-critical data it follows that the probability of a false positive, i.e. that we claim that a sequence is initiated by the wrong sender, is negligible.

Lemma 6. *Let ℓ be the length of S and m be the block length of indicator string. If the security parameter $m \geq \ell + 1$ then probability of a false positive is at most*

$q + (1 - q) \cdot 2^{-(\delta-1)\ell}$ where $(1 - q)$ is the probability that all seeds and pseudo-random strings used for encrypting and fingerprinting are different.

Now, we investigate the efficiency of our algorithm to detect a sequence of consecutive critical messages M_1, \dots, M_d that are initiated by the same sender within a time period \mathcal{T}_i of length Δ . Let m_t denote the number of messages M with timestamp $t = \text{time}(M)$, let $m_{\max} := \max_t m_t$ and let $m_{\min} := \min_t m_t$. If we assume that at every round every party initiate a critical message with probability p_{cm} , then by some standard calculations one can show that $\Pr[m_t \leq \frac{2}{3} \cdot p_{cm}n] \leq e^{-2p_{cm}n/9}$ and $\Pr[m_t \geq \frac{4}{3} \cdot p_{cm}n] \leq e^{-4p_{cm}n/3}$ where n denotes the number of participating parties. Hence, if we choose N such that $N \in 2^{o(p_{cm}n)}$ then with probability $1 - N^{-z}$ we have $\frac{2}{3}p_{cm}n \leq m_t \leq \frac{4}{3}p_{cm}n$ for every constant z . Hence, we can assume that m_{\min} and m_{\max} only deviate from each other by a factor of 2. In the following we assume that $m_{\max} = N^\varepsilon$ for some appropriate chosen values $\varepsilon < 1$.

Lemma 7. *With probability $1 - (e \cdot \Delta \cdot N^{\varepsilon-1})^{-(k-1)\Delta}$ for every constant $k > 1$ the number of different sequences of d consecutive critical messages ending with message M within a period of length Δ is bounded by $(e \cdot \Delta \cdot N^{\varepsilon-1})^d$.*

If we have $\varepsilon = \frac{1}{2}$, then $m_{\max} = \sqrt{N} \in \omega(\ln N)$ and $c^d = (e \cdot \Delta / \sqrt{N})^d$ where the polynomial degree d is a constant given by the system. Whenever a new message arrives at the gathering party, it has to search in the message graph whether there exists a sequence of d consecutive critical messages in the actual period that ends with the recently received message. Thus, the lemma above gives us a time bound for our algorithm for detecting such a sequence. Let \tilde{m}_t denote the number of messages M with timestamp $t = \text{time}(M)$ that do not belong to a sequence of d consecutive critical messages within a period of length Δ and let $\tilde{m}_{\min} := \min_t \tilde{m}_t$. Analyzing the randomized message graph we can show:

Theorem 8. *For every $k > 3$ and every message M with probability $1 - N^{-(k-3)}$ we cannot deduce any information about the history of M if we investigate messages that are initiated in a round $t \leq \text{time}(M) - \frac{N}{\tilde{m}_{\min}} \cdot \left(k \cdot \frac{N^2 \cdot \log_2 N}{N - \tilde{m}_{\min}} + \frac{\Delta}{d-1} \right)$.*

Asymptotically, the time until $|I(t, M)| = N$ is in $\mathcal{O}(N^2 \log(N) / \min\{\tilde{m}_{\min}, N - \tilde{m}_{\min}\})$. Hence, if $\tilde{m}_{\min} = \varepsilon \cdot N$ for $\varepsilon < 1$, then the required time is in $\mathcal{O}(N \log(N))$.

6 Conclusions

In this paper we presented a scheme for data retention that allows self-decryption if the number of critical messages reaches a threshold. As long as the messages cannot be decrypted the sender is anonymous to the gathering party. Furthermore, we introduced the history of messages as subject of privacy. Our scheme ensures the privacy of the history of non-critical messages. For critical messages we propose a protocol that obscures the history. The runtime of this protocol is polynomial in the parameters $N^{\varepsilon-1}$ and Δ but exponential in d . In our protocols all messages of a user are encrypted by the same provider. An interesting question is whether we can extend our protocols such that users can use several providers that do not share information about their customers.

References

1. Agarwal, A., Li, H., Roy, K.: Drg-cache: a data retention gated-ground cache for low power. In: DAC, pp. 473–478. ACM, New York (2002)
2. Benaloh, J.C., Leichter, J.: Generalized secret sharing and monotone functions. In: Goldwasser, S. (ed.) CRYPTO 1988. LNCS, vol. 403, pp. 27–35. Springer, Heidelberg (1990)
3. Blakley, G.: Safeguarding cryptographic keys. In: AFIPS (1979)
4. Blanchette, J.-F., Johnson, D.G.: Data retention and the panoptic society: The social benefits of forgetfulness. *The Information Society* 18, 33–45 (2002)
5. Blum, M., Micali, S.: How to generate cryptographically strong sequences of pseudo random bits. In: FOCS, pp. 112–117 (1982)
6. Boneh, D., Naor, M.: Timed commitments. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 236–254. Springer, Heidelberg (2000)
7. Chaum, D.: Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM* 24(2), 84–88 (1981)
8. Chaum, D.: Security without identification: Transaction systems to make big brother obsolete. *Commun. ACM* 28(10), 1030–1044 (1985)
9. Csirmaz, L.: The size of a share must be large. *J. Cryptology* 10(4), 223–231 (1997)
10. European Parliament and Council. Directive 2006/24/EC (March 2006)
11. Haber, S., Stornetta, W.S.: How to time-stamp a digital document. *J. Cryptology* 3(2), 99–111 (1991)
12. 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)
13. Ito, M., Saito, A., Nishizeki, T.: Secret sharing scheme realizing general access structure. In: Globecom, pp. 99–102 (1987)
14. Jarecki, S., Shmatikov, V.: Handcuffing big brother: an abuse-resilient transaction escrow scheme. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 590–608. Springer, Heidelberg (2004)
15. Krawczyk, H.: Distributed fingerprints and secure information dispersal. In: PODC, pp. 207–218 (1993)
16. Landau, S.: Security, liberty, and electronic communications. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 355–372. Springer, Heidelberg (2004)
17. Marx, G.T.: An ethics for the new surveillance. *Inf. Soc.* 14(3) (1998)
18. Naor, M.: Bit commitment using pseudorandomness. *J. Crypt.* 4(2), 151–158 (1991)
19. Ng, K., Liu, H.: Customer retention via data mining. *Artif. Intell. Rev.* 14(6), 569–590 (2000)
20. Rabin, M.O.: Efficient dispersal of information for security, load balancing, and fault tolerance. *J. ACM* 36(2), 335–348 (1989)
21. Rivest, R.L., Shamir, A., Wagner, D.A.: Time-lock puzzles and timed-release crypto. Technical report, Cambridge, MA, USA (1996)
22. Shamir, A.: How to share a secret. *Commun. ACM* 22(11), 612–613 (1979)
23. van Wanrooij, W., Pras, A.: Data on retention. In: Schönwälder, J., Serrat, J. (eds.) DSOM 2005. LNCS, vol. 3775, pp. 60–71. Springer, Heidelberg (2005)
24. Yao, A.C.-C.: Theory and applications of trapdoor functions. In: FOCS, pp. 80–91 (1982)
25. Zuccato, A., Rannenber, K.: Data retention has serious consequences. CEPIS Position Paper, LSI SIN (04)01 (2004)

Kolmogorov Complexity and Combinatorial Methods in Communication Complexity

Marc Kaplan and Sophie Laplante

LRI, Université Paris-Sud XI, 91405 Orsay CEDEX, France
{kaplan,laplante}@lri.fr

Abstract. We introduce a method based on Kolmogorov complexity to prove lower bounds on communication complexity. The intuition behind our technique is close to information theoretic methods [1,2]. Our goal is to gain a better understanding of how information theoretic techniques differ from the family of techniques that follow from Linial and Shraibman’s work on factorization norms [3]. This family extends to quantum communication, which prevents them from being used to prove a gap with the randomized setting.

We use Kolmogorov complexity for three different things: first, to give a general lower bound in terms of Kolmogorov mutual information; second, to prove an alternative to Yao’s minmax principle based on Kolmogorov complexity; and finally, to identify worst case inputs.

We show that our method implies the rectangle and corruption bounds [4], known to be closely related to the subdistribution bound [2]. We apply our method to the hidden matching problem, a relation introduced to prove an exponential gap between quantum and classical communication [5]. We then show that our method generalizes the VC dimension [6] and shatter coefficient lower bounds [7]. Finally, we compare one-way communication and simultaneous communication in the case of distributional communication complexity and improve the previous known result [7].

1 Introduction

Yao introduced the model of communication complexity in 1979 [8]. It has since become a central model of computation, studied for itself as well as for its numerous applications. The model addresses problems whose inputs are shared among different players, who have to communicate in order to solve them.

We wish to determine, for a given communication problem, how many bits the players have to exchange in order to solve it. A simple answer is that the messages should contain at least enough information to solve the problem. For example, they need to distinguish between inputs that produce different outputs. This idea has led to many lower bound techniques, and in particular to proofs involving information theory.

Shannon’s information theory’s original purpose was to study communication problems [9], so it seems natural that these techniques found applications in the

field of communication complexity. Information complexity is a general lower bound method [1], but in many other cases, ad hoc proofs have been given for specific problems [7,10,5]. One of the appealing features of these proofs is the way they capture the intuition of the hardness of the problem very naturally. However, by using elaborate results in information theory which in turn are based on statistics and probability, the essential mechanics of the proof is not always so readily apparent.

The use of Kolmogorov complexity has proven to be useful for lower bounds in various models, such as time complexity [11], average-case complexity, communication complexity [12], communication complexity of individual instances [13,14], and randomized and quantum query complexity [15]. The intuition that arises from both Kolmogorov complexity and information theory is often close, but they differ in their underlying mechanics: where information theory uses statistics, Kolmogorov complexity uses combinatorics. One of our goals is to capture the intuition of the information theoretic approach, while bringing out the combinatorial nature of these proofs. The main tool from Kolmogorov complexity that we use is incompressibility. It allows us to identify inputs that require a large amount of communication. The second is mutual information between the input and the transcript, which gives us a general expression on the amount of information that the player must exchange.

One of the main open problems in quantum communication complexity is to show an exponential gap between classical and quantum communication complexity, for a total function. The hidden matching relation was introduced to exhibit such a gap [5]. However, this problem falls short of this goal since the problem is a relation. More recently, it has been proved that the gap holds for a partial function [16], but the question remains open for total functions.

Linial and Shraibman's work on randomized and quantum communication complexity [3], and recent subsequent work, such as [17,18], can be viewed as mounting evidence that there is at most a polynomial gap between classical and quantum communication complexity, for total functions. Indeed, their method generalizes most of the previously known lower bound techniques, including discrepancy, trace norm [19], and some Fourier based techniques [20]; and these techniques all extend to the quantum setting. It was recently shown that there are problems where information theoretic techniques can prove significantly stronger randomized lower bounds than the factorization norm and related methods [21], which indicates that information theoretic and related techniques are essential for proving gaps between randomized and quantum communication complexity.

Main results. We give a general technique based on Kolmogorov complexity to prove lower bounds on deterministic and randomized communication complexity. Its formulation is very similar to the information complexity method [1].

We also prove a version of Yao's theorem based on Kolmogorov complexity, which allows us to restrict the random choices to a single incompressible string. By choosing both the worst case input and the random choices of the algorithm to be incompressible, we get the advantage of having them be independent of one another, which tends to simplify the proofs.

We show that our method is related to the corruption [4] and subdistribution bounds [2], placing it on the side of information theoretic methods.

We apply our method to the hidden matching problem [5], and also prove that our technique generalizes the VC dimension [6] and shatter coefficient lower bound [7]. Kolmogorov complexity turns out to be a very good tool in these cases, since it highlights very nicely the combinatorial nature of the proofs. Finally, we use combinatorial techniques to compare one-way and simultaneous communication in the multi-player setting. The result was previously known [7], but we significantly improve the error dependence.

2 Preliminaries

2.1 Communication Complexity

Let X , Y and Z be finite sets and $f : X \times Y \rightarrow Z$. In the communication complexity model, two players, Alice and Bob, each receive an input, and their goal is to compute f . Neither of them sees the other input. To perform this task, they communicate, and at the end, Bob outputs the value of the function.

Messages are sent according to a communication protocol. The cost of the protocol is the sum of messages' length (in the worst case). The communication complexity of the function f is the cost of the best protocol that computes f . We denote it $D(f)$. Notice that, since we are only interested in the communication between the players, we can assume that they have unlimited computational power. We will consider these variants of the model:

- *One way communication* $D^{A \rightarrow B}(f)$: Alice sends a single message to Bob.
- *Simultaneous messages* $D^{\parallel}(f)$: Alice and Bob each send a single message to a referee, who outputs $f(x, y)$.
- *Communication complexity of relations*: Let $\mathcal{R} \subseteq X \times Y \times Z$ be a ternary relation. Alice receives $x \in X$ and Bob $y \in Y$. Bob has to output any z such that $(x, y, z) \in \mathcal{R}$.

One important notion is the *transcript* of the protocol on input (x, y) . This is the concatenation of the messages sent by Alice and Bob when they receive inputs x and y . We assume for simplicity that the protocol has the property that the length of the messages in each round depends only on the round, and the length of the inputs. (We may always pad the messages so that this holds.) For one-way communication, the transcript is just the message sent by Alice to Bob.

A monochromatic rectangle for f is a set $R = S \times T$ with $S \subseteq X$ and $T \subseteq Y$ such that there exists $b \in \{0, 1\}$ and for all $(x, y) \in R$, $f(x, y) = b$. A classical result in communication complexity states that a deterministic protocol partitions the set of inputs into monochromatic rectangles, each rectangle corresponding to a transcript of the protocol [22]. Let μ be a probability distribution over $X \times Y$ and $\varepsilon > 0$. A rectangle is called (μ, ε) -monochromatic for f if there exists a $b \in \{0, 1\}$ such that $\mu(\{(x, y) \in R \mid f(x, y) = b\}) \geq (1 - \varepsilon)\mu(R)$.

In randomized communication complexity, Alice and Bob can toss coins, and the protocols may err with some small probability.

Definition 1. Let $0 \leq \varepsilon \leq \frac{1}{2}$. A probabilistic communication protocol \mathcal{P} is ε -correct for f if for all $(x, y) \in X \times Y$, $\text{Prob}(\mathcal{P}(x, y) \neq f(x, y)) \leq \varepsilon$, where the probability is taken over the randomness of \mathcal{P} .

The randomized communication complexity is the cost of the best probabilistic ε -correct protocol that computes f , and is denoted by $R_\varepsilon(f)$. We usually consider the randomness used in communication protocols explicitly, that is, we assume that before the execution of the protocol, each player receives a random string r_A and r_B from sets $R_A, R_B \subseteq \{0, 1\}^*$. If the randomness is shared, then $R_A = R_B$ and $r_A = r_B$. We denote by $R_\varepsilon^{\text{pub}}(f)$ the randomized communication complexity with shared randomness.

In the distributional model, inputs are chosen according to a distribution, but protocols are deterministic, and can err on some inputs.

Definition 2. Let $0 \leq \varepsilon \leq \frac{1}{2}$ and μ a distribution over the inputs $X \times Y$. A distributional communication protocol \mathcal{P} is (μ, ε) -correct if $\text{Prob}_\mu(\mathcal{P}(x, y) \neq f(x, y)) \leq \varepsilon$.

We denote by $D_\varepsilon^\mu(f)$ the cost of best distributional (μ, ε) -correct protocol. The distributional communication complexity $D_\varepsilon(f)$ is $\max_\mu D_\varepsilon^\mu(f)$. We will consider the special case where μ ranges over rectangular (or product) distributions only. These are distributions μ over $X \times Y$ such that $\mu = \mu_1 \otimes \mu_2$ where μ_1 is a distribution over X and μ_2 a distribution over Y . In this special case, we denote the communication complexity by $D_\varepsilon^\square(f)$. In the general case, Yao's minmax theorem states that distributional communication complexity is equivalent to randomized communication complexity with shared randomness.

Theorem 1. [23] For any $f : X \times Y \rightarrow \{0, 1\}$ and $\varepsilon > 0$, $D_\varepsilon(f) = R_\varepsilon^{\text{pub}}(f)$.

2.2 Kolmogorov Complexity

We recall some basic definitions and properties of Kolmogorov complexity that we use extensively in the rest of the paper [11].

Definition 3

- A set of strings is called prefix free if no string in the set is a prefix of another.
- Let φ be a universal Turing machine and \mathbb{P} a prefix free set. The prefix free Kolmogorov complexity of a string x given y with respect to φ, \mathbb{P} is $K_\varphi(x|y) = \min\{|p| : p \in \mathbb{P} \text{ and } \varphi(p, y) = x\}$.

If ϵ is the empty string, we just write $K_\varphi(x)$ for $K_\varphi(x|\epsilon)$. Henceforth, we fix a universal Turing machine φ , a prefix free set \mathbb{P} , and write K instead of K_φ .

Proposition 1

1. For any finite set X and string σ , there exists a constant c such that for all $x \in X$ $K(x|\sigma) \leq \log |X| + c$.
2. For any finite set X and string σ , there exists an element $x \in X$ such that $K(x|\sigma) \geq \log |X|$. Such elements are called incompressible.
3. There exists a constant c such that, for all x, y, σ : $K(x|\sigma) \leq K(x|y, \sigma) + K(y) + c$.

Corollary 1. *Let X and Y be finite sets. For $x \in X$ and $y \in Y$, $\forall \sigma$, if $K(x, y|\sigma) \geq \log |X| + \log |Y|$ then $K(x|y, \sigma) \geq \log |X|$ and $K(y|x, \sigma) \geq \log |Y|$.*

Such strings x, y are called independent Kolmogorov-incompressible strings.

Given a distribution on strings, they can be coded using the Shannon-Fano code. The next proposition shows how this translates to Kolmogorov complexity.

Proposition 2

1. *Fix a finite set X and a probability distribution μ over X . There exists a constant $c \geq 0$ such that for all $\sigma \in \{0, 1\}^*$, and for all $x \in X$ such that $\mu(x) \neq 0$, $K(x|\sigma) \leq \log(\frac{1}{\mu(x)}) + c$.*
2. *Fix a finite set X and a probability distribution μ over X . For all $\sigma \in \{0, 1\}^*$, there exists $x \in X$ such that $\mu(x) \neq 0$ and $K(x|\sigma) \geq \log(\frac{1}{\mu(x)})$.*

We also use the following asymptotic approximation.

Proposition 3. *Let $n \in \mathbb{N}$ and $0 < \varepsilon < 1$. Then $\log \binom{n}{\lfloor \varepsilon n \rfloor} \sim nH_2(\varepsilon)$, where $H_2(\varepsilon)$ is the entropy of a random variable following a Bernoulli distribution with parameter ε .*

3 Lower Bounds

3.1 Main Theorem in the Deterministic Case

We give a general lower bound on deterministic communication complexity in terms of Kolmogorov mutual information between an input of the problem and the transcript of the protocol. The mutual information between x and y is $K(x) - K(x|y)$, which can be interpreted as how much information about x is gained when y is given, compared to when it is not given. It is a measure of the information that y contains on x . Buhrman et al. have also used Kolmogorov mutual information to analyze the communication of individual instances, but they consider the mutual information between the players' inputs [13,14].

Theorem 2. *Fix $f : X \times Y \rightarrow \{0, 1\}$ and \mathcal{P} an optimal deterministic protocol for f . Denote by $T(x, y)$ the transcript of \mathcal{P} on input (x, y) . Then $\forall \sigma \in \{0, 1\}^*$,*

$$D(f) \geq \max_{(x,y) \in X \times Y} K(x, y|\sigma) - K(x, y|T(x, y), \sigma).$$

Proof. Fix $(x, y) \in X \times Y$. By Propositions 3 and 1, $K(x, y|\sigma) \leq K(x, y|T(x, y), \sigma) + K(T(x, y))$, and $K(T(x, y)) \leq |T(x, y)| \leq D(f)$.

We show that our method implies the corruption lower bound 4.

Definition 4. *For $f : X \times Y \rightarrow \{0, 1\}$, a distribution μ over $X \times Y$ and $\varepsilon > 0$, define $mono_\mu(f, \varepsilon) = \max\{\mu(S) \mid S \text{ is a } (\mu, \varepsilon)\text{-monochromatic rectangle for } f\}$.*

Theorem 3 ([4]). *For $f : X \times Y \rightarrow \{0, 1\}$, μ a distribution over $X \times Y$ and $1/2 > \varepsilon > 0$, $D_\varepsilon^\mu(f) \geq \log \frac{1}{mono_\mu(f, 2\varepsilon)}$.*

Proof. Fix an (μ, ε) -correct protocol \mathcal{P} for f , and by Proposition 2, let (x^*, y^*) be a pair of inputs such that $K(x^*, y^* | \mu, \mathcal{P}, f) \geq \log \frac{1}{\mu(x^*, y^*)}$. Recall that \mathcal{P} induces a partition \mathcal{R} of the input into rectangles. For $S \subseteq X \times Y$, let $Err(S) = \{(x, y) \in S | \mathcal{P}(x, y) \neq f(x, y)\}$, and $\tilde{\mathcal{R}} = \{S \in \mathcal{R} | \mu(Err(S)) > 2\varepsilon\mu(S)\}$. Let $E = \bigcup_{S \in \tilde{\mathcal{R}}} S$ be the inputs not in $(\mu, 2\varepsilon)$ -monochromatic rectangles.

Notice that $\mu(E) = \sum_{S \in \tilde{\mathcal{R}}} \mu(S) \leq 1/2$, otherwise $\mu(Err(X \times Y)) > \varepsilon$, which contradicts the correctness of the protocol. If $(x^*, y^*) \in E$, one could encode (x^*, y^*) by giving an index in E , using the probability distribution induced by μ on E and a Sannon-Fano code (Proposition 1), so $\log \frac{1}{\mu(x^*, y^*)} \leq K(x^*, y^*) \leq \log \frac{\mu(E)}{\mu(x^*, y^*)} \leq \log \frac{1}{2\mu(x^*, y^*)}$, a contradiction. Hence, $(x^*, y^*) \notin E$.

Since $(x^*, y^*) \notin E$, the transcript $T = T(x^*, y^*)$ determines a rectangle R containing (x^*, y^*) such that $\mu(Err(R)) < 2\varepsilon$. By definition, $\mu(R) \leq mono_\mu(f, 2\varepsilon)$. Given T , one can encode (x^*, y^*) by giving its index in R , using the probability distribution induced by μ on R and a Sannon-Fano code. Therefore, $K(x^*, y^* | \mu, \mathcal{P}, f, T) \leq \log \frac{\mu(R)}{\mu(x^*, y^*)} \leq \log \frac{mono_\mu(f, 2\varepsilon)}{\mu(x^*, y^*)}$. Using Theorem 2 with $\sigma = (\mu, \mathcal{P}, f)$, we get $D_\varepsilon^\mu(f) \geq \log \frac{1}{mono_\mu(f, 2\varepsilon)}$, as claimed.

3.2 The Randomized Case: A Kolmogorov Alternative to Yao’s Min-Max Principle

We show how to derive a deterministic protocol from a randomized one, with the same complexity and performance in terms of errors. In the communication complexity model, Alice and Bob have full computational power, so the players can choose an incompressible string in advance (which is in general not computable), and simulate a randomized protocol \mathcal{P} using this string for randomness.

We denote by \mathcal{P}^{r_A, r_B} the deterministic protocol obtained by executing a randomized protocol \mathcal{P} with fixed random strings (r_A, r_B) . This protocol makes errors for some inputs, but the next lemma shows that using incompressible strings, the distribution of errors in the resulting protocol has good properties.

Lemma 1. *Let \mathcal{P} be an ε -correct randomized protocol for $f : X \times Y \rightarrow \{0, 1\}$ and μ a probability distribution on $X \times Y$. For all $S \subseteq X \times Y$, we define $Err_{r_A, r_B}(S) = \{(x, y) \in S : \mathcal{P}^{r_A, r_B}(x, y) \neq f(x, y)\}$. Fix r_A^* et r_B^* such that $K(r_A^*, r_B^* | \mu, \mathcal{P}, S) \geq \log(|R_A||R_B|)$. Then $\mu(Err_{r_A^*, r_B^*}(S)) \leq 2\varepsilon\mu(S)$.*

Proof. Let \tilde{R} denote the bad random strings: $\tilde{R} = \{\mu(r_A, r_B) : \mu(Err_{r_A, r_B}(S)) > 2\varepsilon\mu(S)\}$. We will prove that $|\tilde{R}| < \frac{|R_A||R_B|}{2}$. This is sufficient to conclude that $(r_A^*, r_B^*) \notin \tilde{R}$; otherwise, one could compute it by giving an index in \tilde{R} , which contradicts the assumption $K(r_A^*, r_B^* | \mu, \mathcal{P}, S) \geq \log(|R_A||R_B|)$. \mathcal{P} being ε -correct, we get by summing over $R_A \times R_B$ $\sum_{r_A, r_B} \mu(Err_{r_A, r_B}(S)) \leq |R_A||R_B|\varepsilon\mu(S)$. On the other hand, $\sum_{r_A, r_B} \mu(Err_{r_A, r_B}(S)) \geq \sum_{\tilde{R}} \mu(Err_{r_A, r_B}(S)) > 2\varepsilon\mu(S)|\tilde{R}|$. Combining the two inequalities, we obtain $|\tilde{R}| < \frac{|R_A||R_B|}{2}$.

Compared to the original proof, what we gain by using the Kolmogorov alternative is that we do not require distributional complexity. In distributional

complexity, we have to analyse the behavior of deterministic protocols with respect to a distribution μ over the inputs. Here, by choosing a single random string, and an independent Kolmogorov random hard instance, we analyze a deterministic algorithm acting on a single input. Theorem 4 generalizes Theorem 2 to the randomized case.

Theorem 4. Fix $f : X \times Y \rightarrow \{0, 1\}$ and \mathcal{P} an optimal randomized ε -correct protocol for f . If $T(x, y, r_A, r_B)$ is the transcript of \mathcal{P}^{r_A, r_B} on input (x, y) , then for all $S \subseteq X \times Y$, $(r_A, r_B) \in R_A \times R_B$ and $\sigma \in \{0, 1\}^*$,

$$R_\varepsilon(f) \geq \max_{(x,y) \in S} K(x, y|\sigma) - K(x, y|T(x, y, r_A, r_B), \sigma).$$

Proof. Fix r_A and r_B in \mathcal{P} . By Proposition 1, $R_\varepsilon(f) \geq K((Tx, y, r_A, r_B)|\sigma)$. Using Proposition 3, $K(x, y|\sigma) \leq K(x, y|T(x, y, r_A, r_B), \sigma) + K(T(x, y, r_A, r_B))$. So $R_\varepsilon(f) \geq K(x, y|\sigma) - K(x, y|T(x, y, r_A, r_B), \sigma)$.

4 Applications

4.1 The Hidden Matching Problem

In this section, we study the communication complexity of the hidden matching problem. This relation was introduced to show a gap between randomized and quantum communication complexity 5. The following theorem is the randomized lower bound for the hidden matching problem.

Definition 5. In the Hidden Matching problem $HM_n(x, M)$, Alice receives a string $x \in \{0, 1\}^n$, and Bob a matching M on n vertices. At the end, Bob has to output a triple (i, j, b) such that $x_i \oplus x_j = b$ and $(i, j) \in M$.

Theorem 5. 5 $R_\varepsilon^{A \rightarrow B}(HM_n) \geq \Omega(\sqrt{n})$.

Proof (Sketch). The complete proof will appear in the full version. As in 5, the idea is that each output provides a linear equation $x_i \oplus x_j = b$. Since the protocol is one-way, a single transcript can be used to get many equations of this form, providing substantial information on x .

We fix \mathcal{M} a set of n disjoint matchings. Using Proposition 2 and Corollary 1, we pick independent incompressible input x^* , randomness r_A^*, r_B^* and a subset $\mathcal{M}^* \subset \mathcal{M}$ of size \sqrt{n} . Incompressibility plays several key roles:

- get a lower bound on the complexity x^* ,
- prove that $\Omega(\sqrt{n})$ linearly independent equations can be retrieved from \mathcal{M}^* ,
- prove that the protocol induces few errors in the equations.

To use Theorem 4, we have to prove an upper bound on $K(x^*|T(x^*))$. Using the stated properties, the algorithm goes as follows:

1. Simulate the one-way protocol using message $T(x^*)$ on every $M \in \mathcal{M}^*$,
2. Correct the errors. The errors are given as an auxiliary input,
3. Use the equations to learn $\Omega(\sqrt{n})$ coordinates of x . The $n - \sqrt{n}$ remaining ones are given as an auxiliary input.

The size of the auxiliary input to this program is $n - \sqrt{n} + H_2(2\varepsilon)\sqrt{n}$, which proves the theorem.

4.2 VC Dimension and Shatter Coefficients Lower Bounds

In this section, we consider a general lower bound on one-way communication complexity. This lower bound was previously proved using combinatorial techniques [6] and later re-proved and extended using information theory techniques [7]. Our proof uses only elementary counting arguments.

To any function $f : X \times Y \rightarrow \{0, 1\}$, we associate the communication matrix $M_f(x, y) = f(x, y)$. We identify M_f (or any submatrix) with the set of its rows, which we think of as boolean strings. The VC dimension of M_f is the size of the largest set $Y_0 \subseteq Y$ such that there exists some $X_0 \subseteq X$ of size $2^{|Y_0|}$ and $M_f|_{X_0, Y_0} = \{0, 1\}^{|Y_0|}$. For $l \geq VC(M_f)$, the l -th shatter coefficient of M_f , denoted by $SC(l, M_f)$, is the size of the largest set $X_0 \subseteq X$ such that there exists some $Y_0 \subseteq Y$ of size l and all rows of $M_f|_{X_0, Y_0}$ are different. A witness for $SC(l, M_f)$ is a set $S \subseteq X \times Y$ such that if $S=U \times V$, $|V|=l$ and $|U|=SC(l, M_f)$ and all rows in S are different.

Theorem 6. [6,7] *For every function $f : X \times Y \rightarrow \{0, 1\}$, there exists a constant $c > 0$ such that for every $l > VC(M_f)$:*

$$R_\epsilon^{A \rightarrow B}(f) \geq VC(M_f)(1 - (1 + c)H_2(2\epsilon))$$

$$R_\epsilon^{A \rightarrow B}(f) \geq \log(SC(M_f, l)) - l(1 + c)H_2(2\epsilon)$$

Proof. Notice that $\log(SC(l, M_f))=VC(M_f)$ for $l=VC(M_f)$. Therefore, we just have to prove the second point. Fix an optimal ϵ -correct randomized one-way protocol \mathcal{P} for f . Let $S = U \times V$ be a witness for $SC(l, M_f)$. Pick $x^* \in U$, and $(r_A^*, r_B^*) \in R_A \times R_B$ incompressible. By Corollary 1, $K(r_A^*, r_B^* | f, \mathcal{P}, S, x^*) \geq \log |R_A| + \log |R_B|$ and $K(x^* | r_A^*, r_B^* f, \mathcal{P}, S) \geq \log |U|$. Let $S' = \{x^*\} \times V$. By Lemma 1, $|Err_{r_A^*, r_B^*}(\{x^*\} \times V)| < 2\epsilon |\{x^*\} \times V|$.

Let $T(x, r_A)$ denote the transcript of the protocol \mathcal{P}^{r_A, r_B} on input x . We define an algorithm that computes any $x \in U$ knowing $T(x, r_A^*)$ and r_B^* .

1. Simulate $\mathcal{P}^{r_A^*, r_B^*}(x, y)$ using $T(x, r_A^*)$ for every $y \in Y$.
2. Correct the errors in $\{x\} \times V$. The set of errors is given as an auxiliary input of the program.
3. Compare the obtained row with every row of S . As they are all different, only one corresponds to x .

This program uses $\log \binom{l}{2\epsilon l} \sim lH_2(2\epsilon)$ bits to describe the set of errors (Proposition 3). Therefore, there exists a constant c such that $K(x^* | T(x^*, r_A^*), r_B^*) \leq l(1 + c)H_2(2\epsilon)$. By Theorem 4, we get $R_\epsilon^{A \rightarrow B}(f) \geq \log |U| - l(1 + c)H_2(2\epsilon)$. Since $|U| = SC(l, M_f)$, this suffices to conclude.

5 One Way versus Simultaneous Messages

In the multiparty number-in-hand model, n players have to compute an n -variable function $f : X_1 \times \dots \times X_n \rightarrow \{0, 1\}$. In a simultaneous message protocol,

every player receives an input and sends a message to a referee, who outputs the value of the function. $D^{X_1 \parallel \dots \parallel X_n}(f)$ denotes the complexity of f in this model.

We wish to compare this with two-player protocols in which one player receives one variable, say $x_i \in X_i$, and the other one receives all the other variables. The complexity of f in this model is $D^{X_i \rightarrow X^{-i}}(f)$. Here, we compare the distributional versions of simultaneous and one-way communication, restricted to rectangular distributions. We improve the previous bound [7] on the error probability, from $\sum_i H_2(\varepsilon_i)$ to $\sum_i \varepsilon_i$.

Theorem 7. [7] Fix $f : X_1 \times \dots \times X_n \rightarrow \{0, 1\}$. Then for $\varepsilon \geq (1+1/n) \sum_{i=1}^n \varepsilon_i$, $D_{\varepsilon}^{\parallel, X_1 \parallel \dots \parallel X_n}(f) \leq \sum_{i=1}^n D_{\varepsilon_i}^{\parallel, X_i \rightarrow X^{-i}}(f)$.

Proof (Sketch). Fix a probability distribution on the inputs and n one-way protocols. We design a simultaneous protocol, where each player sends to the referee the message he would have sent in the one-way protocol. The referee outputs the value on which most one-way protocols agree.

In analyzing the errors made by this protocol, the crucial step is to prove that most inputs on which the simultaneous protocol is wrong were already errors in at least one one-way protocol. This relies on the fact that the distribution is product. By combinatorial arguments, we give an upper bound the measure of the set of remaining inputs on which the simultaneous protocol is wrong whereas all one-way protocols are correct.

Acknowledgments

We wish to thank Troy Lee for many useful discussions. The research was supported by the EU 5th framework program QAP, by the French ANR Blanc AlgoQP, and by French ANR Defis program ANR-08-EMER-012 QRAC.

References

1. Bar-Yossef, Z., Jayram, T.S., Kumar, R., Sivakumar, D.: An information statistics approach to data stream and communication complexity. *J. Comput. Syst. Sci.* 68, 702–732 (2004)
2. Jain, R., Klauck, H., Nayak, A.: Direct product theorems for classical communication complexity via subdistribution bounds: extended abstract. In: *Proc. of the 40th Annual ACM Symposium on Theory of Computing (STOC)*, pp. 599–608 (2008)
3. Linial, N., Shraibman, A.: Lower bounds in communication complexity based on factorization norms. *Random Structures and Algorithms* (to appear)
4. Beame, P., Pitassi, T., Segerlind, N., Wigderson, A.: A strong direct product theorem for corruption and the multiparty communication complexity of disjointness. *Computational Complexity* 15, 391–432 (2006)
5. Bar-Yossef, Z., Jayram, T.S., Kerenidis, I.: Exponential separation of quantum and classical one-way communication complexity. *SIAM J. Comput.* 38, 366–384 (2008)
6. Kremer, I., Nisan, N., Ron, D.: On randomized one-round communication complexity. *Computational Complexity* 8, 21–49 (1999)

7. Bar-Yossef, Z., Jayram, T.S., Kumar, R., Sivakumar, D.: Information theory methods in communication complexity. In: Proc. of the 17th Annual IEEE Conference on Computational Complexity (CCC), pp. 93–102 (2002)
8. Yao, A.C.C.: Some complexity questions related to distributive computing (preliminary report). In: Proc. of the 11th Annual ACM Symposium on Theory of Computing (STOC), pp. 209–213. ACM, New York (1979)
9. Shannon, C.: A mathematical theory of communication. *Bell System Technical Journal* 27, 379–423, 623–656 (1948)
10. Jayram, T.S., Kumar, R., Sivakumar, D.: Two applications of information complexity. In: Proc. of the 35th Annual ACM Symposium on Theory of Computing (STOC), pp. 673–682. ACM, New York (2003)
11. Li, M., Vitanyi, P.M.B.: *An Introduction to Kolmogorov Complexity and Its Applications*. Springer, Berlin (1993)
12. Buhrman, H., Jiang, T., Li, M., Vitanyi, P.: New applications of the incompressibility method: Part ii. *Theoretical Computer Science* 235, 59–70 (2000)
13. Buhrman, H., Klauck, H., Vereshchagin, N., Vitányi, P.: Individual communication complexity. *J. Comput. Syst. Sci.* 73, 973–985 (2007)
14. Buhrman, H., Koucký, M., Vereshchagin, N.: Randomised individual communication complexity. In: Proc. of the 23rd Annual IEEE Conference on Computational Complexity (CCC), pp. 321–331 (2008)
15. Laplante, S., Magniez, F.: Lower bounds for randomized and quantum query complexity using kolmogorov arguments. *SIAM J. Comput.* 38, 46–62 (2008)
16. Gavinsky, D., Kempe, J., Kerenidis, I., Raz, R., de Wolf, R.: Exponential separations for one-way quantum communication complexity, with applications to cryptography. *SIAM J. Comput.* 38, 1695–1708 (2008)
17. Lee, T., Shraibman, A.: Disjointness is hard in the multi-party number-on-the-forehead model. In: Proc. of the 23rd Annual IEEE Conference on Computational Complexity (CCC), pp. 81–91 (2008)
18. Lee, T., Shraibman, A., Špalek, R.: A direct product theorem for discrepancy. In: Proc. of the 23rd Annual IEEE Conference on Computational Complexity (CCC), pp. 71–80 (2008)
19. Razborov, A.: Quantum communication complexity of symmetric predicates. *Izvestiya: Mathematics* 67, 145–159 (2003)
20. Raz, R.: Fourier analysis for probabilistic communication complexity. *Computational Complexity* 5, 205–221 (1995)
21. Degorre, J., Kaplan, M., Laplante, S., Roland, J.: The communication complexity of non-signaling distributions. Technical Report quant-ph/0804.4859, arXiv e-Print archive (2008)
22. Kushilevitz, E., Nisan, N.: *Communication Complexity*. Cambridge University Press, New York (1997)
23. Yao, A.C.C.: Lower bounds by probabilistic arguments (extended abstract). In: Proc. of the 24th Annual IEEE Symposium on Foundations of Computer Science (FOCS), pp. 420–428. IEEE, Los Alamitos (1983)

An Almost Totally Universal Tile Set

Grégory Lafitte^{1,*} and Michael Weiss^{2,**}

¹ Laboratoire d'Informatique Fondamentale de Marseille (LIF),
CNRS – Aix-Marseille Université,
39, rue Joliot-Curie, F-13453 Marseille Cedex 13, France

² Università degli Studi di Milano,
Bicocca Dipartimento di Informatica, Sistemistica e Comunicazione,
336, Viale Sarca, 20126 Milano, Italy

Abstract. Wang tiles are unit size squares with colored edges. In this paper, we approach one aspect of the study of tilings computability: the quest for a universal tile set. Using a complex construction, based on Robinson's classical construction and its different modifications, we build a tile set \mathfrak{V} (pronounced *ayin*) which *almost always* simulates any tile set. By way of Banach-Mazur games on tilings topological spaces, we prove that the set of \mathfrak{V} -tilings which do not satisfy the universality condition is meager in the set of \mathfrak{V} -tilings.

1 Introduction

Wang was the first to introduce in [Wan61] the study of tilings with colored tiles where a tile is a unit size square with colored edges. Two tiles can be assembled if their common edge has the same color. To tile consists in assembling tiles from a tile set (a finite set of tiles) on the grid \mathbb{Z}^2 .

Since Berger [Ber66] it is known that Wang tilings can simulate Turing machines. As a model of computation, tilings raise computability questions. One of the first, related to most models of computation, is the existence of universality. To approach such a problem we need a proper notion of reduction. In [LW07], a first approach to reduction, and by extension, universality, was given. Intuitively, a tiling P *simulates* a tiling Q if the tiles of Q can be encoded with macro-tiles of P .

This notion of simulation was then improved in [LW08a] (a close definition is also introduced in [DRS08]) to obtain simulations between tile sets. A tile set τ *totally simulates* a tile set τ' if any τ' -tiling is simulated by a τ -tiling and if any τ -tiling simulates a τ' -tiling. In [LW08a], it has been proved that there exists a tile set that totally simulates any periodic tile set, *i.e.*, tile sets that generate at least one periodic tiling (a tiling invariant by translation of two independent vectors). The question of the existence of a totally universal tile set was asked: does there exists a tile set totally simulating any tile set that tiles the plane? Because

* This author has been supported by the French ANR grant *Sycomore*.

** This author has been supported by the Swiss FNS post-doc grant.

of the amount of properties that such a tile set would have (having maximal Kolmogorov complexity [DLS01], non-recursivity [Han74, Mye74], self-similarity of any kind [DRS08], Turing universality, invariance by recursive modification [LW08b], ...), it has been conjectured that it does not exist.

In this paper, we combine some of the most complex constructions on tilings to build a tile set \mathfrak{V} which is almost totally universal: almost all \mathfrak{V} -tilings simulate at least one tiling for any tile set (by *almost* we mean that the subset of \mathfrak{V} -tilings that do not satisfy this property is a meager set in the set of \mathfrak{V} -tilings). And therefore, \mathfrak{V} has the particularity of having *almost always*¹ all the properties enumerated previously. The construction of \mathfrak{V} uses different technical tools and is mainly based on the aperiodic and self-similar tile set of Robinson within which simulations of Turing machines can be carried out. A detailed explanation of this construction can be found in [Rob71, Han74, Mye74, AD96, DLS01, DLS04]. In [LW08a], and in [DRS08] (ingeniously avoiding the use of Robinson's tilings construction), it has been shown how a Turing machine can be used to simulate a tile set, in the sense that the Turing machine produces space \times time diagrams isomorphic to the tiles of a tile set. With this simulation of Turing machines, simulation of tile sets in Robinson's tiling is made possible. The other tool used for the construction is synchronization. This principle was first used by Hanf and Myers [Han74, Mye74] to build a non-recursive tile set, *i.e.*, a tile set that generates only tilings that cannot be defined by a recursive function. We show how to make a synchronization between squares of Robinson's construction in a new way. All these different tools make possible the construction of an almost totally universal tile set.

The last step consists in proving the *almost* part. One of the main tool to prove the meagerness of a set is to use topological games like Banach-Mazur games [Oxt57]. This perfect information game is played by two players on a topological space. A classical result on Banach-Mazur games shows that if Player II has a winning strategy, then A is meager (or, in an equivalent way, the set $X \setminus A$ is residual).

In [LW08c], a topological study of tilings has been made and games *à la* Banach-Mazur on them have been introduced. These games are played on two topological spaces: the Besicovitch one (where the distance between two tilings is defined as the asymptotic proportion of different tiles between them) and the Cantor one (where the distance between two tilings is related to the biggest pattern centered around the origin that they have in common). In this paper we restrict these games to the set of tilings generated by a tile set and we prove, using these games, that the tile set \mathfrak{V} is almost totally universal.

In the following section, we recall the basic notions concerning tilings and the notions of simulation between tile sets. We also recall the two main topological spaces that can be used on the set of tilings generated by a tile set. Then, in section 3, we show how a synchronization can be made between the different squares in Robinson's construction. In the last section, we build the tile set \mathfrak{V} and show that it is almost totally universal.

¹ *Almost always* means that almost all of its tilings have the properties.

2 Basic Notions

2.1 Tilings and Simulations

We start by recalling the basic notions of tilings. A tile is an oriented unit size square with colored edges from C , where C is a finite set of colors. A tile set is a finite set of tiles. To tile consists in placing the tiles of a given tile set on the grid \mathbb{Z}^2 such that two adjacent tiles share the same color on their common edge. Since a tile set can be described with a finite set of integers, we can enumerate the tile sets, and τ_i designates the i^{th} tile set.

Let τ be a tile set. A tiling P generated by τ is called a τ -tiling. It is associated to a tiling function f_P where $f_P(x, y)$ gives the tile at position (x, y) in P . When we say that we superimpose the tiles of a tile set τ on the tiles of a tile set τ' , we mean that for any tile $t \in \tau$ and any tile $t' \in \tau'$, we build a tile $u = t \times t'$ where the colors of the sides of u are the cartesian product of the colors of the sides of t and t' . Then two tiles $u_1 = t_1 \times t'_1$ and $u_2 = t_2 \times t'_2$ match if and only if t_1 and t_2 match and t'_1 and t'_2 match.

Different notions of reductions have been introduced in [LW07]. We recall some of the basic notions here. A pattern is a finite tiling. If it is generated by τ , we call it a τ -pattern. We say that a τ -tiling P *simulates* a τ' -tiling Q if there exist two integers a, b and an application R from the $a \times b$ τ -patterns to the tiles of τ' and if we can cut regularly P in rectangular patterns of size $a \times b$ such that if we replace these rectangular patterns in P by their corresponding tiles given by R we obtain Q . One can see that P does with *macro-tiles*, *i.e.*, rectangular patterns which represent the tiles of another tile set, what Q does with tiles. We denote the reduction by $Q \triangleleft^R P$. We generalize this notion to simulations between a set of tilings and a tile set: a set of τ -tilings A *totally simulates* a tile set τ' if there exist $a, b \in \mathbb{Z}$ and a reduction R from the $a \times b$ patterns of τ to the tiles of τ' such that for any τ' -tiling Q , there exists a τ -tiling $P \in A$ such that $Q \triangleleft^R P$, and such that for any τ -tiling $P \in A$, there exists a τ' -tiling Q such that $Q \triangleleft^R P$. We denote it by $\tau \triangleleft A$ (or $\tau' \triangleleft^R A$ to specify the reduction R). If A corresponds to the whole set of τ -tilings, then we say that τ simulates τ' , and we denote it by $\tau' \triangleleft \tau$.

From this pseudometric we obtain a notion of universality: we say that a set of τ -tilings A is *totally universal* if $\tau' \triangleleft A$ for any tile set τ' that tiles the plane. If A corresponds to the whole set of τ -tilings, then we say that τ is totally universal. The existence of such a tile set is still open. In this paper we aim at constructing a tile set which is *almost always* totally universal. To have a clear definition of *almost* we recall some notions of topology and topological games. A deeper study of these topological spaces can be found in [LW08c, BDJ08].

The topologies defined in the following subsections are topologies used in cellular automata and adapted to tilings. Different definitions of these topologies can be given and we present here the restrictive case where the distances are defined only between tilings generated by the same tile set.

2.2 The Besicovitch and Cantor Topologies on Tilings

The first metric we introduce is a metric *à la* Besicovitch. This metric deals with the whole tiling and gives the asymptotic proportion of different tiles between two tilings. For two τ -tilings P and Q , we call P_n and Q_n the square patterns of size $2n + 1$ centered around the origin. The distance $d_B(P, Q)$ is given by:

$$\delta_B(P, Q) = \limsup_{n \rightarrow \infty} \frac{\#\{(x, y) \mid f_{P_n}(x, y) \neq f_{Q_n}(x, y)\}}{(2n + 1)^2}.$$

Therefore, the Besicovitch distance between two τ -tilings corresponds to the asymptotic proportion of different tiles between them. This is a pseudometric on the set of tilings generated by a tile set. We can obtain a metric by adding the condition that two tilings are equivalent if the distance between them is 0. Two tilings not generated by the same tile set are at distance 1. We obtain a topological space by defining the open sets as the balls $\mathcal{B}_B(Q, \epsilon)$, *i.e.*, all tilings at distance at most ϵ of Q .

The second metric, the Cantor one, deals with the local structure of the tilings while the Besicovitch one deals with their global behavior. We first define the function $p : \mathbb{N} \rightarrow \mathbb{Z}^2$ such that $p(0) = (0, 0)$, $p(1) = (0, 1)$, $p(2) = (1, 1)$, $p(3) = (1, 0) \dots$ and p keeps having the behavior of a spiral afterward. The metric d_C between two τ -tilings P and Q is defined as $d_C(P, Q) = 2^{-i}$, where i is the smallest integer such that $f_P(p(i)) \neq f_Q(p(i))$, *i.e.*, i is the size of the greatest common pattern of P and Q centered around the origin.

d_C is a metric on the set of τ -tilings. As before, we can obtain naturally a topological space by defining the open sets as the balls $\mathcal{B}_C(Q, \epsilon)$, *i.e.*, all tilings at distance at most ϵ of Q . One can note that in Cantor topology, the set of tilings having in common the same pattern centered around the origin is a clopen set [LW08c, BDJ08].

2.3 Games on Tilings

Now that we have defined notions of topologies on sets of tilings generated by a tile set, the natural next step for studying these sets is to consider infinite games on tilings. In [LW08c], the following definitions of Banach-Mazur games on tilings have been given:

Let X be a set of tilings generated by a tile set and C be a subset of X .

The first game $G(X, C)_B$ is played on Besicovitch topology and has the following rules: Player I chooses a τ -tiling P_1 and an integer n_1 . Player II chooses a tiling $P_2 \in \mathcal{B}_B(P_1, 1/n_1)$ and chooses an integer $n_2 > n_1$ and so on. Player II wins the game if $\bigcap_{n > 1} \mathcal{B}_B(P_i, 1/n_i) \in C$.

*The second game $G(X, C)_C$ is played on Cantor topology and has the following rules: Player I chooses a square pattern A_1 centered around the origin. Player II chooses a square pattern A_2 which is an extension of A_1 , *i.e.*, the tiling function of A_2 restricted to the domain of A_1 is the tiling function of A_1 , and so on. From the sequence of patterns $\{A_i\}$ we can obtain an infinite tiling P . Player II wins the game if $P \in C$.*

The main application of Banach-Mazur games is the study of meager sets. A classical topological result is that a subset C of X is meager, *i.e.*, is the union of countably many nowhere dense subsets, if and only if Player II has a winning strategy for the game $G(X, X \setminus C)$. *Meagerness* is thus a topological notion of *small* or *negligible* subsets. We obtain the notion of *almost total universality*: a tile set τ is *almost totally universal* if there exists a set of τ -tilings A which is totally universal and such that A is residual in the set of τ -tilings in both Besicovitch and Cantor topologies.

Therefore, an almost totally universal tile set is a tile set totally universal up to a meager set: just a small subset (of its tilings) prevents it to be totally universal. In the following section, we explain some constructions needed to build an almost totally universal tile set.

3 Synchronization within Robinson's Construction

In this section, we show how to synchronize squares of Robinson's tiling (we refer the reader to [AD96] for an explanation of this construction). By synchronization, we mean that any square of any level works on an initial segment of an infinite input. Synchronization was first introduced in [Han74, Mye74] and used in [DLS01]. We propose our own synchronization, adapted for our purpose.

3.1 Synchronization between Squares of Same Level

The first goal to achieve, is to prove that all the squares of a same level have the same information, *i.e.*, any square of a certain level in Robinson's construction have the same input word w on their first line. Since two neighbor squares, either vertical or horizontal, can share the information they have on their facing sides then we need to prove that we can obtain a square which has on its four sides the same input word w . We just need to pass the bits of the input word from the south side to the west side. Then, we can transmit these bits to the north and east sides.

The information going from the south to the west side will pass through three kinds of tiles: it first goes through a tile that transmits the information vertically, then passes a corner and finally goes through tiles that transmit the information horizontally until it reaches the west side. The only condition to add to be sure that all the bits will pass from the south to the west side (like in figure [I](#)) is to force any tile which is not obstructed (obstructions are colored in gray in figure [II](#)) to be one of the three kinds of tiles that transmit information. The obstructed tiles can either transmit vertical or horizontal information, or transmit nothing. Finally, neighbor squares of same level can check if they are computing on the same input word. Therefore all squares of a same level work on the same input word.

3.2 Synchronization between Levels

We now want to synchronize the input word between different levels, *i.e.*, that if w_i is the input word of the squares of level i then w_i is the central word of w_j

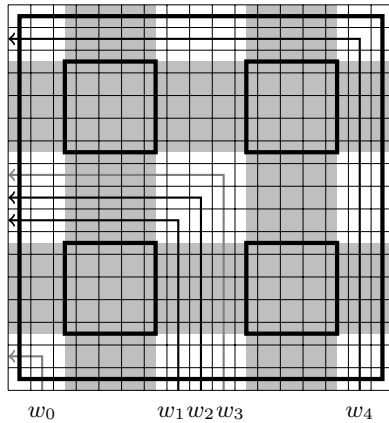


Fig. 1. The transmission of the bits of w from the south side to the west side

for any $j > i$, *i.e.*, there exists two words w_1 and w_2 of same length such that $w_j = w_1 w_i w_2$. In this way all squares of all levels obtain the same computation. We recall that in Robinson’s tilings, the squares of level even are colored in black and the squares of level odd are colored in light-gray.

We need to choose a square (the only one) that communicates its input word to the higher level. We give sixteen different labels to the black squares (one of them is labeled in gray) and two kinds of label for the light-gray squares. This is enough to guarantee that any black square of a level n , has a gray square of level $n - 1$ in its south-west corner (figure 2.a). This is this gray square who passes the information from its east side to the south side of the square of level n .

To pass the information, we use the induction process of figure 2.b. The same technique as before is used. We can do this since the number of columns between two neighbor squares is the same as the number of columns in a square. Then, with an induction process, we will pass all the bits from the east side of the gray square to the south side of the black square of higher level.

At the end of the process, the gray row contains the bits of w and the black square of upper level has access to this code and can compute on it. Therefore, any square computes on an initial segment of the same infinite input.

4 An Almost Totally Universal Tile Set

4.1 Description of the Construction

In this section we construct an almost totally universal tile set. We use three Turing machines M , N and P that we simulate in the synchronized construction explained previously. The three machines works on an infinite string $i_1 i_2 \dots$ where i_j is the code of a tile set of j tiles that tiles the plane. M checks if the input is well written. If not, M stops. We add another restriction to M : we want that the code of \mathfrak{V} appears as an input. The tile set \mathfrak{V} will have access to its own

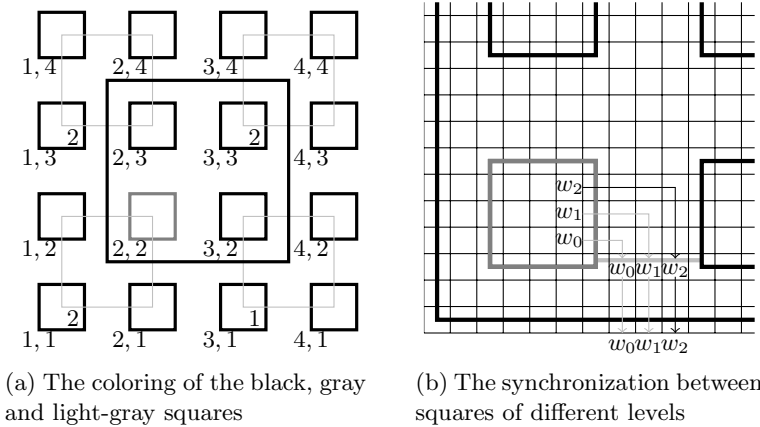


Fig. 2.

code. One can prove, using Kleene’s recursion theorem, that a tile set can have access to its own code (see [LW08b, DRS08]). Let m be the number of tiles of \mathfrak{Y} . M checks that the input contains two codes of tile sets of m tiles, and checks that one of them is the code of \mathfrak{Y} . Therefore, the input has to be of the following form: $i_1\$i_2\$ \dots i_m\$i'_m\$i_{m+1}\$ \dots$, where i'_m is the code of \mathfrak{Y} .

The second machine N checks for any n such that any of the τ_{i_j} ’s, $i_j < n$, can tile a square of size n . If there exists an integer m such that a tile set τ_{i_j} cannot tile a square of size m , then N stops.

The last machine P is a machine that simulates the tile sets of the input, *i.e.*, it generates space times diagrams isomorphic to the tiles of the tile sets (for more detailed explanations on simulation between tile sets, see [LW08a, DRS08]).

To start the simulations of these machines in our tilings, we first force that the only computation tile which exists in the squares of level 1 is the tile, say t_0 , representing the initial state of M , N and P . By synchronization, this means that any middle tile of the first line of any square of this construction corresponds also to this tile, and therefore, the computation will begin in any square. We now allow the completion of the first line of any square with tiles representing any letter from the alphabet $\{0, 1, \$\}$. We obtain a tiling where any first line of any square represents the central subword of a bi-infinite input $w \in \{0, 1, \$\}$ and all of these subwords contain in their middle the tile t_0 representing the initial states of the Turing machines.

In all squares of our construction, the computation on the same infinite input is carried out. If one of these machines reaches a final state, then the tiling remains incomplete. Therefore, if the tiling is complete, then M , N and P compute on a word of the form: $i_1\$i_2\$ \dots i_m\$i'_m\$i_{m+1}\$ \dots$, as stated before. Then P simulates any of the tile set i_j , and thus, \mathfrak{Y} totally simulates any tile set τ_{i_j} . Since the index of \mathfrak{Y} is given also in input, then P also simulates a \mathfrak{Y} -tiling. In fact, by transitivity of the simulation, it self-simulates infinitely many times. Each time \mathfrak{Y} self-simulates, it also simulates a set of tile sets $\{\tau_{i'_j}\}_{j>0}$ since it simulates a

\mathfrak{Y} -tiling that simulates this set. So, a \mathfrak{Y} -tiling simulates an infinite number of tile sets of n tiles for any n . Since the set of tile sets of n tiles is finite, and *a fortiori* the set of tile sets of n tiles that tiles the plane, then a \mathfrak{Y} -tiling must simulate infinitely many times some tile sets.

4.2 The Construction Gives an Almost Totally Universal Tile Set

We have obtained a tile set \mathfrak{Y} such that any \mathfrak{Y} -tiling simulates, for any n , with repetitions, an infinity of tile sets composed of n tiles. We now show that this tile set \mathfrak{Y} is almost totally universal.

Theorem 1. *The tile set \mathfrak{Y} is almost totally universal.*

Proof. Let A be the set of \mathfrak{Y} -tilings that simulate at least one tiling for any tile set and $B = \mathcal{T}_{\mathfrak{Y}} \setminus A$, where $\mathcal{T}_{\mathfrak{Y}}$ is the set of \mathfrak{Y} -tilings. A is totally universal. We show that A is residual in $\mathcal{T}_{\mathfrak{Y}}$ in both topologies:

We first show that A is residual in $\mathcal{T}_{\mathfrak{Y}}$ (in the Cantor topology) by showing that Player II has a winning strategy in the game $G(A, \mathcal{T}_{\mathfrak{Y}})_C$. In this game, Player I first chooses a \mathfrak{Y} -pattern centered around the origin. Player II extends this pattern and so on. Player II wants to obtain a final \mathfrak{Y} -tiling that simulates any tile set that tiles the plane. Player I wants to obtain a final tiling such that at least one tile set is never simulated.

Let $\Omega = \{\tau_1, \tau_2, \dots\}$ be the set of tile sets that tile the plane, and ordered by the number of their tiles first, and then by a lexicographic order of the colors of the tiles. The following strategy is, of course, not recursive since Ω is Π_1 . At step n , Player II wants to force the simulation of the n^{th} tile set of Ω . Let m_n be the \mathfrak{Y} -pattern played by Player I. Player II wants to force the code of τ_n to appear somewhere in the tiling. When done, by synchronization the final tiling has to simulate a τ_n -tiling. If the code of the tile set τ_n can be written on the input word, then Player II writes this code and forces the simulation of τ_n . Otherwise, we know that \mathfrak{Y} self-simulates infinitely many times, which means that there exists, for any \mathfrak{Y} -tiling and for any s , an integer $m > s$ such that \mathfrak{Y} self-simulates with squares of size m . Any of these self-simulations represents a \mathfrak{Y} -tiling which simulates other tile sets depending on the infinite input on which it is computing. Therefore, it is enough for Player II to look for the smallest self-simulation where it is possible to write the code of τ_{n+1} . Such a self-simulation always exists. By transitivity of the simulation, this guarantees that the final tiling will simulate τ_{n+1} .

By induction, Player II builds a tiling which simulates at least one tiling for any tile set in Ω . Therefore this tile set is in A , and A is residual in $\mathcal{T}_{\mathfrak{Y}}$ with the Cantor topology.

We now show that B is meager in $\mathcal{T}_{\mathfrak{Y}}$, in the Besicovitch topology. The first move of Player I consists in playing a \mathfrak{Y} -tiling P_1 and an integer n_1 to define the open ball $\mathcal{B}_B(P_1, 1/n_1)$. P_1 simulates at least one tile set of one tile. Without loss of generality, we can suppose that this tile set is the first of our enumeration of tile sets that tile the plane (it can be reordered if necessary). Player II wants

to be sure that after he has played, the code of τ_1 cannot be removed. This code appears regularly in the tilings which means that there exists an m such that all bits of τ_1 appear in all squares of size m in P_1 . If Player II chooses an integer m_1 bigger than m^2 then he is sure that no bits of τ_1 can be changed by Player I since any tiling that has at least one bit of the code τ_1 changed is at least at distance $1/m^2$ of P_1 (by synchronization, changing one bit corresponds to changing one bit in any square of size m).

We now suppose that Player II has already chosen a tiling P_i that simulates all the tile sets in $\{\tau_1, \tau_2, \dots, \tau_{i-1}\}$ and has chosen an integer big enough to force Player I to play a tiling which simulates also $\{\tau_1, \tau_2, \dots, \tau_{i-1}\}$. Player I chooses a tiling P_i and an integer n_i . We show that Player I can choose a tiling $Q_i \in \mathcal{B}_B(P_i, 1/n_i)$ that simulates all of $\{\tau_1, \tau_2, \dots, \tau_{i-1}\} \cup \{\tau_i\}$.

We first make some remarks. If a \mathfrak{Y} -tiling P simulates a tile set τ , it means that there exists a level of squares j in P where the simulation of the tiles of τ is made. Of course, not all the tiles of P are concerned by this simulation. We can bound the proportion of tiles that are concerned by this simulation. Indeed, only the tiles which are in squares of level j , and the tiles which are in communication zones between these squares are influenced by this simulation. Therefore the bound is close to $3/4$. The exact proportion is not important, since we just need the proportion of tiles concerned by a simulation to be strictly less than 1.

Let S be a \mathfrak{Y} -tiling. \mathfrak{Y} self-simulates, therefore S simulates a \mathfrak{Y} -tiling, say S_1 . By the previous remark, at most $3/4$ of the tiles of S are used to simulate S_1 . Since S_1 is also a \mathfrak{Y} -tiling, then S_1 simulates a \mathfrak{Y} -tiling, say S_2 . $3/4$ of the tiles of S_1 are used to simulate S_2 , and by transitivity, $(3/4)^2$ of the tiles of S are used to simulate S_2 . By induction, we obtain a sequence $\{S, S_1, S_2, \dots\}$ of \mathfrak{Y} -tilings such that $(3/4)^n$ of the tiles of S are used to simulate S_n .

Because of this remark, Player I can modify P_i such that it simulates a new \mathfrak{Y} -tiling S_t by changing a proportion of tiles in P_i smaller than $1/n_i$. This tiling S_t has the particularity of having the code of the i^{th} tile set of Ω in its input and thus, simulates τ_i . Player II plays this tilings Q_i which is at distance less than $1/n_i$ of P_i and which simulates τ_i . Any index of the different tile sets of $\{\tau_1, \tau_2, \dots, \tau_{n+1}\}$ appears, or is simulated, regularly in the tiling P_{2n+2} : there exists an integer m such that any bit of these indexes appears in all squares of size m . As before, if Player II chooses an integer greater than m^2 , he guarantees that none of these tiles can be changed, and therefore, the only possibility for Player I is to choose a \mathfrak{Y} -tiling that simulates any tile set of the set $\{\tau_1, \tau_2, \dots, \tau_{n+1}\}$.

By induction, the tiling obtained at the end of the game is a tiling that simulates all tile sets of Ω . Therefore this tile set is in A , and A is residual in $\mathcal{T}_{\mathfrak{Y}}$ within the Besicovitch topology. □

Acknowledgements

We warmly thank Guillaume Theyssier who, in the first place, convinced one of the co-authors of the possible existence of the previously described tile set. We are also indebted to Bruno Durand for his pertinent remarks on previous work that undoubtedly encouraged us in pursuing in this direction.

References

- [AD96] Allauzen, C., Durand, B.: Appendix A: Tiling problems. The classical decision problem, 407–420 (1996)
- [BDJ08] Ballier, A., Durand, B., Jeandel, E.: Structural aspects of tilings. In: *Proceeding of the Symposium on Theoretical Aspects of Computer Science*, pp. 61–72 (2008)
- [Ber66] Berger, R.: The undecidability of the domino problem. *Mem. Amer. Math Soc.* 66, 1–72 (1966)
- [DLS01] Durand, B., Levin, L.A., Shen, A.: Complex tilings. In: *STOC*, pp. 732–739 (2001)
- [DLS04] Durand, B., Levin, L.A., Shen, A.: Local rules and global order. *Mathematical Intelligencer* 27(1), 64–68 (2004)
- [DRS08] Durand, B., Romashchenko, A.E., Shen, A.: Fixed point and aperiodic tilings. In: *Developments in Language Theory*, pp. 276–288 (2008)
- [Han74] Hanf, W.P.: Nonrecursive tilings of the plane. I. *J. Symb. Log* 39(2), 283–285 (1974)
- [LW07] Lafitte, G., Weiss, M.: Universal tilings. In: Thomas, W., Weil, P. (eds.) *STACS 2007. LNCS*, vol. 4393, pp. 367–380. Springer, Heidelberg (2007)
- [LW08a] Lafitte, G., Weiss, M.: Simulations between tilings. In: Beckmann, A., Dimitracopoulos, C., Löwe, B. (eds.) *Logic and Theory of Algorithms*, 4th Conference on Computability in Europe, CiE 2008, Athens, Greece, June 2008, University of Athens (2008)
- [LW08b] Lafitte, G., Weiss, M.: Computability of tilings. In: *International Federation for Information Processing, Fifth IFIP International Conference on Theoretical Computer Science*, vol. 273, pp. 187–201 (2008)
- [LW08c] Lafitte, G., Weiss, M.: A topological study of tilings. In: Agrawal, M., Du, D.-Z., Duan, Z., Li, A. (eds.) *TAMC 2008. LNCS*, vol. 4978, pp. 375–387. Springer, Heidelberg (2008)
- [Mye74] Myers, D.: Nonrecursive tilings of the plane. II. *J. Symb. Log* 39(2), 286–294 (1974)
- [Oxt57] Oxtoby, J.C.: Tilings: recursivity and regularity. *Contribution to the theory of games III*(39), 159–163 (1957)
- [Rob71] Robinson, R.M.: Undecidability and nonperiodicity for tilings of the plane. *Inv. Math.* 12, 117–209 (1971)
- [Wan61] Wang, H.: Proving theorems by pattern recognition II. *Bell Systems Journal* 40, 1–41 (1961)

Linear Kernel for Planar Connected Dominating Set

Daniel Lokshtanov¹, Matthias Mnich², and Saket Saurabh¹

¹ Universitetet i Bergen, Institutt for Informatikk,
Postboks 7803, 5020 Bergen, Norway
{daniello,saket.saurabh}@ii.uib.no

² Technische Universiteit Eindhoven, Faculteit Wiskunde en Informatica,
Postbus 513, 5600 MB Eindhoven, The Netherlands
m.mnich@tue.nl

Abstract. We provide polynomial time data reduction rules for CONNECTED DOMINATING SET in planar graphs and analyze these to obtain a linear kernel for the planar CONNECTED DOMINATING SET problem. To obtain the desired kernel we introduce a method that we call *reduce or refine*. Our kernelization algorithm analyzes the input graph and either finds an appropriate reduction rule that can be applied, or zooms in on a region of the graph which is more amenable to reduction. We find this method of independent interest and believe that it will be useful to obtain linear kernels for other problems on planar graphs.

1 Introduction

Preprocessing of data is one of the oldest and widely used methods in practical algorithms. Parameterized Complexity provides a natural way to measure the quality of preprocessing. In parameterized complexity a problem Π consists of a pair (I, k) where I is the input and k is a parameter (which typically is the solution size). A problem Π is said to have a *kernelization algorithm* if there exists a preprocessing algorithm, which given a parameterized instance (I, k) of Π , runs in time polynomial in $|I|$ and k and outputs a simpler instance (I', k') of Π , such that (I, k) is a yes-instance if and only if (I', k') is a yes-instance and the size of (I', k') is bounded by a function of k alone. The reduced instance I' is called the kernel for the problem. The problem Π is said to have a polynomial (linear) kernel if the reduced instance is bounded by a polynomial (linear) function of k .

Kernelization has been extensively studied, resulting in polynomial kernels for a variety of problems. Notable examples include a $2k$ kernel for VERTEX COVER [6], a $355k$ kernel for DOMINATING SET in planar graphs [1] which later was improved to a $67k$ kernel [5], and a $O(k^2)$ kernel for FEEDBACK VERTEX SET [15] parameterized by the solution size. A significant amount of research has gone into providing linear kernels for NP-hard problems on planar graphs. A foundation for linear kernelization in planar graphs was built by Alber et al. [1] who gave a $335k$ -sized kernel for planar DOMINATING SET. The main

ingredient in the analysis of the reduced instance was the notion of region decomposition for the input planar graph where the number of regions depended linearly on the size of the parameter. These ideas were later abstracted by Guo and Niedermeier [12] who gave a framework to obtain linear kernels for planar graph problems possessing a certain “locality property”. This framework has been successfully applied to yield linear kernels for the CONNECTED VERTEX COVER, MINIMUM EDGE DOMINATING SET, MAXIMUM TRIANGLE PACKING, EFFICIENT EDGE DOMINATING SET, INDUCED MATCHING and FULL-DEGREE SPANNING TREE problems [12][13][14]. However, the framework proposed by Guo and Niedermeier [12] in its current form is not able to handle problems like FEEDBACK VERTEX SET and ODD CYCLE TRANSVERSAL because these do not admit the “locality property” required by the framework. Recently, Bodlaender and Penninx [3] and Bodlaender et al. [4] have obtained linear kernels for FEEDBACK VERTEX SET and CYCLE PACKING on planar graphs respectively.

The list of problems for which linear kernels are known for planar graphs excludes problems which demand the solution to be connected. The mere exception is the the CONNECTED VERTEX COVER problem for which the reduction rules for planar VERTEX COVER apply [12]. In this article we try to fill this void by studying the CONNECTED DOMINATING SET problem for planar graphs from the viewpoint of linear kernelization. The problem is defined as follows.

CONNECTED DOMINATING SET: Given a graph $G = (V, E)$ and a positive integer k , the objective is to find a subset $D \subseteq V$ of size at most k such that $G[D]$ is connected and for every vertex $v \in V$ either $v \in D$ or one of its neighbors is in D .

CONNECTED DOMINATING SET is a well-studied NP-hard problem that finds applications in various network design problems. It remains NP-hard when restricted to the class of planar graphs [9], and has a $O(\log n)$ -approximation algorithm [10]. The parameterized version of the problem is known to be W[2]-complete for general graphs and admits a sub-exponential time parameterized algorithm for planar graphs [7]. In general graphs the problem has also been studied in the realm of moderately exponential time algorithms leading to an algorithm with running time $O(1.9407^n)$ [8]. Here we provide polynomial time data reduction rules for CONNECTED DOMINATING SET in planar graphs which lead to a linear kernel for the problem. In particular, we prove the following theorem.

Theorem 1. *The CONNECTED DOMINATING SET problem parameterized by solution size k has a linear kernel on planar graphs.*

This answers a question asked by Guo [11] during the visit of the third author to Jena in 2007. Our results are based on the *reduce-or-refine* technique, that we introduce here. Until now the notion of *region decomposition* was used only in the analysis of the kernel size, and not explicitly applied in the reduction rules. We utilize the fact that a region decomposition can be obtained in polynomial time given a solution set S . In particular, we compute S using the known polynomial time approximation scheme for CONNECTED DOMINATION SET and compute

the decomposition from S using algorithms described by Alber et al. [1] and Guo and Niedermeier [12]. The main technical part of our proofs is devoted to showing that if a region contains more vertices than a fixed constant, we can in polynomial time find a vertex in this region whose removal will not affect the size of an optimal solution. The idea is to check whether the region contains more than a fixed constant number of copies of a particular structure. If so then we can *reduce* the graph by removing a vertex in such a structure. If there are few or no copies of the structure in this region then we can *zoom in on*, or *refine*, to a smaller region that still contains many vertices but completely excludes the structure. The process is then repeated for a different “bad” structure until the region we have zoomed in on looks so simple that it is easy to identify a vertex to remove.

Since the number of regions in our computed region decomposition is $O(k)$, if the graph has too many vertices then we can identify a region in which a useless vertex can be found. Thus we obtain the desired linear upper bound on the size of the kernel.

2 Preliminaries

In this section we collect necessary definitions and results required to obtain a linear kernel for the problem. We also give a few basic reduction rules for CONNECTED DOMINATING SET.

Let $G = (V, E)$ be a connected planar graph, without loops or multiple edges. For each vertex $v \in V$, define the *open neighborhood of v* as $N(v) = \{u \in V \mid \{u, v\} \in E\}$ and the *closed neighborhood of v* as $N[v] = N(v) \cup \{v\}$. A vertex v is *universal for G* if $N[v] = V$. A path in G between distinct vertices v, w is called $[v, w]$ -*path*. Let $G[U]$ denote the induced graph on U , for any vertex set $U \subseteq V$. For a graph $G = (V, E)$ a subset $D \subseteq V$ is called a *dominating set* if for every vertex $v \in V$ either $v \in D$ or $N(v) \cap D \neq \emptyset$. For a graph G , the size of a minimum dominating set is denoted by $\gamma(G)$ and the size of a minimum connected dominating set is denoted by $\gamma_c(G)$. A graph that can be drawn in the plane without edge crossing is called *planar graph*. A *plane graph* is a planar graph with a fixed embedding in the plane. Throughout the paper, we assume that we are working with an arbitrary but fixed embedding of G in the plane.

With respect to connected dominating sets, the following reductions rules will frequently help to simplify the input graph. If G has a universal vertex v then $\{v\}$ is a minimum connected dominating set for G . Henceforth we assume that G has no universal vertex.

Lemma 1. *Let G be a graph and let v be a vertex of G contained in some minimum connected dominating set S of G . Let G_v be the graph obtained from G by removing the edges of $G[N(v)]$. Then $\gamma_c(G) = \gamma_c(G_v)$.*

Whenever possible, we remove “twin vertices” from the graph.

Lemma 2. *Let $G = (V, E)$ be a graph and let u, u' be distinct vertices such that $N[u] = N[u']$. Then $\gamma_c(G) = \gamma_c(G - u')$.*

This reduction rule is only used to reduce the graph in practice; we do not use it in the analysis of the kernel size. Now we evoke the notions of a region and a region decomposition that were first introduced by Alber et al. [1].

Definition 1. Let G be a plane graph and let v, w be distinct vertices of G . A region $R(v, w)$ between v and w is a closed subset of the plane such that

- the boundary of $R(v, w)$ is formed by two simple $[v, w]$ -paths each of length at most three, and
- all vertices strictly inside region $R(v, w)$ belong to $N(v) \cup N(w)$, and are called inner vertices of $R(v, w)$.

If $R(v, w), R'(v, w)$ are regions between v and w then $R(v, w) \cup R'(v, w)$ denotes the region that is defined by the union of the closed subsets of the plane defined by $R(v, w)$ and $R'(v, w)$. We use $V(R(v, w))$ to denote the set of inner vertices of the region $R(v, w)$.

Definition 2. Let $G = (V, E)$ be a plane graph and let $S \subseteq V$. An S -region decomposition of G is a set \mathcal{R} of regions $R(v, w)$ between distinct vertices $v, w \in S$ such that

- each region $R(v, w)$ contains no vertex from $S \setminus \{v, w\}$, and
- any two distinct regions can only intersect in their boundaries.

For an S -region decomposition \mathcal{R} , let $V(\mathcal{R}) = \cup_{R(v,w) \in \mathcal{R}} V(R(v, w))$. An S -region decomposition \mathcal{R} of G is maximal if there is no region $R(v, w)$ such that $\mathcal{R} \cup \{R(v, w)\}$ is an S -region decomposition of G satisfying $V(\mathcal{R}) \subsetneq V(\mathcal{R} \cup \{R(v, w)\})$.

We now state two known results about maximal region decompositions. The results say that given a plane graph G and a dominating set S , one can obtain an S -region decomposition of G with $O(\gamma_c(G))$ regions that together cover all but $O(\gamma_c(G))$ vertices of G .

Proposition 1 (Guo and Niedermeier [12]). Let G be a plane graph and let S be a dominating set of G . There exists a maximal S -region decomposition of G containing at most $3\gamma(G)$ regions.

Proposition 1 has a constructive proof by a polynomial-time algorithm.

Proposition 2 (Alber et al. [1]). Let G be a plane graph and let S be a dominating set of G . If \mathcal{R} is a maximal S -region decomposition of G then at most $170\gamma(G)$ vertices of G do not belong to \mathcal{R} .

Since any connected dominating set of a graph is also a dominating set, Propositions 1 and 2 together imply that a planar graph G has a maximal S -region decomposition for a connected dominating set S with $O(\gamma_c(G))$ regions covering all but $O(\gamma_c(G))$ vertices of G .

3 A Reduce-or-Refine Scheme

In this section, we provide a polynomial time algorithm to bound the number of vertices per region by some constant C . As long as there exists a region with more than C vertices, this region will either be “refined” into multiple regions or some vertices will be removed from it. We show that in polynomial time the algorithm produces an instance where the number of vertices in each region is bounded by a constant and the total number of regions is $O(k)$.

Lemma 3. *Let $G = (V, E)$ be a plane graph and let $R(v, w)$ be a region between $v, w \in V$. There exists a minimum connected dominating set S of G such that S contains at most two inner vertices from $R(v, w)$.*

Let $N_R(v, w)$ denote the common neighborhood of v and w in the region $R(v, w)$, that is,

$$N_R(v, w) = \{u \in R(v, w) \mid u \in N(v) \cap N(w)\}.$$

Case 1: $N_R(v, w)$ contains at least 106 vertices.

Let x_1, \dots, x_ℓ be a labeling of the vertices in $N_R(v, w)$ such that for all $i = 1, \dots, \ell - 1$, there is a region $r_i(v, w)$ between v and w with clockwise ordering (v, x_i, w, x_{i+1}) of boundary vertices. We define a coloring c on the set $\{r_1(v, w), \dots, r_{\ell-1}(v, w)\}$. We color the region $r_i(v, w)$ black, white, black-and-white, or transparent according to the scheme outlined below. Refer to Figure 1 for an example.

- black, if $r_i(v, w)$ contains some inner vertices adjacent to v and no inner vertices adjacent to w ,
- white, if $r_i(v, w)$ contains some inner vertices adjacent to w and no inner vertices adjacent to v ,
- black-and-white, if $r_i(v, w)$ contains some inner vertices adjacent to v and some inner vertices adjacent to w ,
- transparent, if $r_i(v, w)$ contains no inner vertices.

The *black (white) weight* of c is the number of regions that are colored black or black-and-white (white or black-and-white).

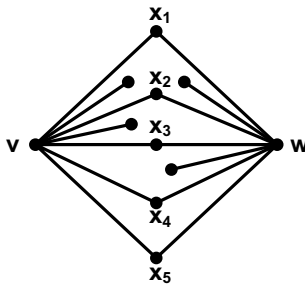


Fig. 1. Regions $r_i = r_i(v, w)$ for $i = 1, 2, 3, 4$. The coloring c colors r_1 black-and-white, r_2 black, r_3 white and r_4 transparent.

Observation 1. *Let $G = (V, E)$ be a plane graph and let $R(v, w)$ be a region between $v, w \in V$. If the coloring c has black weight at least 7 then any minimum connected dominating set of G containing at most two inner vertices of $R(v, w)$ contains v . Similarly, if the coloring c has white weight at least 7 then any minimum connected dominating set of G containing at most two inner vertices of $R(v, w)$ contains w .*

Case 1.1: The coloring c has black weight at least 8 and white weight at least 8.

Let S be a minimum connected dominating set. Note that S must contain v and w , by Observation [□](#). Now apply Lemma [□](#) to turn the induced subgraphs $G[N(v)]$ and $G[N(w)]$ into independent sets.

Lemma 4. *(reduce) Let $G = (V, E)$ be a plane graph and let $R(v, w)$ be a region between $v, w \in V$. Suppose that coloring c has black weight at least 8 and white weight at least 8. Let y be an inner vertex of a black or black-and-white region such that y is a neighbor of v but not a neighbor of w . Then $\gamma_c(G) = \gamma_c(G - y)$.*

An analogous reduction rule applies for inner vertices y inside some white or black-and-white region.

Case 1.2: The coloring c has black weight at least 8 and white weight at most 7.

In this case, there exists a region $r(v, w)$ that is colored black.

Lemma 5. *(reduce) Let G be a plane graph and let $R(v, w)$ be a region between $v, w \in V$. Suppose that coloring c has black weight at least 8 and white weight at most 7. Let y be an inner vertex of a black region such that y is a neighbor of v but not a neighbor of w . Then $\gamma_c(G) = \gamma_c(G - y)$.*

The case of coloring c having large white weight and small black weight is similar.

Case 1.3: The coloring c has black weight at most 7 and white weight at most 7.

Lemma 6. *(refine) Let $G = (V, E)$ be a plane graph and let $R(v, w)$ be a region between $v, w \in V$ such that $|N_R(v, w)| \geq 106$. Suppose that coloring c has black weight at most 7 and white weight at most 7. Then there exists a region $R'(v, w)$ such that $|N_{R'}(v, w)| \geq 8$ and containing only transparent regions from $\{r_1(v, w), \dots, r_{\ell-1}(v, w)\}$.*

Case 1.4: The coloring c colors all regions transparent.

Observation 2. *Let $G = (V, E)$ be a plane graph and let $R(v, w)$ be a region between $v, w \in V$ such that $|N_R(v, w)| \geq 7$. If the coloring c has black weight equal to zero and white weight equal to zero then any minimum connected dominating set of G containing at most two inner vertices of $R(v, w)$ contains at least one of v and w .*

Lemma 7. (reduce) Let $G = (V, E)$ be a plane graph and let $R(v, w)$ be a region between $v, w \in V$ such that $|N_R(v, w)| \geq 8$. Suppose that the coloring c has black weight equal to zero and white weight equal to zero. Let y be an inner vertex of $R(v, w)$. Then $\gamma_c(G) = \gamma_c(G - y)$.

We summarize Case 1.

Lemma 8. (reduce) There is an algorithm that, given a plane graph $G = (V, E)$ and a region $R(v, w)$ between vertices $v, w \in V$ such that $|N(v, w)| \geq 106$, in polynomial time computes a subgraph G' of G with fewer vertices than G such that $\gamma_c(G') = \gamma_c(G)$.

Proof. The algorithm proceeds as follows. First, it constructs the coloring c of the regions $r_1(v, w), \dots, r_{\ell-1}(v, w)$.

If c has black weight at least 8 and white weight at least 8 then let y be an inner vertex of a black or black-and-white region that is a neighbor of v but not a neighbor of w . Now by Lemma 4, $G' = G - y$ is a subgraph of G with the desired properties.

If c has black weight at least 8 and white weight at most 7 then let y be an inner vertex of a black region and let $G' = G - y$. By Lemma 5 it holds $\gamma_c(G') = \gamma_c(G)$. Proceed similarly if c has black weight at most 7 and white weight at least 8, in which case we let $G' = G - y'$ for some inner vertex y' of a white region.

If c has black weight at most 7 and white weight at most 7 then by Lemma 6 there exists a region $R'(v, w)$ entirely contained in $R(v, w)$ such that any inner vertex of $R'(v, w)$ is a common neighbor of v and w . Thus by Lemma 7, letting y be an inner vertex of $R'(v, w)$ makes $G' = G - y$ a subgraph of G with the desired properties. □

Case 2: $N_R(v, w)$ contains at most 105 vertices.

Lemma 9. (refine) Let $G = (V, E)$ be a plane graph and let $R(v, w)$ be a region between $v, w \in V$ such that $|N_R(v, w)| \leq 105$. Let m be the number of vertices in $R(v, w)$ other than v and w . Then there is a region $R'(v, w)$ entirely contained in $R(v, w)$ that contains at least $m/104 + 2$ vertices and such that $N_{R'}(v, w) = \emptyset$.

Case 3: $N_R(v, w)$ contains no inner vertices of $R(v, w)$.

Let $\mathcal{P} = \{P_1, \dots, P_m\}$ be a maximum-size set of internally vertex-disjoint induced $[v, w]$ -paths of length three entirely contained in $R(v, w)$, such that for all $i = 1, \dots, m-1$ the vertices of $P_i \cup P_{i+1}$ form the boundary of a region $s_i(v, w)$ not containing vertices from P_j for any $j \notin \{i, i + 1\}$. For each $i = 1, \dots, m$, let v_i be the internal vertex of P_i that is adjacent to v , and let w_i be the internal vertex of P_i that is adjacent to w .

Case 3.1: There are at least 12 internally vertex-disjoint $[v, w]$ -paths of length three.

Observation 3. Let $G = (V, E)$ be a plane graph and let $R(v, w)$ be a region between $v, w \in V$. If $|\mathcal{P}| \geq 12$ then every minimum connected dominating set of G containing at most two inner vertices of $R(v, w)$ contains both v and w .

Lemma 10. (reduce) Let $G = (V, E)$ be a plane graph and let $R(v, w)$ be a region between $v, w \in V$. If $|\mathcal{P}| \geq 12$ then $\gamma_c(G) = \gamma_c(G - v_3)$.

Case 3.2: There are at most 11 internally vertex-disjoint $[v, w]$ -paths of length three.

Lemma 11. (refine) Let $G = (V, E)$ be a plane graph and let $R(v, w)$ be a region between $v, w \in V$ such that $N_R(v, w)$ contains no inner vertices of $R(v, w)$. Let m be the number of vertices in $R(v, w)$ other than v and w . If $|\mathcal{P}| \leq 11$ then there is a region $R'(v, w)$ entirely contained in $R(v, w)$ that contains at least $m/10 + 2$ vertices such that every $[v, w]$ -path in $R(v, w)$ contains at least one border vertex of $R(v, w)$ except v and w .

Case 3.3: There are no internally vertex-disjoint $[v, w]$ -paths of length three.

Lemma 12. (reduce) Let $G = (V, E)$ be a plane graph and $R(v, w)$ be a region between $v, w \in V$ with at least 1274 vertices such that every $[v, w]$ -path in $R(v, w)$ contains at least one border vertex of $R(v, w)$ other than v and w . There is a polynomial time algorithm that given G and $R(v, w)$ computes a subgraph G' of G with $|V(G')| < |V(G)|$ and $\gamma_c(G') = \gamma_c(G)$.

We summarize the case analysis.

Lemma 13. There is a polynomial time algorithm that, given a plane graph G and a region $R(v, w)$ between vertices $v, w \in V$ with at least 1322672 vertices, computes a subgraph G' of G such that G' has fewer vertices than G and $\gamma_c(G') = \gamma_c(G)$.

Proof. If $|N_R(v, w)| \geq 106$ then Lemma 8 applied to G and $R(v, w)$ yields the desired subgraph G' . Otherwise, by Lemma 9 there exists a region $R'(v, w)$ with at least 12720 vertices entirely contained in $R(v, w)$ and such that no common neighbours of v and w are inner vertices of $R'(v, w)$. If $R'(v, w)$ contains at least 12 internally vertex-disjoint induced $[v, w]$ -paths of length three then by Lemma 10 then $G' = G - v_3$ is the desired subgraph of G . If $R'(v, w)$ contains at most 11 internally vertex-disjoint induced $[v, w]$ -paths of length three then by Lemma 11 there is a region $R''(v, w)$ such that $R''(v, w)$ contains at least 1274 vertices and there is no $[v, w]$ -path such that all its internal vertices are inner vertices of $R''(v, w)$. In this case Lemma 12 implies that a subgraph G' of G with $V(G') < V(G)$ and $\gamma_c(G') = \gamma_c(G)$ can be computed in polynomial time. \square

We are now in position to give a proof of our main result.

Proof of Theorem 1: We prove that the planar CONNECTED DOMINATING SET problem with parameter k admits a kernel of size $3968187k$. We give an algorithm that given an integer k and a plane graph $G = (V, E)$ with at least $3968187k$ vertices, in polynomial time either concludes that $\gamma_c(G) > k$ or computes a subgraph G' of G such that G' has fewer vertices than G and $\gamma_c(G') = \gamma_c(G)$. The

algorithm proceeds as follows. First, let $\epsilon = 1/3968186$ and compute a $(1 + \epsilon)$ -approximation S_1 of a minimum connected dominating set for G using the PTAS of Demaine and Hajiaghayi [7]. If S_1 has strictly more than $(1 + \epsilon)k$ vertices then answer $\gamma_c(G) > k$ and stop.

Otherwise, compute a maximal S_1 -region decomposition \mathcal{R} of G via the algorithm by Guo and Niedermeier [12]. By Proposition 1, there are at most $3(1 + \epsilon)k$ regions in \mathcal{R} , and by Proposition 2 at most $170(1 + \epsilon)k$ vertices do not belong to any region in \mathcal{R} . By the pigeonhole principle, there exists a region $R(v, w) \in \mathcal{R}$ between vertices $v, w \in V$ that contains at least 1322672 vertices. Hence by Lemma 13, a subgraph G' of G with fewer vertices than G and $\gamma_c(G') = \gamma_c(G)$ can be computed in polynomial time. \square

4 Conclusion and Further Work

In this paper we showed that CONNECTED DOMINATING SET in planar graphs admits a linear kernel. This is the first linear kernel for a “connectivity” problem on planar graphs that does not follow directly from the framework of Guo and Niedermeier [12].

Our algorithm is impractical for two reasons. The first is the huge constant in the kernel size, and the second is the choice of $\epsilon = 1/3968186$ in the PTAS for DOMINATING SET that yields an unmanageable running time. We think that both these problems can be remedied; choosing $\epsilon = 1$ yields a 2-approximation for DOMINATING SET in planar graphs that runs quite quickly, at the cost of a factor 2 in the kernel size. Also, the constant in our kernelization can be improved significantly. In this paper we focused only on showing the existence of a linear kernel and in many places we deliberately picked a proof that yielded a higher constant but was more readable and understandable. It would be interesting to see how far the kernel size can be reduced; we believe a constant below 1000 to be achievable. A possible way to attack this problem would be to eliminate the “refine” steps and re-analyzing the cases, taking into account the noise that the “refine” steps removed.

The linear kernel for the DOMINATING SET problem has been extended from planar graphs to graphs excluding a complete bipartite graph $K_{3,h}$ as a minor [2]. A natural question to ask is whether the same can be done for the minimum connected dominating set problem. Finally, it would be interesting to see whether the reduce or refine technique could be applied to achieve this, or to give kernels for other problems.

References

1. Alber, J., Fellows, M.R., Niedermeier, R.: Polynomial-time data reduction for dominating set. *J. ACM* 51(3), 363–384 (2004) (electronic)
2. Alon, N., Gutner, S.: Kernels for the dominating set problem on graphs with an excluded minor. Technical report, ECCO Report TR08-066 (2008)
3. Bodlaender, H.L., Penninkx, E.: A linear kernel for planar feedback vertex set. In: Grohe, M., Niedermeier, R. (eds.) *IWPEC 2008*. LNCS, vol. 5018, pp. 160–171. Springer, Heidelberg (2008)

4. Bodlaender, H.L., Penninkx, E., Tan, R.B.: A linear kernel for the k -disjoint cycle problem on planar graphs. LNCS, vol. 5369, pp. 306–317. Springer, Berlin (2008)
5. Chen, J., Fernau, H., Kanj, I.A., Xia, G.: Parametric duality and kernelization: lower bounds and upper bounds on kernel size. *SIAM J. Comput.* 37(4), 1077–1106 (2007)
6. Chen, J., Kanj, I.A., Jia, W.: Vertex Cover: Further observations and further improvements. *Journal of Algorithms* 41(2), 280–301 (2001)
7. Demaine, E.D., Hajiaghayi, M.: Bidimensionality: new connections between FPT algorithms and PTASs. In: *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 590–601. ACM, New York (2005) (electronic)
8. Fomin, F.V., Grandoni, F., Kratsch, D.: Solving connected dominating set faster than 2^n . *Algorithmica* 52(2), 153–166 (2008)
9. Garey, M.R., Johnson, D.S.: *Computers and intractability. A guide to the theory of NP-completeness*, A Series of Books in the Mathematical Sciences. W. H. Freeman and Co., San Francisco (1979)
10. Guha, S., Khuller, S.: Approximation algorithms for connected dominating sets. *Algorithmica* 20(4), 374–387 (1998)
11. Guo, J.: Private communication (2007)
12. Guo, J., Niedermeier, R.: Linear problem kernels for NP-hard problems on planar graphs. In: Arge, L., Cachin, C., Jurdziński, T., Tarlecki, A. (eds.) *ICALP 2007*. LNCS, vol. 4596, pp. 375–386. Springer, Heidelberg (2007)
13. Guo, J., Niedermeier, R., Wernicke, S.: Fixed-parameter tractability results for full-degree spanning tree and its dual. In: Bodlaender, H.L., Langston, M.A. (eds.) *IWPEC 2006*. LNCS, vol. 4169, pp. 203–214. Springer, Heidelberg (2006)
14. Moser, H., Sikdar, S.: The parameterized complexity of the induced matching problem in planar graphs. In: Preparata, F.P., Fang, Q. (eds.) *FAW 2007*. LNCS, vol. 4613, pp. 325–336. Springer, Heidelberg (2007)
15. Thomassé, S.: Bidimensionality: new connections between FPT algorithms and PTASs. In: *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 115–119. ACM, New York (2009) (electronic)

A Simple Greedy Algorithm for the k -Disjoint Flow Problem*

Maren Martens

Zuse Institute Berlin, Takustr. 7, 14195 Berlin, Germany
martens@zib.de

Abstract. In classical network flow theory the choice of paths, on which flow is sent, is only restricted by arc capacities. This, however, is not realistic in most applications. Many problems restrict, e.g., the number of paths being used to route a commodity. One idea to increase reliability of routings, e.g., in telecommunication, is to copy a demand and send the copies along disjoint paths. Such problems theoretically result in the k -disjoint flow problem (k -DFP). This problem is a variant of the classical multicommodity flow problem with the additional requirement that the number of paths to route a commodity is bounded by a given parameter. Moreover, all paths used by the same commodity have to be arc disjoint.

We present a simple greedy algorithm for the optimization version of the k -DFP where the objective is to maximize the sum of routed demands. This algorithm generalizes a greedy algorithm by Kolman and Scheideler (2002) that approximates the corresponding unsplittable flow problem, in which every commodity may be routed along a single path only. We achieve an approximation factor of $O(k_{\max}\sqrt{m}/k_{\min})$, where m is the number of arcs and k_{\max} (k_{\min}) is the maximum (minimum) bound on the number of paths allowed to route any of the commodities. We argue that this performance guarantee is best possible for instances where k_{\max}/k_{\min} is constant, unless $\mathcal{P} = \mathcal{NP}$.

1 Introduction

Problem Definition and Notation. During the past couple of years path constrained network flows gained increasing interest. Starting from the classical maximum s - t -flow problem introduced by Ford and Fulkerson [7] in 1956, research went on towards multicommodity flows (see, e.g., [2]). In the *multicommodity flow problem (MCFP)* we have a directed graph (or digraph) $D = (V, A)$ with arc capacities $u : A \rightarrow \mathbb{R}^+$ and commodities $\mathcal{T} \subseteq V \times V$. We number the commodities from 1 to K . The i th commodity is denoted as (s_i, t_i) and has a nonnegative demand $d_i \in \mathbb{R}^+$. For every commodity, the whole demand has to be routed through the network obeying the arc capacities. More formally, for every $i = 1, \dots, K$, we have to find a set \mathcal{P}_i of s_i - t_i -paths together with a flow function $f : \mathcal{P}_i \rightarrow \mathbb{R}^+$ such that $\sum_{P \in \mathcal{P}_i} f(P) = d_i$ and $\sum_i \sum_{P \in \mathcal{P}_i: a \in P} f(P) \leq u(a)$, for every $a \in A$. We call a path P with positive flow value $f(P)$ a *flow path*. As

* Research partially supported by the Federal Ministry of Education and Research (BMBF), Germany, as part of the project “EUREKA 100GET Technologies”.

in general it is not possible to route all demands simultaneously while obeying all arc capacities, several optimization versions of the MCFP have been introduced. Among the most popular ones is the maximization of the sum of routed demands. In other words, we have to choose a subset of commodities $\mathcal{T}' \subseteq \mathcal{T}$ of maximum total demand such that there exist flow paths along which we can satisfy all demands in \mathcal{T}' while obeying the arc capacities. This optimization version is also considered in this paper.

However, for many applications it is not realistic to have no restrictions on the choice of flow paths other than the ones induced by the arc capacities. In container routing problems, e.g., the number and sizes of containers bound the number of paths that may be used for a routing and the amount of flow that may be sent along a single path. In telecommunications, demands are usually routed along a single path rather than being split up arbitrarily. To gain reliability for connections, it is common to copy a demand several times and then send all copies along disjoint paths. This results in the *k-disjoint flow problem (k-DFP)*: Given a digraph $D = (V, A)$ with arc capacities $u : A \rightarrow \mathbb{R}^+$, commodities $\mathcal{T} \subseteq V \times V$, and demands d_i , we now also have a parameter $k_i \in \mathbb{N}$, for every commodity i . The task is to find a feasible multicommodity flow as above where, for every i , the number of s_i - t_i -flow paths is exactly k_i , i.e., $|\mathcal{P}_i| = k_i$, all paths in \mathcal{P}_i are arc disjoint, and every path $P \in \mathcal{P}_i$ carries the same amount of flow, i.e., $f(P) = d_i/k_i$. Here, we consider the related optimization problem to maximize the sum of routed demands as it is introduced above. A feasible solution to this optimization problem is called a *k-disjoint flow*.

We use m to denote the number of arcs in a digraph and n to denote the number of its nodes. Further, we use $k_{\min} := \min_{i=1, \dots, K} k_i$ and $k_{\max} := \max_{i=1, \dots, K} k_i$ respectively as the minimum and maximum upper bound on the permitted number of flow paths.

Related Results from the Literature. To the best of our knowledge, the first and only who considered the *k-DFP* as introduced above were Bagchi, Chaudhary, Scheideler, and Kolman [4]. In contrast to the present paper, however, they do not admit different bounds on the number of flow paths for different commodities. In their considerations Bagchi et al. use the “flow number” F , which provides information on the level of interconnectedness of a network. In [13] it is proven that $F = O(\Delta \alpha^{-1} \log n)$ with Δ being the maximum node degree and α being the expansion of the network, i.e., the value of the minimum quotient cut. Bagchi et al. show that the greedy algorithm that they develop for the *k-DFP* yields an approximation factor of $O(k^3 F \log(kF))$ for the optimization version to maximize the sum of satisfied demands. However, this holds only for instances with unit arc capacities and maximum demand at most k . Also in [4], Bagchi et al. consider problems that are closely related to the *k-DFP*. Those are the integral splittable flow problem (ISF) and the *k-splittable flow problem (k-SFP)*. The latter is basically the same as the *k-DFP*. However, in the *k-SFP* the (at most) k flow paths for the same commodity do not have to be disjoint. Nor it is required that exactly k paths are used to route a demand or that all flow paths of one commodity carry the same amount. For the maximization version of the

k -SFP, Bagchi et al. develop a greedy algorithm with performance guarantee $O(k^2 F)$, again under the assumption that they have unit arc capacities and the maximum demand is at most k . In the ISF integral demands may be satisfied by an arbitrary number of flow paths carrying an integral amount of flow. Also for this problem Bagchi et al. introduce a greedy algorithm and obtain an approximation factor of $O(F)$, for instances with uniform arc capacities in which the maximum demand is at most the capacity of the arcs. The ISF has already earlier been studied by Guruswami et al. [8] who obtain an approximation factor of $O(\sqrt{m\Delta} \log^2 m)$. They also prove that there is no approximation factor better than $O(\sqrt{m})$, unless $\mathcal{P} = \mathcal{NP}$.

Baier, Köhler, and Skutella [6] generalize the k -SFP by allowing different bounds on the number of flow paths for different commodities. However, they do not examine the optimization version that is considered in this paper. For the objective to minimize the maximum overload of an arc while satisfying all demands, they show how to obtain the same asymptotic approximation factors for the k -SFP as for the unsplittable flow problem (UFP). The UFP is the special case of the k -SFP where only a single path may be used to route a demand. This problem was introduced by Kleinberg [11] in 1996. The best known and possible approximation factor for the maximization problem of the UFP is $O(\sqrt{m})$, see, e.g., [13] for a simple greedy algorithm that does not require any restrictions on the size of the arc capacities. Variants of the k -SFP with restrictions on the flow paths, e.g., length or flow bounds, are considered in [16,17]. Surveys on unsplittable and k -splittable flows can be found in [5,12,15], the second of which also gives a good survey on the arc disjoint paths problem.

Also closely related to the k -DFP are k -flows, first considered by Kishimoto and Takeuchi [9,10]. A k -flow is defined as a nonnegative linear combination of elementary k -flows where an elementary k -flow is a tuple consisting of k arc disjoint paths, each path carrying exactly one unit of flow. Algorithms for the maximum k -flow problem are developed in [13]. Note that an elementary k -flow is the same as a special k -disjoint flow with $k_i = d_i$.

Contribution of this Paper. To the best of our knowledge, we present the first approximation algorithm for the maximization version of the k -DFP that is independent from the flow number and does not require restrictions on the arc capacities or demands. This algorithm is a simple combinatorial greedy algorithm that generalizes the bounded greedy algorithm for the UFP that was introduced by Kolman and Scheideler [13]. We show that our algorithm always approximates the optimum within a factor $O(k_{\max}\sqrt{m}/k_{\min})$ and argue that this factor is best possible for instances where k_{\max}/k_{\min} is constant, unless $\mathcal{P} = \mathcal{NP}$.

Our results can easily be generalized to the case where we require node disjointness for all paths routing the same commodity. We explain this generalization after introducing the basic algorithm for k -DFP.

2 A Simple Greedy Algorithm

In Section 2.1 we introduce a simple greedy algorithm for the optimization version of the k -DFP to maximize the sum of routed demands. This greedy

algorithm is based on, but generalizes an algorithm by Kolman and Scheider [13] for the UFP. The biggest new challenge for the k -DFP is to find k_i disjoint flow paths (instead of a single path), for every commodity i , that obey the arc capacities and also use only few arcs up to their total capacity. In the analysis following the algorithm in Section 2.2 we also use ideas from [13]. However, we carefully have to deal with the new generalization of the algorithm.

2.1 The Algorithm

Our algorithm basically reduces k -DFP to the *minimum cost flow problem (MCFP)*. In the MCFP, we have a digraph $D = (V, A)$ with arc capacities $u : A \rightarrow \mathbb{R}^+$, arc costs $c : A \rightarrow \mathbb{R}^+$, and supplies/demands $d_v \in \mathbb{R}$, for $v \in V$, such that $\sum_v d_v = 0$. (We call d_v a *supply* if it is positive and a *demand* if it is negative.) The task is to find a feasible *minimum cost flow (MCF)* that satisfies all supplies/demands, i.e., a flow $f : A \rightarrow \mathbb{R}^+$ such that $f(a) \leq u(a)$, for all $a \in A$, $\sum_{a=(v,w) \in A} f(a) - \sum_{a=(w,v) \in A} f(a) = d_v$, for all $v \in V$, and $\sum_a c(a)f(a)$ is minimal. We call a MCF *integral* if $f(a) \in \mathbb{N}$, for all $a \in A$. An (integral) MCF can be computed in polynomial time; for runtimes and details on various algorithms, see, e.g., [18]. Moreover, it can be decomposed into flows on paths between supply and demand nodes and flows on cycles, see also [18]. Since the cycles do not contribute to the net inflow at nodes, we can delete them and obtain a decomposition into flows on paths.

Algorithm 1 specifies an $O(k_{\max}\sqrt{m}/k_{\min})$ -approximation algorithm for k -DFP. It computes two different solutions \mathcal{P} and \mathcal{P}' and, at the end, chooses the better of those two. We decide for exactly one commodity i at a time if we route its demand in one or both of the solutions or if we reject it and do not route any flow for it at all. For every decision, we use special capacity and cost functions that depend on the arc capacities u of the considered k -DFP and on the k -disjoint flow already routed. Assume that we already decided to route flow along paths in a set \mathcal{P} . Then, for every $a \in A$, we set the capacity $u_{\mathcal{P}}(a) = 1$, if $\sum_{P \in \mathcal{P}: a \in P} f(P) + d_i/k_i \leq u(a)$, and $u_{\mathcal{P}}(a) = 0$ otherwise. This means that $u_{\mathcal{P}}$ is 1 on exactly those arcs where d_i/k_i can be routed without violating the arc capacity. We set the cost $c_{\mathcal{P}}(a) = 1$, if $\sum_{P \in \mathcal{P}: a \in P} f(P) + d_i/k_i > u(a)/2$, and $c_{\mathcal{P}}(a) = 0$ otherwise.

The arc capacities in Algorithm 1 ensure that, for every commodity i , the computed integral flows use exactly k_i arc disjoint paths. For every commodity i that is routed in the path set \mathcal{P} computed by Algorithm 1, it is furthermore true that f uses at most \sqrt{m} arcs in their upper halves, i.e., the total flow after routing i is larger than half the arc capacity on at most \sqrt{m} arcs.

Note that f' is constructed in the same way as f , except that f' is unrestricted in the set of arcs that are used in their upper halves. The following example shows that it is important to also compute f' and return the bigger of both flows: Assume we have a single commodity in a network consisting of two nodes s and t and two parallel arcs from s to t with capacities 1 each. We consider the demand $d = 2$ and $k = 2$. Then in \mathcal{P}' we route 1 unit of flow along either of the

Algorithm 1. $O(k_{\max}\sqrt{m}/k_{\min})$ -approximation for the k -DFP

Input: An instance of k -DFP.

Output: A k -disjoint flow that routes at least $1/O(k_{\max}\sqrt{m}/k_{\min})$ of the maximum sum of simultaneously routable demands.

Sort the commodities such that $d_1/k_1 \geq d_2/k_2 \geq \dots \geq d_K/k_K$.

for $i = 1$ **to** K **do**

Compute an integral MCF f that routes k_i units from s_i to t_i in D with arc capacities $u_{\mathcal{P}}$ and cost function $c_{\mathcal{P}}$ as above, for $\mathcal{P} := \bigcup_{j=1}^{i-1} \mathcal{P}_j$.

if $cost(f) \leq \sqrt{m}$ **then**

Decompose f into flows on s_i - t_i -paths.

Use this set of flow paths, say \mathcal{P}_i , to route d_i units of flow from s_i to t_i in the network with original arc capacities.

end

else /* The case where no flow f exists or $cost(f) > \sqrt{m}$. */
| Reject commodity i and do not route any flow from s_i to t_i .

end

Compute an integral flow f' that routes k_i units from s_i to t_i in D with arc capacities $u_{\mathcal{P}'}$ as above, for $\mathcal{P}' := \bigcup_{j=1}^{i-1} \mathcal{P}'_j$. (Set $f' = 0$, if no such flow exists.)

Decompose f' into flows on s_i - t_i -paths.

Use this set of flow paths, say \mathcal{P}'_i , to route d_i (or 0, if $f' = 0$) units of flow from s_i to t_i in the network with original arc capacities.

end

Return the one of $\mathcal{P} := \bigcup_{i=1}^K \mathcal{P}_i$ and $\mathcal{P}' := \bigcup_{i=1}^K \mathcal{P}'_i$ that routes more flow.

parallel arcs. \mathcal{P} , however, is empty as, when routing 1 unit of flow along both arcs, 2 ($> \sqrt{m} = \sqrt{2}$) arcs are used in their upper halves.

Theorem 1. Algorithm 1 runs in $O(K(T_{MCF} + mn))$ time, where T_{MCF} is the time needed to find an integral minimum cost flow.

Proof. Sorting the commodities uses $O(K \log K)$ time. The decomposition of flows can be done in $O(mn)$ time, see, e.g., [14]. The integral flow f' can be computed using an algorithm for the basic maximum s - t -flow problem, see, e.g., [18]. In particular, it can be done quicker than computing an integral minimum cost flow.

With the following well-known graph transformation (see, e.g., [18]) we can easily generalize our results for the (arc) disjoint flow problem to the *node disjoint flow problem* where we require node disjointness instead of arc disjointness for all paths routing the same commodity: Substitute each node v by two new nodes v_1 and v_2 together with an arc (v_1, v_2) . Then let all arcs entering v in the original digraph enter v_1 in the new digraph. Similarly, let all arcs leaving v in the original digraph leave v_2 in the new digraph. Capacities of newly introduced arcs (v_1, v_2) are infinite. Then arc disjointness in the new digraph is equivalent to node disjointness in the original digraph. Note that all our asymptotic results still hold for the node disjoint flow problem as the number of arcs in the new

digraph is $O(m + n) = O(m)$, for m and n being the number of arcs and nodes in the original digraph.

2.2 Performance Guarantee

In this section we prove that Algorithm 1 always terminates with a k -disjoint flow that routes at least $1/O(k_{\max}\sqrt{m}/k_{\min})$ of the maximum sum of simultaneously routable demands.

Theorem 2. *Algorithm 1 yields an $O(k_{\max}\sqrt{m}/k_{\min})$ -approximation algorithm for k -DFP.*

Proof. Consider an instance of k -DFP. Let \mathcal{O} be the set of commodities routed in an optimal solution OPT for the considered maximization problem and let $\mathcal{O}' \subseteq \mathcal{O}$ be those commodities that are rejected in Algorithm 1, i.e., those commodities that are not routed by any of \mathcal{P} or \mathcal{P}' . Let \mathcal{A} be the set of commodities that are routed in the final solution returned by Algorithm 1. To simplify the notation, we also use \mathcal{P} to denote the set of commodities that are routed on paths in \mathcal{P} .

For any set \mathcal{C} of commodities, we define $\|\mathcal{C}\| := \sum_{i \in \mathcal{C}} d_i$. It holds that

$$\|\mathcal{O} \setminus \mathcal{O}'\| \leq \sum_{P \in \mathcal{P}} f(P) + \sum_{P \in \mathcal{P}'} f(P) \leq 2\|\mathcal{A}\| \quad ,$$

because every commodity in $\mathcal{O} \setminus \mathcal{O}'$ is routed on paths in \mathcal{P} or in \mathcal{P}' .

Consider a commodity $j \in \mathcal{O}'$ and let \mathcal{O}_j be the set of paths that is used to route j in OPT. Since, in particular, the paths in \mathcal{O}_j are not used to route j in \mathcal{P} one of the following has to be true: (i) There is an arc a used by \mathcal{O}_j that would be overloaded by d_j/k_j together with the flow already routed in \mathcal{P} at the time of j 's rejection, or (ii) there are \sqrt{m} arcs in the paths of \mathcal{O}_j that j would have used in their upper halves in \mathcal{P} .

Let us examine reason (i) a bit closer. We show that the total demand that is rejected for reason (i) is at most $4k_{\max}\sqrt{m}\|\mathcal{P}\|/k_{\min}$, which is at most $4k_{\max}\sqrt{m}\|\mathcal{A}\|/k_{\min}$.

Since in Algorithm 1 the commodities are processed in order of decreasing d_i/k_i , the flow on a in \mathcal{P} at the time of j 's rejection is larger than $u(a)/2$. Consider a path $P \in \mathcal{P}$ participating in this flow and using a in its upper half. (We say that P uses a in its upper half if, at the time when P is chosen for \mathcal{P} , the flow on P together with the flow already routed along a in \mathcal{P} exceeds $u(a)/2$.) Let c be the commodity that is routed along P . We call c a *culprit* for j on a and define its *weight* as

$$w_c^j(a) := \frac{d_c \cdot d_j}{k_c \cdot k_j \cdot u(a)} \quad . \tag{1}$$

A commodity $c \in \mathcal{P}$ is a culprit on at most every arc e that it uses in its upper half and for at most every commodity $i \in \mathcal{O}'$ that is routed along e in OPT. We know that the number of arcs that c uses in their upper halves is bounded by

\sqrt{m} . Therefore, the total weight of c as a culprit for commodities in \mathcal{O}' over all arcs is

$$\begin{aligned}
 W_c := \sum_{\substack{e \in A, i \in \mathcal{O}': c \text{ is} \\ \text{culprit for } i \text{ on } e}} w_c^i(e) &\leq \sum_{\substack{e \in A: c \text{ uses } e \\ \text{in its upper half}}} \left(\sum_{\substack{i \in \mathcal{O}': i \text{ is routed} \\ \text{along } e \text{ in OPT}} w_c^i(e) \right) \\
 &\leq \sum_{\substack{e \in A: c \text{ uses } e \\ \text{in its upper half}} \frac{d_c}{k_c} \leq \sqrt{m} \cdot \frac{d_c}{k_c} \quad , \quad (2)
 \end{aligned}$$

where in the second inequality we use that the sum of d_i/k_i over all i that are routed along e in OPT is at most $u(e)$, because OPT is a feasible k -disjoint flow.

Note that the sum of flow values of culprits for j on a is at least $u(a)/4$. In case $d_j/k_j > u(a)/4$, this is true by the order of commodities. For the case when $d_j/k_j \leq u(a)/4$, it is true because otherwise we could route j along a without violating a 's capacity. (Note that if the sum of flow values of culprits for j on a was less than $u(a)/4$, then at the time of j 's rejection the total amount of flow routed along a would be at most $u(a)/2 + u(a)/4 = 3u(a)/4$.)

Let \mathcal{Q} bet the set of culprits for j on a . It follows that the total weight of culprits for j over all arcs is

$$\begin{aligned}
 W^j := \sum_{\substack{e \in A, c \in \mathcal{P}: c \text{ is} \\ \text{culprit for } j \text{ on } e}} w_c^j(e) &\geq \sum_{c \in \mathcal{Q}} w_c^j(a) = \sum_{c \in \mathcal{Q}} \frac{d_c \cdot d_j}{k_c \cdot k_j \cdot u(a)} \\
 &\geq \frac{u(a)}{4} \cdot \frac{d_j}{k_j \cdot u(a)} \geq \frac{d_j}{4k_{\max}} \quad . \quad (3)
 \end{aligned}$$

From (2) we get that the overall total weight of culprits for commodities in \mathcal{O}' is

$$\sum_{c \in \mathcal{P}} W_c \leq \sum_{c \in \mathcal{P}} \sqrt{m} \cdot \frac{d_c}{k_{\min}} = \sqrt{m} \cdot \frac{\|\mathcal{P}\|}{k_{\min}} \quad . \quad (4)$$

It follows from (3) and (4) that

$$\begin{aligned}
 \sum_{\substack{j \in \mathcal{O}': j \text{ is rejected} \\ \text{for reason (i)}}} d_j &\leq 4k_{\max} \sum_{\substack{j \in \mathcal{O}': j \text{ is rejected} \\ \text{for reason (i)}}} W^j \\
 &\leq 4k_{\max} \sum_{j \in \mathcal{O}'} W^j = 4k_{\max} \sum_{c \in \mathcal{P}} W_c \leq 4k_{\max} \sqrt{m} \frac{\|\mathcal{P}\|}{k_{\min}} \quad , \quad (5)
 \end{aligned}$$

i.e., the total demand that is rejected for reason (i) is at most $4k_{\max} \sqrt{m} \|\mathcal{P}\| / k_{\min} \leq 4k_{\max} \sqrt{m} \|\mathcal{A}\| / k_{\min}$.

For the analysis of rejection reason (ii), first note that an arc a that j would have used in its upper half in \mathcal{P} has capacity $u(a) < 2d_j/k_j$ or carried flow in \mathcal{P} at the time of j 's rejection. By the order of commodities in Algorithm 1, this flow would use at least a quarter of a 's capacity, as otherwise j would not use

a in its upper half. Therefore, we can state the two cases as follows: (a) There are more than $\sqrt{m}/2$ arcs a in the paths of \mathcal{O}_j such that $d_j/k_j > u(a)/2$ or (b) there are $\sqrt{m}/2$ arcs a in the paths of \mathcal{O}_j each with flow value at least $u(a)/4$ in \mathcal{P} .

Obviously, in OPT, we have less than $2\sqrt{m}$ commodities such that each of them uses more than half of the capacity of more than $\sqrt{m}/2$ arcs, as otherwise at least one arc would be overloaded—a contradiction. Choose a commodity $I \in \mathcal{P}'$ such that $d_I/k_I = \max_{i \in \mathcal{P}'} d_i/k_i$. Note that d_I/k_I is at least as big as the biggest ratio d_i/k_i routed in OPT. Therefore, the total demand of commodities rejected for reason (ii-a) is

$$\begin{aligned} \sum_{\substack{j \in \mathcal{O}' : j \text{ is rejected} \\ \text{for reason (ii-a)}}} d_j &\leq k_{\max} \sum_{\substack{j \in \mathcal{O}' : j \text{ is rejected} \\ \text{for reason (ii-a)}}} \frac{d_j}{k_j} \leq k_{\max} \cdot 2\sqrt{m} \cdot \frac{d_I}{k_I} \\ &\leq k_{\max} \cdot \frac{2\sqrt{m}}{k_{\min}} \cdot \sum_{P \in \mathcal{P}'} f(P) \leq k_{\max} \cdot \frac{2\sqrt{m}}{k_{\min}} \cdot \|\mathcal{A}\| \end{aligned}$$

For case (ii-b), we denote the $M \geq \sqrt{m}/2$ arcs in \mathcal{O}_j , for which \mathcal{P} uses at least a quarter of their capacities, as a_1, \dots, a_M . Now the analysis is similar with the one in rejection reason (i): We call a commodity $c \in \mathcal{P}$ a *culprit* for j if it is routed along at least one of a_1, \dots, a_M . The weight of this culprit for j on arc a_l is $w_c^j(a_l)$ as defined in (1).

Every commodity $c \in \mathcal{P}$ is a culprit on at most every arc e it is routed on and for at most every commodity $i \in \mathcal{O}'$ routed along e in OPT. Using similar arguments as in (2), for c as a culprit for commodities in \mathcal{O}' , we have a total weight of at most $d_c m/k_c \leq d_c m/k_{\min}$, where we use that c is routed along at most m arcs due to the arc disjointness of its flow paths.

Recall that j has culprits on all arcs a_1, \dots, a_M . Let \mathcal{Q}_l be the set of culprits for j on a_l , $l = 1, \dots, M$. By definition of case (ii-b), the total flow sent through a_l along paths in \mathcal{Q}_l is at least $u(a_l)/4$. It follows that j has culprits with total weight

$$W^j = \sum_{l=1}^M \sum_{q \in \mathcal{Q}_l} \frac{d_q \cdot d_j}{k_q \cdot k_j \cdot u(a_l)} \geq \sum_{l=1}^M \frac{u(a_l)}{4} \cdot \frac{d_j}{k_j} \cdot \frac{1}{u(a_l)} \geq \frac{\sqrt{m}}{2} \cdot \frac{d_j}{4k_{\max}}$$

With arguments similar to those in (5), we can conclude that the total demand that is rejected for reason (ii-b) is at most $8k_{\max}\sqrt{m}\|\mathcal{A}\|/k_{\min}$.

Therefore, the total sum of rejected demands is at most $14k_{\max}\sqrt{m}\|\mathcal{A}\|/k_{\min}$. So we conclude

$$\begin{aligned} \|\mathcal{O}\| &\leq \|\mathcal{O}'\| + \|\mathcal{O} \setminus \mathcal{O}'\| \leq 14k_{\max}\sqrt{m}\|\mathcal{A}\|/k_{\min} + 2\|\mathcal{A}\| \\ &= (14k_{\max}\sqrt{m}/k_{\min} + 2)\|\mathcal{A}\| \end{aligned}$$

2.3 Hardness of the Approximation

In this section we argue that, unless $\mathcal{P} = \mathcal{NP}$, the approximation factor of $O(k_{\max}\sqrt{m}/k_{\min})$ is best possible for the k -DFP, for instances where k_{\max}/k_{\min}

is constant. Note that, if k_{\max}/k_{\min} is constant, Algorithm 1 yields an $O(\sqrt{m})$ -approximation algorithm.

We use a result by Guruswami et al. [8] to prove the following theorem.

Theorem 3. *For any $\epsilon > 0$, it is \mathcal{NP} -hard to approximate the k -DFP within $O(m^{1/2-\epsilon})$.*

Proof. Note that with $d_i = k_i = 1$, for all commodities i , and arc capacities equal to 1, we obtain the arc disjoint paths problem as a special case of the k -DFP. In [8], it is proven that it is \mathcal{NP} -hard to approximate the arc disjoint paths problem within $O(m^{1/2-\epsilon})$, for any $\epsilon > 0$.

3 Conclusion

We present the first approximation algorithm for the k -DFP that is independent from any graph parameters or restrictions on the sizes of demands or on the arc capacities. Moreover, it is the first algorithm handling different bounds on the permitted number of flow paths for different commodities. Our algorithm stands out due to its simplicity. Its performance guarantee of $O(k_{\max}\sqrt{m}/k_{\min})$ is best possible for instances where k_{\max}/k_{\min} is constant. However, it is an interesting open question if it is best possible in general. For the unsplittable flow problem, Kolman and Scheideler [13] obtain an approximation factor of $O(\sqrt{m})$, which is best possible. Although the k -DFP seems to be harder due to the challenge of finding a set of flow paths for every commodity such that all paths for the same commodity are arc disjoint, it is still justified to ask if the approximation factor of $O(\sqrt{m})$ is also achievable for the k -DFP.

References

1. Aggarwal, G.C., Orlin, J.B.: On multiroute maximum flows in networks. *Networks* 39, 43–52 (2002)
2. Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: *Network Flows*. Prentice-Hall, Englewood Cliffs (1993)
3. Aneja, Y.P., Chandrasekaran, R., Nair, K.P.K.: Parametric analysis of overall mincuts and applications in undirected networks. *Information Processing Letters* 85, 105–109 (2003)
4. Bagchi, A., Chaudary, A., Scheideler, C., Kolman, P.: Algorithms for fault-tolerant routing in circuit switched networks. In: *Fourteenth ACM Symposium on Parallel Algorithms and Architectures* (2002)
5. Baier, G.: *Flows with Path Restrictions*. PhD thesis, TU Berlin (2003)
6. Baier, G., Köhler, E., Skutella, M.: On the k -splittable flow problem. *Algorithmica* 42, 231–248 (2005)
7. Ford, L.R., Fulkerson, D.R.: Maximal flow through a network. *Canadian Journal of Mathematics* 8, 399–404 (1956)
8. Guruswami, V., Khanna, S., Rajaraman, R., Shepherd, B., Yannakakis, M.: Near-optimal hardness results and approximation algorithms for edge-disjoint paths and related problems. *Journal of Computer and System Sciences* 67, 473–496 (2003)

9. Kishimoto, W.: A method for obtaining the maximum multiroute flows in a network. *Networks* 27, 279–291 (1996)
10. Kishimoto, W., Takeuchi, M.: On m -route flows in a network. *IEICE Transactions* (1993) (in Japanese)
11. Kleinberg, J.M.: *Approximation Algorithms for Disjoint Path Problems*. PhD thesis, Massachusetts Institute of Technology (May 1996)
12. Kolliopoulos, S.G.: Edge-disjoint paths and unsplittable flow. In: Gonzalez, T.F. (ed.) *Handbook of Approximation Algorithms and Metaheuristics*. Chapman & Hall/CRC, Boca Raton (2007)
13. Kolman, P., Scheideler, C.: Improved bounds for the unsplittable flow problem. In: *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 184–193 (2002)
14. Korte, B., Vygen, J.: *Combinatorial Optimization. Theory and Algorithms*. Springer, Berlin (2000)
15. Martens, M.: *Path-Constrained Network Flows*. PhD thesis, Universität Dortmund (2007)
16. Martens, M., Skutella, M.: Flows on few paths: Algorithms and lower bounds. *Networks* 48(2), 68–76 (2006)
17. Martens, M., Skutella, M.: Length-bounded and dynamic k -splittable flows. In: *Operations Research Proceedings 2005*, pp. 297–302 (2006)
18. Schrijver, A.: *Combinatorial Optimization. Polyhedra and Efficiency*. Springer, Berlin (2003)

Minimizing AND-EXOR Expressions for Multiple-Valued Two-Input Logic Functions (Extended Abstract)

Takaaki Mizuki¹, Hitoshi Tsubata², and Takao Nishizeki²

¹ Cyberscience Center, Tohoku University,
Aramaki-Aza-Aoba 6-3, Aoba-ku, Sendai 980-8578, Japan
tm-paper@rd.isc.tohoku.ac.jp

² Graduate School of Information Sciences, Tohoku University, Aramaki-Aza-Aoba
6-6-05, Aoba-ku, Sendai, 980-8579, Japan

Abstract. A minimum ESOP (Exclusive-OR Sum-of-Products) form of a logic function f is an AND-EXOR 2-level expression of f having the minimum number of product terms. In the paper we deal with multiple-valued 2-input logic functions f , and give an algorithm to find a minimum ESOP form of a given function f in polynomial time.

1 Introduction

An ESOP (Exclusive-OR Sum-of-Products) form of a logic function f is an AND-EXOR 2-level expression of f , i.e., a logical expression that combines products of literals by Exclusive-ORs. For example,

$$f(x_1, x_2, x_3) = (x_1 \wedge \bar{x}_2 \wedge x_3) \oplus (\bar{x}_1 \wedge \bar{x}_2 \wedge \bar{x}_3) \oplus (x_2 \wedge \bar{x}_3) \quad (1)$$

is an ESOP form defining a (2-valued) 3-input logic function f . The logic function f can be expressed by another ESOP form, say

$$f(x_1, x_2, x_3) = x_1 \bar{x}_2 \oplus \bar{x}_3. \quad (2)$$

(Hereafter, as in this expression, we omit the conjunction symbol \wedge .) The ESOP form in Eq. (1) has exactly three product terms, while the ESOP form in Eq. (2) has only two product terms. Thus, there exists a minimization problem regarding ESOP forms. This paper deals with such a minimization problem; more specifically, we give an efficient algorithm to minimize ESOP forms for multiple-valued 2-input logic functions.

1.1 ESOP Forms

First of all, we formally define “multiple-valued input logic functions” and “literals.” Throughout the paper, for a positive integer m , we define \mathbb{Z}_m as follows:

$$\mathbb{Z}_m \stackrel{\text{def}}{=} \{0, 1, \dots, m-1\}.$$

Let $n \geq 1$ be the number of logical variables, and let $m_1, m_2, \dots, m_n \geq 2$ be n positive integers. Then, a function

$$f(x_1, x_2, \dots, x_n)$$

such that

$$f : \mathbb{Z}_{m_1} \times \mathbb{Z}_{m_2} \times \dots \times \mathbb{Z}_{m_n} \rightarrow \{0, 1\}$$

is called a *multiple-valued n -input logic function*; in particular, when $m_1 = m_2 = \dots = m_n = m$, we call it an *m -valued n -input logic function*. (Needless to say, when $m = 2$, it is a 2-valued n -input logic function and hence is a so-called Boolean function.) Furthermore, for an i -th variable x_i , $1 \leq i \leq n$, and a subset $S \subseteq \mathbb{Z}_{m_i}$, we define a function $x_i^S : \mathbb{Z}_{m_i} \rightarrow \{0, 1\}$, called a *literal*, as follows:

$$x_i^S(x_i) = \begin{cases} 1 & \text{if } x_i \in S; \\ 0 & \text{otherwise.} \end{cases}$$

We often denote $x_i^S(x_i)$ simply by x_i^S . When $S = \mathbb{Z}_{m_i}$, the literal x_i^S is just the constant 1; also, when $S = \emptyset$, the literal $x_i^S (= x_i^\emptyset)$ is just the constant 0. For instance, when $m_i = 2$, i.e., $\mathbb{Z}_{m_i} = \{0, 1\}$, there are exactly four literals $x_i^{\{0,1\}}$, $x_i^{\{0\}}$, $x_i^{\{1\}}$ and x_i^\emptyset , which are often denoted by 1, \bar{x}_i , x_i and 0, respectively.

A product $x_1^{S_1} x_2^{S_2} \dots x_n^{S_n}$ of literals is called a *product term*. If a logic function $f(x_1, x_2, \dots, x_n)$ has a logical expression

$$F = \bigoplus_{(S_1, S_2, \dots, S_n)} x_1^{S_1} x_2^{S_2} \dots x_n^{S_n} \tag{3}$$

which combines product terms by Exclusive-ORs, then the expression F is called an *ESOP form*. If an ESOP form F has a product term containing a literal $x_i^\emptyset (= 0)$, then the resulting ESOP form F' obtained by removing such a product term represents the same logic function as the original ESOP form F . For an ESOP form F , we denote by $\tau(F)$ the number of product terms in F . A *minimum ESOP form* of a logic function f is an ESOP form having the minimum number of product terms among all possible ESOP forms representing f .

1.2 Known Results

For many decades, the problem of minimization or simplification of ESOP forms has attracted much attention of the researchers in the logic design community. (A comprehensive survey appears in a book [6].) Although no efficient algorithm to minimize ESOP forms has been known, many good heuristic algorithms to simplify ESOP forms have been proposed (e.g. [15][7][8][11]). On the other hand, there also exist efficient exact minimization algorithms which efficiently work only for a limited (small) number of variables or product terms (e.g. [3][9][10]).

Historically, the binary case of $m_1 = m_2 = \dots = m_n = 2$, namely ESOP forms for 2-valued input logic functions have been much investigated, of course; the most famous ESOP form is probably a Reed-Muller expression. For such a 2-valued input case, there are many good heuristic (or exact) algorithms to simplify

(or minimize) ESOP forms (e.g. [1,3,9,11]). In particular, the best known upper bound on the number $\tau(F)$ of product terms in a minimum ESOP form F for any 2-valued n -input logic function is $\tau(F) \leq 29 \cdot 2^{n-7}$ (provided that $n \geq 7$) [2].

On the other hand, there are relatively a small number of papers dealing with multiple-valued input logic functions, but there are a few works on the case where integers m_i are larger than 2. In particular, the case of $m_1 = m_2 = \dots = m_n = 4$, namely ESOP forms for 4-valued input logic functions have been greatly studied, e.g. [5,7]; it is motivated by improving input decoders in PLA (Programmable Logic Array) structures. Furthermore, the case where $m_1 = m_2 = \dots = m_{n-1} = 2$ and $m_n \geq 3$ has been studied in [10].

1.3 Our Results

As mentioned in the previous subsection, no efficient ESOP minimization algorithm for general logic functions has been known; in particular, for the multiple-valued input case, every existing efficient minimization algorithm, to our knowledge, has a limitation in the input sizes m_i . In this paper, instead of restricting the input sizes m_i in multiple-valued input logic functions, we fix the number n of variables to 2. We thus deal with m -valued 2-input logic functions, and give an algorithm to find a minimum ESOP form of any given function in polynomial time in m , say time $O(m^3)$, where m is any integer larger than 1.

It is known that the minimization of ESOP forms of m -valued 2-input logic functions, which this paper addresses, can be applied to improving a cryptographic protocol [4], as follows. The cost (communication complexity) of the cryptographic protocol developed in [4] to securely compute a function $f(a, b)$ is proportional to the number $\tau(F)$ of product terms in an ESOP form F of f . Thus, if one can find a minimum ESOP form of f , then one can achieve the most efficient secure computation by the protocol. Therefore, applying the results in this paper to the protocol proposed in [4], one can execute the protocol most efficiently.

The remainder of the paper is organized as follows. In Section 2, we present some preliminaries necessary to explain our algorithm. In Section 3, we introduce a method to express an ESOP form of an m -valued 2-input logic function in a matrix form. This matrix-based expression helps us to easily and intuitively understand the minimization of ESOP forms. In Section 4, we present our efficient algorithm to find a minimum ESOP form of any given m -valued 2-input logic function by using elementary row operations for matrices. This paper concludes in Section 5 with some discussions.

2 Preliminaries

In this section, we define some terms and present some of the known results.

2.1 Multiple-Valued Shannon Expansion

Let $f(a, b)$ be an m -valued 2-input logic function with variables a and b , that is, let

$$f : \mathbb{Z}_m \times \mathbb{Z}_m \rightarrow \{0, 1\}.$$

Then, throughout this paper, we call the ESOP form

$$f(a, b) = a^{\{0\}}b^{T_1} \oplus a^{\{1\}}b^{T_2} \oplus \dots \oplus a^{\{m-1\}}b^{T_m}$$

the *multiple-valued Shannon expansion* of f . It should be noted that $T_i \subseteq \mathbb{Z}_m$, $i \in \{1, 2, \dots, m\}$, is uniquely determined as follows:

$$T_i = \{b \in \mathbb{Z}_m \mid f(i - 1, b) = 1\}.$$

Consider for example a 5-valued 2-input logic function $h(a, b)$, whose truth table is given in Table 1. The multiple-valued Shannon expansion of h is

$$h(a, b) = a^{\{0\}}b^{\{0,3,4\}} \oplus a^{\{1\}}b^{\{1,4\}} \oplus a^{\{2\}}b^{\{1,3\}} \oplus a^{\{3\}}b^{\{0,3,4\}} \oplus a^{\{4\}}b^{\{0,1,3\}}. \quad (4)$$

Table 1. A truth table for the 5-valued 2-input logic function $h(a, b)$

		b				
		0	1	2	3	4
	0	1	0	0	1	1
	1	0	1	0	0	1
a	2	0	1	0	1	0
	3	1	0	0	1	1
	4	1	1	0	1	0

Let F be the multiple-valued Shannon expansion of an m -valued 2-input logic function f , then the number $\tau(F)$ of the product terms in F satisfies $\tau(F) = m$ (before removing a product term containing a constant literal b^\emptyset).

2.2 Transformation Rules for ESOP Forms

One of the most famous currently known algorithms to simplify ESOP forms is EXMIN2, which was developed by Sasao [5]. The transformation rule for ESOP forms described in the following Theorem 1 is one of the rules utilized by the algorithm EXMIN2. Hereafter, the binary operator \oplus for two sets denotes the symmetric difference of the two sets, that is,

$$S \oplus T = (\overline{S} \cap T) \cup (S \cap \overline{T}).$$

Theorem 1 ([5]). For any four literals $a^{S_p}, b^{T_p}, a^{S_q}, b^{T_q}$ of two variables a and b ,

$$a^{S_p}b^{T_p} \oplus a^{S_q}b^{T_q} = a^{S_p \oplus S_q}b^{T_p} \oplus a^{S_q}b^{T_p \oplus T_q}$$

holds, where $S_p, T_p, S_q, T_q \subseteq \mathbb{Z}_m$.

Note that, according to Theorem 1, if $T_p = T_q$, then

$$a^{S_p}b^{T_p} \oplus a^{S_q}b^{T_p} = a^{S_p \oplus S_q}b^{T_p} \oplus a^{S_q}b^\emptyset = a^{S_p \oplus S_q}b^{T_p},$$

and hence the number of product terms decreases by exactly 1. For example, applying the transformation rule in Theorem 1 to the first and fourth product terms in the ESOP form in Eq. (4) results in

$$\begin{aligned}
 h(a, b) &= a^{\{0\} \oplus \{3\}} b^{\{0,3,4\}} \oplus a^{\{1\}} b^{\{1,4\}} \oplus a^{\{2\}} b^{\{1,3\}} \oplus a^{\{3\}} b^{\{0,3,4\} \oplus \{0,3,4\}} \oplus a^{\{4\}} b^{\{0,1,3\}} \\
 &= a^{\{0,3\}} b^{\{0,3,4\}} \oplus a^{\{1\}} b^{\{1,4\}} \oplus a^{\{2\}} b^{\{1,3\}} \oplus a^{\{4\}} b^{\{0,1,3\}}.
 \end{aligned}
 \tag{5}$$

In this paper, as seen later in Sections 3 and 4, applying the transformation rule given in Theorem 1, we propose an efficient algorithm to find a minimum ESOP form of any given m -valued 2-input logic function.

3 ESOP Matrices

In this section, we propose a method for expressing an ESOP form of an m -valued 2-input logic function in a Boolean matrix. The method makes it easier for us to intuitively understand the transformations of ESOP forms.

We begin with an example; consider the following ESOP form of the 5-valued 2-input logic function h (already seen in Eq. (5)):

$$a^{\{0,3\}} b^{\{0,3,4\}} \oplus a^{\{1\}} b^{\{1,4\}} \oplus a^{\{2\}} b^{\{1,3\}} \oplus a^{\{4\}} b^{\{0,1,3\}}.$$

Given such a 5-valued ESOP form having 4 product terms, we construct a Boolean 4×10 matrix as follows:

$$\left(\begin{array}{cc|cc} 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \end{array} \right) \begin{array}{l} \leftarrow a^{\{0,3\}} b^{\{0,3,4\}} \\ \leftarrow a^{\{1\}} b^{\{1,4\}} \\ \leftarrow a^{\{2\}} b^{\{1,3\}} \\ \leftarrow a^{\{4\}} b^{\{0,1,3\}} \end{array}$$

which represents the ESOP form above; the first row of the Boolean matrix corresponds to the first term $a^{\{0,3\}} b^{\{0,3,4\}}$ in the ESOP form, i.e., the five elements in the left half of the first row correspond to the literal $a^{\{0,3\}}$, and the five elements in the right half correspond to the literal $b^{\{0,3,4\}}$, and so on. (Notice that each literal is described by a bit pattern of length 5.)

Generally, let $F = \bigoplus_{i=1}^t a^{S_i} b^{T_i}$ be an m -valued ESOP form, then we call the following $t \times 2m$ Boolean matrix H the *ESOP matrix* of F :

$$H = \left(\begin{array}{cccc|cccc} \ell_{11} & \ell_{12} & \cdots & \ell_{1m} & r_{11} & r_{12} & \cdots & r_{1m} \\ \ell_{21} & \ell_{22} & \cdots & \ell_{2m} & r_{21} & r_{22} & \cdots & r_{2m} \\ \vdots & \vdots & \cdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ \ell_{t1} & \ell_{t2} & \cdots & \ell_{tm} & r_{t1} & r_{t2} & \cdots & r_{tm} \end{array} \right)$$

where

$$\ell_{ij} = a^{S_i}(j - 1) = \begin{cases} 1 & \text{if } j - 1 \in S_i; \\ 0 & \text{if } j - 1 \notin S_i \end{cases}$$

and

$$r_{ij} = b^{T_i}(j - 1) = \begin{cases} 1 & \text{if } j - 1 \in T_i; \\ 0 & \text{if } j - 1 \notin T_i \end{cases}$$

for every $i \in \{1, 2, \dots, t\}$ and $j \in \{1, 2, \dots, m\}$.

Given an ESOP form of t product terms, its corresponding $t \times 2m$ ESOP matrix is uniquely determined. Conversely, given a $t \times 2m$ Boolean matrix, the corresponding ESOP form of t product terms is also uniquely determined.

Hereafter, for a $t \times 2m$ ESOP matrix H , partitioning H into the left block H^L and the right block H^R , we often write

$$H = (H^L | H^R),$$

where H^L and H^R are $t \times m$ matrices. For example, let H be the ESOP matrix of the multiple-valued Shannon expansion

$$f(a, b) = a^{\{0\}}b^{T_1} \oplus a^{\{1\}}b^{T_2} \oplus \dots \oplus a^{\{m-1\}}b^{T_m}$$

of an m -valued 2-input logic function f , then H^L must be an identity matrix (unit matrix) of size m , i.e., H must be like

$$H = (I | H^R),$$

where I denotes the identity matrix (also in the sequel).

Furthermore, for each of the left block H^L and the right block H^R of a $t \times 2m$ ESOP matrix H , using row vectors ℓ_i and r_i of length m , we often write

$$H = (H^L | H^R) = \left(\begin{array}{c|c} \ell_1 & r_1 \\ \ell_2 & r_2 \\ \vdots & \vdots \\ \ell_t & r_t \end{array} \right).$$

As will be seen in Section 4, our algorithm uses the following two transformation rules (named R1 and R2) for ESOP forms:

(R1) $a^{S_p}b^{T_p} \oplus a^{S_q}b^{T_q} = a^{S_p \oplus S_q}b^{T_p} \oplus a^{S_q}b^{T_p \oplus T_q}$ (Theorem 1);

(R2) $a^{S_p}b^{T_p} \oplus a^{S_q}b^{T_q} = a^{S_q}b^{T_q} \oplus a^{S_p}b^{T_p}$ (commutativity of Exclusive-OR).

Considering the two rules above applied to ESOP matrices, we naturally get the following two definitions.

Definition 1. Given an ESOP matrix of t rows, applying rule R1 to the p -th and q -th rows means the following row operation:

$$\left(\begin{array}{c|c} \ell_1 & r_1 \\ \vdots & \vdots \\ \ell_p & r_p \\ \vdots & \vdots \\ \ell_q & r_q \\ \vdots & \vdots \\ \ell_t & r_t \end{array} \right) \xrightarrow{\text{R1}_{(p,q)}} \left(\begin{array}{c|c} \ell_1 & r_1 \\ \vdots & \vdots \\ \ell_p \oplus \ell_q & r_p \\ \vdots & \vdots \\ \ell_q & r_p \oplus r_q \\ \vdots & \vdots \\ \ell_t & r_t \end{array} \right),$$

where the operator \oplus represents bitwise Exclusive-OR of two row vectors.

Definition 2. Given an ESOP matrix of t rows, applying rule R2 to the p -th and q -th rows means the following row operation:

$$\left(\begin{array}{c|c} \ell_1 & r_1 \\ \vdots & \vdots \\ \ell_p & r_p \\ \vdots & \vdots \\ \ell_q & r_q \\ \vdots & \vdots \\ \ell_t & r_t \end{array} \right) \xrightarrow{\text{R2}_{(p,q)}} \left(\begin{array}{c|c} \ell_1 & r_1 \\ \vdots & \vdots \\ \ell_q & r_q \\ \vdots & \vdots \\ \ell_p & r_p \\ \vdots & \vdots \\ \ell_t & r_t \end{array} \right).$$

Based on these two definitions, the following lemma immediately holds.

Lemma 1. Let F be an arbitrary ESOP form, and let H be the ESOP matrix of F . Assume that applying rule R1 (R2) to the p -th and q -th product terms in F results in an ESOP form F' , and that applying rule R1 (R2) to the p -th and q -th rows of H results in a matrix H' . Then, H' is the ESOP matrix of F' .

Note that rules R1 and R2 for an ESOP matrix H can be regarded just as the elementary row operations (on a Boolean matrix) for each of the left block H^L and the right block H^R of H .

4 Our Algorithm

We are now ready to present our algorithm.

Given a truth table of an m -valued 2-input logic function f as an input, the following algorithm outputs a minimum ESOP form of f .

[Our algorithm]

1. Find the multiple-valued Shannon expansion $\bigoplus_{i=1}^m a^{\{i-1\}} b^{T_i}$ of f from the truth table of f , and let

$$(I|H^R)$$

be its $m \times 2m$ ESOP matrix. (Recall that the left block of the ESOP matrix of a multiple-valued Shannon expansion is always an identity matrix I .)

2. Apply a series of rules R1 and R2 to the ESOP matrix so that the right block H^R is transformed into a Boolean matrix in row echelon form. (Using a known algorithm, e.g. Gaussian elimination algorithm, one can obtain such a row echelon form within an $O(m^2)$ number of transformations. Each transformation can be done in $O(m)$ bit operations.) Note that, since these transformations are elementary row operations for each of the left and right blocks, the rank of the right block never changes, and hence its rank remains $\text{rank}(H^R)$ (after the series of rules R1 and R2), where $\text{rank}(M)$ denotes the rank of a matrix M .

3. Note that the current ESOP matrix in row echelon form has an all-zero submatrix O in the lower part of its right block as follows:

$$\left(\begin{array}{c|c} * & * \\ * & O \end{array} \right) \text{rank}(H^R).$$

Construct the ESOP form corresponding to this ESOP matrix, and remove all the $m - \text{rank}(H^R)$ terms containing constant 0. The resulting ESOP form is the output of our algorithm.

We now demonstrate the execution of our algorithm with the 5-valued 2-input logic function h which was given in Table 1. In step 1 of our algorithm, the multiple-valued Shannon expansion of h is given in Eq. (4), and hence we have a 5×10 Boolean matrix

$$\left(\begin{array}{ccccc|ccccc} 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 \end{array} \right)$$

as its ESOP matrix. In step 2, applying a series of four transformations $R1_{(1,4)}$, $R1_{(1,5)}$, $R1_{(2,3)}$ and $R1_{(2,5)}$ to the matrix, we make the right block of the ESOP matrix be a Boolean matrix in row echelon form

$$\left(\begin{array}{ccccc|ccccc} 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{array} \right).$$

In step 3, from the ESOP matrix above, our algorithm outputs the following ESOP form:

$$a^{\{0,3,4\}}b^{\{0,3,4\}} \oplus a^{\{1,2,4\}}b^{\{1,4\}} \oplus a^{\{2\}}b^{\{3,4\}},$$

which is a minimum ESOP form of h as will be guaranteed in Theorem 2.

Since applying rules R1 and R2 in step 2 of our algorithm can be regarded exactly as elementary row operations for each of the left and right blocks as mentioned above, the rank of the right block never changes, and hence its rank remains $\text{rank}(H^R)$. Therefore, the following Lemma 2 holds.

Lemma 2. *Let f be an m -valued 2-input logic function, and let*

$$(I|H^R)$$

be the ESOP matrix of its multiple-valued Shannon expansion. Then, our algorithm outputs an ESOP form F such that $\tau(F) = \text{rank}(H^R)$.

Using Lemma 2, one can verify the correctness of our algorithm and obtains the following Theorem 2.

Theorem 2. *For every m -valued 2-input logic function f , our algorithm outputs a minimum ESOP form.*

Proof. Omitted due to the page limitation. □

5 Conclusions

In this paper, we first introduced a method for expressing an ESOP form as a matrix, and then, utilizing the method, we proposed an algorithm to find a minimum ESOP form of any given m -valued 2-input logic function in $O(m^3)$ bit operations.

Lemma 2 and Theorem 2 also imply that, given an m -valued 2-input logic function f , the minimum number of product terms among all the ESOP forms of f is equal to $\text{rank}(H^R)$, where

$$(I|H^R)$$

is the ESOP matrix of the multiple-valued Shannon expansion of f . Furthermore, even if a logic function f is given in an ESOP form F which is not necessarily that of the multiple-valued Shannon expansion of f , one can efficiently find a minimum ESOP form of f having $\min\{\text{rank}(H^L), \text{rank}(H^R)\}$ product terms by extending the results in Section 4, where

$$(H^L|H^R)$$

is the ESOP matrix of F .

We have so far considered the minimization of ESOP forms of m -valued 2-input logic functions, namely only for the case of $m_1 = m_2 = m$. However, even for the case of $m_1 \neq m_2$, i.e., for multiple-valued 2-input logic functions $f : \mathbb{Z}_{m_1} \times \mathbb{Z}_{m_2} \rightarrow \{0, 1\}$ with $m_1 \neq m_2$, one can easily construct an efficient minimization algorithm by redefining ESOP matrices as $t \times (m_1 + m_2)$ matrices.

Acknowledgments

We thank Dr. Xiao Zhou and Dr. Takehiro Ito for their valuable discussions and suggestions. This work was supported by KAKENHI (19700002).

References

1. Fleisher, H., Tavel, M., Yeager, J.: A computer algorithm for minimizing Reed-Muller canonical forms. *IEEE Transactions on Computers* 36(2), 247–250 (1987)
2. Gaidukov, A.: Algorithm to derive minimum ESOP for 6-variable function. In: *Proceedings of the Fifth International Workshop on Boolean Problems, Freiberg (2002)*
3. Hirayama, T., Nishitani, Y., Sato, T.: A faster algorithm of minimizing AND-EXOR expressions. *IEICE Trans. Fundamentals* E85-A(12), 2708–2714 (2002)
4. Mizuki, T., Otagiri, T., Sone, H.: An application of ESOP expressions to secure computations. *Journal of Circuits, Systems, and Computers* 16(2), 191–198 (2007)
5. Sasao, T.: EXMIN2: a simplification algorithm for exclusive-or sum-of-products expressions for multiple-valued-input two-valued-output functions. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 12(5), 621–632 (1993)

6. Sasao, T.: *Switching Theory for Logic Synthesis*. Kluwer Academic Publishers, Boston (1999)
7. Sasao, T., Besslich, P.: On the complexity of mod-2 sum PLA's. *IEEE Transactions on Computers* 39(2), 262–266 (1990)
8. Song, N., Perkowski, M.A.: Minimization of exclusive sum-of-products expressions for multiple-valued input, incompletely specified functions. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 15(4), 385–395 (1996)
9. Stergiou, S., Papakonstantinou, G.: Exact minimization of ESOP expressions with less than eight product terms. *Journal of Circuits, Systems and Computers* 13(1), 1–15 (2004)
10. Stergiou, S., Voudouris, D., Papakonstantinou, G.: Multiple-value exclusive-or sum-of-products minimization algorithms. *IEICE Trans. Fundamentals* E87-A(5), 1226–1234 (2004)
11. Ye, Y., Roy, K.: An XOR-based decomposition diagram and its application in synthesis of AND/XOR networks. *IEICE Trans. Fundamentals* E80-A(10), 1742–1748 (1997)

Exact and Experimental Algorithms for a Huffman-Based Error Detecting Code*

Paulo Eustáquio Duarte Pinto¹, Fábio Protti^{2,**},
and Jayme Luiz Szwarcfter^{3,**}

¹ Instituto de Matemática e Estatística
Universidade Estadual do Rio de Janeiro
Rio de Janeiro, RJ, Brasil
pauloedp@ime.uerj.br

² Instituto de Matemática
Universidade Federal do Rio de Janeiro
Caixa Postal 68.530, CEP 21941-590, Rio de Janeiro, RJ, Brasil
fabiop@dcc.ufrj.br

³ Instituto de Matemática, Núcleo de Computação Eletrônica and COPPE-Sistemas
Universidade Federal do Rio de Janeiro
Caixa Postal 68511, CEP 21945-970, Rio de Janeiro, RJ, Brasil
jayme@nce.ufrj.br

Abstract. Even codes are Huffman based codes in which every encoding contains an even number of 1's, thus having the ability of detecting the occurrence of an odd number of 1-bit errors in the message. The motivation for defining such codes comes from a problem posed by Hamming in 1980. Even codes have been studied for the case in which symbols have uniform probabilities. In this work, we consider the general situation of arbitrary probabilities. An exact algorithm for constructing an optimal even code is described with complexity $O(n^3)$, where n is the number of symbols. Further, two heuristics for constructing nearly optimal even codes are presented, both requiring $O(n \log n)$ time. The cost of the even code constructed by the second heuristics is at most 16,7% higher than the cost of a Huffman code, for the same probabilities. However, computational experiments suggest that, for practical purposes, this value seems to be better: about 5% or less, for n large enough. This corresponds to the amount of redundancy to be added to the message in order to keep the ability of error detection. Concerning undetected errors, we obtain bounds on the probability of producing k consecutive erroneous symbols, suggesting that such a probability is small for large messages.

Keywords: Data compression, error detection, even codes, Huffman codes, Hamming codes.

* This work is partially supported by CNPq and FAPERJ.

** Partially supported by Conselho Nacional de Desenvolvimento Científico e Tecnológico - CNPq and Fundação de Amparo à Pesquisa do Estado do Rio de Janeiro - FAPERJ, Brazilian research agencies.

1 Introduction

Huffman codes [1] appear as one of the most traditional methods of coding. An important aspect of these codes is the possibility of handling encodings of variable sizes. A great number of extensions and variations of the classical Huffman codes have been described through the time. For instance, Faller [2], Gallager [3], Knuth [4], and Milidiú, Laber and Pessoa [5] addressed adaptative methods for the construction of Huffman trees. Huffman trees with minimum height were described by Schwartz [6]. The construction of Huffman type trees with length constraints was considered by Turpin and Moffat [7], Larmore and Hirschberg [8], and Milidiú and Laber [9,10]. On the other hand, Hamming formulated algorithms for the construction of error detecting codes [11]. Further, he posed the problem of describing an algorithm that would combine advantages of Huffman codes with the noise protection of Hamming codes. The idea is to define a prefix code in which the encodings contain redundancies that allow the detection of certain kinds of errors. This is equivalent to forbid some encodings which signal an error when received. Such a code is a Hamming-Huffman code, and its representing binary tree is a Hamming-Huffman tree. In a Huffman tree, all leaves correspond to encodings. In a Hamming-Huffman tree, there are encoding leaves and error leaves. Hitting an error leaf in the decoding process points out the existence of an error. The problem posed by Hamming is to devise a method for detecting the occurrence of an error of one bit, as illustrated in the following example from [11], p.76. Table 1 shows the symbols and their corresponding encodings. Figure 1 depicts the corresponding Hamming-Huffman tree. Error leaves are represented by black nodes. An error of one bit in any encoding would lead to an error leaf in the decoding process.

Table 1. Example of a Hamming-Huffman Code

Symbol	Encoding
a	000
b	0110
c	1010
d	1100
e	1111

Motivated by the above problem, we have proposed [12] a special prefix code, called even code, in which each encoding contains an even number of 1's. This code has the ability of detecting any odd error (an error formed by an odd number of corrupted bits in a message), and also the majority of even errors (defined analogously), as we shall see in Section 5. Moreover, an even code is much more powerful for detecting errors than a code created by simply appending one parity bit at the end of the encoded message.

In [12], the study was restricted to symbols having uniform probabilities. The present work considers the general situation of arbitrary probabilities. An

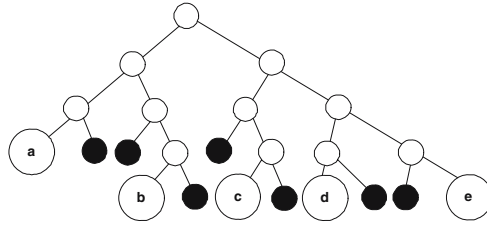


Fig. 1. A Hamming-Huffman tree

extended abstract containing part of the results appears in [13]. First, we describe an exact algorithm for constructing an optimal even code, for a given set of symbols, each one with a given probability. The algorithm employs dynamic programming and its complexity is $O(n^3)$, where n is the number of symbols. Next, we propose two heuristics for approximating an optimal code, based on Huffman’s algorithm. The time required for computing an even code, for both heuristics, is $O(n \log n)$. We show that the cost of an even code constructed by the second heuristics is at most 16,7% higher than the cost of a Huffman code for the same probabilities. That is, less than 16,7% higher than the corresponding optimal even code. However, for practical purposes, this value seems to be much better. In fact, several computational experiments obtained values less than 5%, except for small values of n . This corresponds to the amount of redundancy to be added to the message in order to keep the ability of error detection.

The structure of the paper is as follows. In Section 2 we describe the exact algorithm for constructing an optimal even code. The heuristics are formulated in Section 3. In Section 4 we present bounds for the cost difference between even trees and corresponding Huffman trees. A probabilistic model to evaluate the error detection capability of an even tree is developed in Section 5. In Section 6 we present experimental results on the costs of even trees and their error detection capabilities.

The following definitions are necessary.

Let $\mathcal{S} = \{s_1, \dots, s_n\}$ be a set of elements called *symbols*. Each $s_i \in \mathcal{S}$ is associated with a probability f_i . We assume $f_i \leq f_{i+1}$, for $1 \leq i < n$.

An *encoding* e_i for a symbol $s_i \in \mathcal{S}$ is a finite sequence of 0’s and 1’s, associated to s_i . Each 0 and 1 is a *bit* of e_i . The *parity* of e_i is the parity of the number of 1’s contained in e_i . A subsequence of e_i starting from its first bit is a *prefix* of e_i . The set of encodings for all symbols of \mathcal{S} is a *code* \mathcal{C} for \mathcal{S} . A code in which every encoding does not coincide with a prefix of any other encoding is a *prefix code*.

A *message* M is a sequence of symbols. The *encoded message* associated with M is the corresponding sequence of symbol encodings. The *parity* of an encoded message is the number of 1’s it contains.

A *binary tree* is a rooted tree T in which every node z other than the root is labelled *left* or *right* in such a way that any two siblings have different labels. Say that T is *trivial* when it consists of a single node. A *binary forest* is a set of binary trees. A *path* of T is a sequence of nodes z_1, \dots, z_t such that z_i is the parent of z_{i+1} , for $1 \leq i < t$. The value $t - 1$ is the *size* of the path, whereas

all z_i 's are *descendants* of z_1 . If z_1 is the root then z_1, \dots, z_t is a *root path* and, in addition, if z_t is a leaf, then z_1, \dots, z_t is a *root-leaf path* of T . The *depth* of a node is the size of the root path to it. For a node z of T , $T(z)$ denotes the *subtree of T rooted at z* , that is, the binary tree containing all descendants of z in T (including z itself). The *left subtree* of z is the subtree $T(z')$, where z' is the left child of z . Similarly, define the *right subtree* of z . The left and right subtrees of the root of T are denoted by T_L and T_R , respectively. A *strictly binary tree* is one in which every node is a leaf or has two children. A *full binary tree* is a strictly binary tree in which all root-leaf paths have the same size. A *complete binary tree* is a binary tree where the null subtrees are located at the last two levels. In a binary tree T , the edges of T leading to left children are labelled 0, whereas those leading to right children are labelled 1. The *parity* of a node z is the parity of the number of 1's among the edges forming the root path to z . A node is *even* or *odd*, according to its parity.

A (*binary tree*) *representation* of a code \mathcal{C} is a binary tree T such that there exists a one-to-one correspondence between encodings $e_i \in \mathcal{C}$ and root-leaf paths p_i of T in such a way that e_i is precisely the sequence of labels, 0 or 1, of the edges forming p_i . A code admits a binary tree representation if and only if it is a prefix code. Let d_i be the depth of the leaf of T associated to symbol s_i . Define the *cost* of T as the sum $c(T) = \sum_{i=1}^n f_i d_i$. Hence, the cost of a trivial tree is 0. An *optimal code* (tree) is one with minimum cost. A *full representation tree* of \mathcal{C} is a binary tree T^* obtained from the representation tree T of \mathcal{C} , by adding a new leaf as the second child of every node having exactly one child. The original leaves of T are the *encoding leaves*, whereas the newly introduced leaves are the *error leaves*. Clearly, in the case of Huffman trees, there are no error leaves.

An *even* (*odd*) *code* is a prefix code in which all encodings are even (odd). Similarly, an *even* (*odd*) *tree* is a tree representation of an even (odd) code. The trees T_2, T_3, T_4, T_{11} in Figure 2 are examples of even trees for 2, 3, 4, and 11 symbols, respectively (from left to right). The tree T_{11} is an optimal even tree for symbols with uniform probabilities.

It is easy to see that even codes can detect the occurrence of odd errors, as follows. Since all the encodings are even, the encoded message is also even. By changing the values of an odd number of bits, the encoded message becomes odd. This implies that, in the full tree representation of the code, either an error leaf will be hit during the decoding process, or the process terminates at some odd node of the tree. It should be noted that odd codes do not have this property.

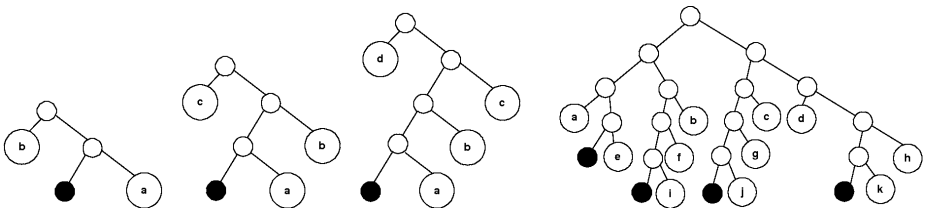


Fig. 2. Examples of even trees: T_2, T_3, T_4, T_{11}

For example, if we have a code $\mathcal{C} = \{1, 01\}$ and a message 01, if the value of the first bit changes to 1 the message would be erroneously decoded without pointing out an error.

2 Exact Algorithm

In this section, we describe an exact algorithm for constructing an optimal even tree for symbols with arbitrary probabilities. That is, our aim is to find an even code \mathcal{C} for \mathcal{S} having minimum cost. In fact, we propose a solution for a slightly more general problem.

For $m \leq n$, denote $\mathcal{S}_m = \{s_1, \dots, s_m\}$. A *parity forest* F for \mathcal{S}_m is a set of q even trees and q odd trees, for some $q \in \{1, \dots, m\}$, such that the even parity leaves of even trees and odd parity leaves of odd trees correspond to the symbols of \mathcal{S}_m . Define the *cost* of F as the sum of the costs of its trees. Say that F is (m, q) -*optimal* when its cost is the minimum among all forests for \mathcal{S}_m having q even trees and q odd trees. Denote by $c(m, q)$ the cost of an (m, q) -optimal forest. Define the function

$$A_i = \begin{cases} \sum_{j=1}^i f_j, & \text{if } i > 0 \\ 0, & \text{otherwise} \end{cases}$$

By using this notation, the solution of our problem is a tree having as subtrees the ones of an $(n, 1)$ -optimal forest. Its cost is $c(n, 1) + A_n$.

The following theorem describes the computation of $c(m, q)$.

Theorem 1. *Let q, m be integers such that $1 \leq q \leq m \leq n$. Then:*

- (1) *If $m = q$ then $c(m, q) = 0$.*
- (2) *If $m > q$, then $c(m, q) = \min_{0 \leq i \leq q} \{c(m - i, 2q - i) + A_{m-i}\}$.*

Theorem 1 leads to a dynamic programming algorithm for determining $c(m, q)$, for all $1 \leq q \leq m \leq n$, as follows. Start by evaluating the function A_i for $1 \leq i \leq n$. The parameter m varies increasingly, $1 \leq m \leq n$. The first cost to be computed is $c(m, m)$, which is 0 by (1). For each such m , vary q decreasingly, $1 \leq q < m$, and for each such pair m, q , compute $c(m, q)$ by applying (2). The computation finishes when $c(n, 1)$ is calculated, since our target is to obtain $c(n, 1) + A_n$. There are $O(n^2)$ subproblems. The evaluation of each one is performed in constant time, when using (1), or in $O(n)$ time, when the evaluation is by (2). Consequently, the time complexity is $O(n^3)$. The space requirements are $O(n^2)$.

3 Heuristics

In this section we describe two heuristics to obtain even codes. Heuristics 1 is very simple and is based on a slight modification of the classical Huffman algorithm [1]. Heuristics 2 adds possible improvements to the previous one. As we shall see, those improvements allow to yield even codes very close to optimal ones.

3.1 Heuristics 1

Given n symbols with probabilities f_1, f_2, \dots, f_n , Heuristics 1 consists of two steps:

Step 1. *Run Huffman's algorithm in order to obtain a Huffman tree T_H for the n symbols.*

Step 2. *Convert T_H into an even tree T_{U_1} in the following way: for each odd leaf z corresponding to a symbol s_i , create two children z_L and z_R such that:*

- *the left child z_L is an error leaf;*
- *the right child z_R is the new encoding leaf corresponding to s_i . (We call z_R an augmented leaf.)*

Observe that the overall running time of Heuristics 1 is $O(n \log n)$, since it is dominated by Step 1. Step 2 can be easily done in $O(n)$ time.

3.2 Heuristics 2

Now we present three possible ways to improve the heuristics previously described. As we shall see, these improvements do not increase the running time in practice, and produce a qualitative increase of performance with respect to the cost of the generated code.

Improvement I. During Step 1 (execution of Huffman's algorithm), add the following test:

Among the candidate pairs of the partial trees to be merged at the beginning of a new iteration, break ties by giving preference to a pair of trees T_1 and T_2 such that T_1 is trivial and T_2 is not.

In other words, the idea is to avoid merging trivial trees as much as possible. The reason why this strategy is employed is explained below.

In T_H , there exist two sibling leaves for each merge operation of trivial trees occurring along the algorithm. Of course, one of the siblings is an odd leaf. When we force a trivial tree to be merged with a non-trivial one, we minimize the number of pairs of sibling leaves in T_H , and thus the number of those "assuredly odd" leaves. In many cases, this strategy decreases the additional cost needed to produce the even tree in Step 2.

Let us denote by T_{H_1} the Huffman tree obtained by Improvement I. It is worth remarking that this improvement does not affect the essence of Huffman's algorithm, since T_{H_1} is a plausible Huffman tree.

Moreover, it is possible to implement Improvement I in constant time by keeping two heaps H' and H'' during the execution of Huffman's algorithm, where the nodes of H' contain trivial trees and the nodes of H'' the remaining ones. At the beginning of the algorithm, H' contains n nodes and H'' is empty. When starting a new iteration, simply check whether the roots of H' and H'' form a candidate pair of partial trees to be merged; if so, the merging is performed.

Improvement II. Modify T_{H_1} by repeatedly applying the following operation in increasing depth order:

If there exist two nodes z', z'' at the same depth of T_{H_1} such that z' is an odd leaf and z'' is an even internal node, exchange the positions of z' and z'' .

Observe each single application of the above operation decreases the number of odd leaves in T_{H_1} by one unit. Each time we find k odd leaves and ℓ even internal nodes at some depth i , we perform $\min\{k, \ell\}$ operations and proceed to depth $i + 1$.

It is clear that the number of such operations is bounded by the number of leaves of T_{H_1} . Since a single operation can be done by modifying a constant number of pointers, the overall complexity of Improvement II is $O(n)$.

Denote by T_{H_2} the Huffman tree obtained by Improvement II. Again, the essence of Huffman's algorithm is not affected, since T_{H_2} is still a plausible Huffman tree.

Improvement III. Apply Step 2 on T_{H_2} . Let T be the even tree obtained. Then redistribute the symbols among the leaves of T as follows:

Whenever there exist two leaves z_i, z_j (of even parities) in T with depths $d_i \leq d_j$, representing symbols s_i, s_j with probabilities $f_i \leq f_j$, respectively, then exchange the symbols assigned to z_i and z_j .

Observe that each single re-assignment performed above reduces the cost of the resulting even tree by $(d_j - d_i)(f_j - f_i)$.

The entire process can be implemented in the following way: after applying Step 2, order the leaves z_1, z_2, \dots, z_n of T according to its respective depths d_1, d_2, \dots, d_n using bucket sort. Then reassign the leaves to symbols, such that leaf z_i with depth d_i is assigned to symbol s_{n-i+1} with probability f_{n-i+1} . (Recall that $f_1 \leq f_2 \leq \dots, f_n$.) By performing this procedure we restore distortions possibly introduced by Improvement II. The time required for this operation is therefore $O(n)$. Consequently, the overall time bound for Heuristics 2 is $O(n \log n)$, using $O(n)$ space.

Other possible improvements were considered and then discarded, as they did not bring experimental relevant advantages. One of those marginal improvements is to exchange an error leaf with a subtree whose root is deeper than that error leaf.

4 Bounds

In this section, we present an analytical upper bound for the cost of the even tree generated by Heuristics 2 with respect to the cost of the corresponding Huffman tree.

The terminology employed in this section is the following: given n symbols with probabilities f_1, f_2, \dots, f_n , T_H is the Huffman tree for these symbols; T_E is the corresponding optimal even tree; T_{U_1} is the even tree obtained by applying Heuristics 1; and T_{U_2} is the even tree obtained by applying Heuristics 2. Observe that

$$c(T_H) \leq c(T_E) \leq c(T_{U_2}) \leq c(T_{U_1}).$$

Lemma 1. $c(T_{U_2}) \leq c(T_H) + \frac{\sum_{i=1}^n f_i}{2}$.

Lemma 1 states that the maximum cost difference between an even tree obtained by Heuristics 2 and the corresponding Huffman tree is 0.5. Next theorem gives another bound that can be tighter for low entropy distributions, if the cost of the Huffman tree is lower than 3.

Theorem 2. $c(T_{U_2}) \leq \frac{7}{6}c(T_H)$, if $n > 4$.

The bound given by Theorem 2 cannot be improved. We show this fact by exhibiting an infinite family \mathcal{F} of Huffman trees where, given any $\epsilon > 0$, we can always find a tree $T_H \in \mathcal{F}$ such that, for the corresponding tree T_{U_2} obtained from Heuristics 2, we have $\frac{7}{6}c(T_H) - c(T_{U_2}) \leq \epsilon$.

This family is illustrated in Figure 3. Each tree of the family is characterized by two parameters n and q , $n \geq 4$ and $\frac{2}{5} \geq q > 0$. Each tree has n leaves and depth $n - 2$. Three leaves (l_{n-2} , l_{n-1} and l_n) have depth 2 and probability $\frac{1-q}{3}$. The remaining $n - 3$ leaves l_1, \dots, l_{n-3} are arranged in the tree as follows: one leaf (l_1) has depth $n - 2$ and probability $\frac{q}{2^{n-4}}$; $n - 4$ leaves (l_i , $2 \leq i \leq n - 3$) have depths $n - i$ and probabilities $\frac{q}{2^{n-i-2}}$, respectively. In this family of trees, there are only two odd parity leaves: l_1 and l_{n-1} .

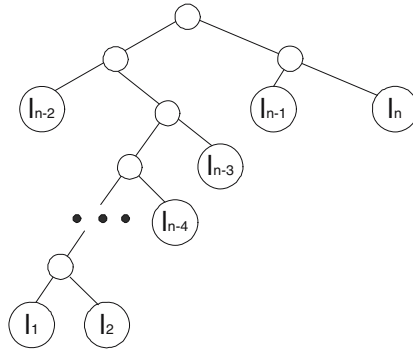


Fig. 3. Special family of Huffman trees

Theorem 3. Let T_H be a tree belonging to \mathcal{F} . Then T_H is a Huffman tree.

Theorem 4. Given $\epsilon > 0$, there exists $T_H \in \mathcal{F}$ such that the corresponding even tree T_{U_2} obtained from T_H by applying Heuristics 2 satisfies $\frac{7}{6}c(T_H) - c(T_{U_2}) \leq \epsilon$.

We found an exact upper bound for the cost difference between a Huffman tree and an even tree obtained by Heuristics 2. Clearly this is also an upper bound (possibly not exact) for the cost difference between an optimal even tree and the corresponding Huffman tree. Nevertheless, it remains open the determination of a tight upper bound for this difference. Experimental results suggest that it is quite lower than the bound given above.

5 Probabilistic Model for Error Detection

In this section we develop a probabilistic model to assess the error detection capability of an even tree. In the next section we present some experimental results to validate the model.

Let us suppose that one or more errors occurred in a coded message, that is, one or more bits are corrupted. After the first error, we can have a sequence of erroneously decoded symbols. The following model evaluates the probability of occurrence of a sequence of k erroneous decodings, and the overall probability to point out the error.

Let s_i be the first symbol in a coded message where the error was introduced and T the corresponding even tree used to encode the message. First we shall calculate the probability $P_p(T, k)$ of the next k symbols (including s_i) being erroneously decoded. $P_p(T, k)$ is the probability of error propagation through the next $k - 1$ symbols.

In this model, after the introduction of the first error, the remaining bit string to be decoded is assumed to be a random string. In terms of the decoding tree, this means that each deviation in the tree has the same probability, which is $1/2$. Consider that the first corrupted bit corresponds to the root of the tree. Then, the probability of leaf l_j being hit is $P_l(j) = 2^{-d_j}$, where d_j is the depth of l_j . We define two parameters for T : $Pt_c(T)$ and $Pt_e(T)$. $Pt_c(T)$ is the probability of any encoding leaf to be reached starting from the root, and its value is $Pt_c(T) = \sum_{j=1}^n 2^{-d_j}$. $Pt_e(T)$ is the corresponding probability, considering error leaves, and is given by $Pt_e(T) = \sum_{j=1}^{Ne(n)} 2^{-d_j}$, where $Ne(n)$ is the number of error leaves of T .

Clearly, $Pt_c(T) + Pt_e(T) = 1$.

The error propagation probability through k symbols is then $P_p(T, k) = Pt_c(T)^k = (1 - Pt_e(T))^k$, decreasing exponentially with k . Note that assuming the first error to occur exactly at the root only reduces this probability, since error leaves are placed at the two last levels of the tree. Thus, we will neglect the fact that the error can also occur at any node other than the root.

Now let us consider how to evaluate the error detection capability of T . Let b be the average number of symbols in a message. The error detection probability of T for messages with b symbols, $P_m(T, b)$, is the complement of the probability that the errors propagate until the end of the message.

If errors in the message are independent and one error occurs with probability q , then the probability that the i -th bit of the message is the first corrupted bit is given by

$$\sum_{k=1}^{b-i+1} q^k \frac{\binom{b-i}{k-1}}{\binom{b}{k}},$$

which can be approximated to q/b if q is small. That is, this probability is uniform along the b symbols in the message, and its value is $1/b$. Then $P_m(T, b)$ can be estimated as:

$$P_m(T, b) = 1 - \sum_{k=1}^b P_p(T, k)/b = 1 - (1 - Pt_e(T) - (1 - Pt_e(T))^{b+1})/(b \cdot Pt_e(T)).$$

We can refine this model, by considering the total number of errors introduced in the message and not only the first error. Once T detects any odd number of corrupted bits, and considering that half of error situations are related to an odd number of corrupted bits, we divide the probability of error propagation by 2. We can yet consider that, when decoding the last symbol, the process will not necessarily finish at a leaf. It might be the case that the last node hit is an internal one. If the process ends at an internal node, then the error is pointed out by the decoding process. As the number of internal nodes is not less than the number of encoding leaves, we will again divide the error propagation probability by 2.

This refinement leads to a new estimation:

$$P_m(T, b) = 1 - \sum_{k=1}^b (1 - Pt_e(T))^{k-1}/4b = 1 - (1 - (1 - Pt_e(T))^b)/(4b \cdot Pt_e(T)).$$

The above expression shows that, in order to estimate the error detection capability of an even tree T , the main parameter to consider is $Pt_e(T)$, related to error leaves. We now explore bounds for this parameter in the next two theorems.

Theorem 5. *If T is an optimal even tree for n symbols with uniform probabilities then $1/16 \leq Pt_e(T) \leq 1/4$.*

Theorem 6. *If T is an optimal even tree for n symbols with arbitrary probabilities then $2^{-n} \leq Pt_e(T) \leq 1/4$.*

The bounds of Theorems 5 and 6 are given for optimal even trees. If we consider even trees obtained by Heuristics 2, the only difference we have is the lower bound for uniform probabilities. Heuristics 2 always uses trees isomorphic to T_2 and T_3 in Figure 2. If T is an even tree obtained by Heuristics 2 with minimum $Pt_e(T)$, then T must be composed only by trees isomorphic to T_3 . This situation occurs for $n = 3 \cdot 2^k$, $k \geq 0$. Hence, T has 2^k error leaves, all having depth equal to $k + 3$, as it is presented in [12]. Thus, in this case,

$$Pt_e(T) = 2^k \cdot 2^{-(k+3)} = 2^{k-k-3} = 1/8.$$

6 Experimental Results

6.1 Costs of Even Tress

The experimental results of this subsection are summarized in Tables 2 to 4. The tables present the costs of the trees obtained by the algorithms described in Sections 2 and 3, for several values of n , obtained via a Pascal program running on a Pentium IV computer with 1.8 GHz and 256Mb RAM.

Table 2. Comparisons with uniform probabilities

n	$c(T_H)$ (A)	$c(T_E)$ (B)	$c(T_{U_1})$ (C)	$c(T_{U_2})$ (D)	$\frac{B-A}{A}$ %	$\frac{C-A}{A}$ %	$\frac{D-A}{A}$ %	n	$c(T_H)$ (A)	$c(T_E)$ (B)	$c(T_{U_1})$ (C)	$c(T_{U_2})$ (D)	$\frac{B-A}{A}$ %	$\frac{C-A}{A}$ %	$\frac{D-A}{A}$ %
64	6.00	6.50	6.50	6.50	8.3	8.3	8.3	576	9.22	9.67	9.72	9.67	4.8	5.4	4.8
128	7.00	7.50	7.50	7.50	7.1	7.1	7.1	640	9.40	9.80	9.90	9.80	4.3	5.3	4.3
192	7.67	8.00	8.17	8.00	4.4	6.5	4.4	704	9.55	9.91	10.05	9.91	3.8	5.2	3.8
256	8.00	8.50	8.50	8.50	6.3	6.3	6.3	768	9.67	10.00	10.17	10.00	3.5	5.2	3.5
320	8.40	8.80	8.90	8.80	4.8	6.0	4.8	832	9.77	10.15	10.27	10.15	3.9	5.1	3.9
384	8.67	9.00	9.17	9.00	3.9	5.8	3.9	896	9.86	10.29	10.36	10.29	4.4	5.1	4.4
448	8.86	9.29	9.36	9.29	4.8	5.7	4.8	960	9.93	10.40	10.43	10.40	4.7	5.0	4.7
512	9.00	9.50	9.50	9.50	5.6	5.6	5.6	1024	10.00	10.50	10.50	10.50	5.0	5.0	5.0

Table 3. Comparisons with arbitrary probabilities

n	$c(T_H)$ (A)	$c(T_E)$ (B)	$c(T_{U_1})$ (C)	$c(T_{U_2})$ (D)	$\frac{B-A}{A}$ %	$\frac{C-A}{A}$ %	$\frac{D-A}{A}$ %	n	$c(T_H)$ (A)	$c(T_E)$ (B)	$c(T_{U_1})$ (C)	$c(T_{U_2})$ (D)	$\frac{B-A}{A}$ %	$\frac{C-A}{A}$ %	$\frac{D-A}{A}$ %
64	5.82	5.97	6.38	6.03	2.7	9.7	3.7	576	8.92	9.03	9.39	9.09	1.2	5.3	2.0
128	6.76	6.88	7.28	6.92	1.9	7.8	2.4	640	9.10	9.22	9.60	9.31	1.4	5.6	2.3
192	7.33	7.52	7.84	7.59	2.7	7.0	3.6	704	9.20	9.33	9.69	9.41	1.4	5.4	2.4
256	7.75	7.87	8.25	7.90	1.6	6.5	2.0	768	9.33	9.53	9.83	9.58	2.1	5.3	2.7
320	8.08	8.20	8.57	8.30	1.5	6.1	2.7	832	9.44	9.63	9.93	9.67	2.0	5.2	2.4
384	8.34	8.54	8.85	8.60	2.5	6.1	3.1	896	9.57	9.73	10.08	9.77	1.7	5.3	2.1
448	8.52	8.68	9.04	8.73	1.9	6.0	2.4	960	9.64	9.78	10.14	9.82	1.5	5.2	1.9
512	8.72	8.84	9.22	8.87	1.3	5.8	1.7	1024	9.75	9.87	10.24	9.89	1.3	5.1	1.5

Table 4. Comparisons with arbitrary probabilities

n	$c(T_H)$ (A)	$c(T_{U_1})$ (B)	$c(T_{U_2})$ (C)	$\frac{B-A}{A}$ %	$\frac{C-A}{A}$ %	n	$c(T_H)$ (A)	$c(T_{U_1})$ (B)	$c(T_{U_2})$ (C)	$\frac{B-A}{A}$ %	$\frac{C-A}{A}$ %
1000	9.71	10.22	9.87	5.2	1.6	45000	15.21	15.71	15.45	3.3	1.6
2000	10.72	11.21	10.87	4.6	1.5	50000	15.36	15.86	15.61	3.3	1.7
3000	11.31	11.81	11.57	4.5	2.3	55000	15.49	15.99	15.71	3.2	1.4
4000	11.72	12.22	11.87	4.3	1.3	60000	15.62	16.12	15.80	3.2	1.2
5000	12.05	12.55	12.26	4.1	1.8	65000	15.74	16.24	15.89	3.2	1.0
10000	13.04	13.55	13.25	3.9	1.6	70000	15.85	16.35	16.01	3.2	1.0
15000	13.61	14.11	13.80	3.7	1.4	75000	15.95	16.45	16.14	3.1	1.2
20000	14.05	14.55	14.25	3.6	1.5	80000	16.04	16.54	16.25	3.1	1.3
25000	14.36	14.86	14.61	3.5	1.8	85000	16.13	16.63	16.35	3.1	1.4
30000	14.62	15.12	14.81	3.4	1.3	90000	16.21	16.71	16.45	3.1	1.5
35000	14.85	15.35	15.01	3.4	1.1	95000	16.29	16.79	16.54	3.1	1.5
40000	15.05	15.55	15.25	3.3	1.4	100000	16.36	16.86	16.61	3.1	1.6

In Tables 2 and 3 we compare $c(T_H)$, $c(T_E)$, $c(T_{U_1})$ and $c(T_{U_2})$, for $64 \leq n \leq 1024$. Table 2 refers to uniform probabilities, and Table 3 to arbitrary probabilities, obtained from the a standard routine for generating random numbers

in the range 1 to 10000. (We found no significant variations by modifying this range.) All the probabilities were further normalized so that the total sum is 1. In Table 4 we compare the two heuristics with Huffman's algorithm for n in the range 1000 to 100000.

The main result observed in Table 2 is that, for uniform probabilities, Heuristics 2 equals the Exact Algorithm, while Heuristics 1 does not. The main explanation for this fact is that, when the Huffman tree is a complete binary tree, improvements of Heuristics 2 apply very well. It can also be observed the small difference between Huffman's algorithm and the other methods, and the decrease of the relative costs when n increases. It can still be confirmed a theoretical result stated in [12]: the cost difference between the optimal even tree and the Huffman tree lays in the interval $[1/3, 1/2]$, being maximum (equal to $1/2$) when the number of lays in $n = 2^k$ for some integer k , and minimum (equal to $1/3$) when $n = 3 \cdot 2^k$.

Next, we examine the results presented in Table 3, for arbitrary probabilities. First, compare data from Tables 2 and 3. We can see that all data in columns 2 to 5 in Table 3 are smaller than the corresponding ones in Table 2. This is an expected behavior, since the cost has the tendency to decrease as long as probabilities get unbalanced. The relative difference between $c(T_E)$ and $c(T_H)$ decreases considerably as n increases. The same occurred for Heuristics 2, suggesting that it is also well applied for this situation, although it does not equal the optimal solution. However, for Heuristics 1, the behavior is quite different. Both the absolute value of the difference to $c(T_H)$ and the relative value increased. Thus, Heuristics 2 outperforms Heuristics 1 in this situation.

Table 4 illustrates the costs obtained for large values of n and arbitrary probabilities. The costs compared are $c(T_H)$, $c(T_{U_1})$ and $c(T_{U_2})$. The main results obtained from Table 3 are confirmed, that is, Heuristics 2 is far better than Heuristics 1. Moreover, the relative differences of costs from the two heuristics to Huffman's algorithm again decrease. Those differences become negligible for large values of n .

Finally, from the three tables, we can observe a gap between the upper theoretical bound presented in Section 4 and the experimental results, since all the relative differences between the costs of the even trees obtained by Heuristics 2 and the Huffman trees were at most 5%, for n large enough. It seems to be interesting to search for tighter bounds for this situation.

6.2 Error Detection

Here we consider experimental results on error detection. Table 5 illustrates theoretical and experimental values for $P_m(T, b)$, which is the probability of error detection of an even tree, considering the model of section 5.

In the first column we present a range of values for b , the number of symbols in a message, varying from 10 to 5000. In the second column, the theoretical values for $P_m(T, b)$ are shown, according to the model presented in Section 5, using $Pt_e(T) = 0.1248$, which is nearly the average value for this parameter. The third column contains experimental values for $P_m(T, b)$, the probability of error

Table 5. Probability of error detection

b	$P_m(T, b)$	
	Theoretical $(1 - (1 - 0.8752^b)/(0.4992 * b))$	Experimental
10	0.852500	0.903540
25	0.922732	0.987620
50	0.959987	0.990953
100	0.979968	0.994894
250	0.991987	0.998482
500	0.995994	0.999150
1000	0.997997	0.999595
2500	0.999199	0.999831
5000	0.999599	0.999922

detection, obtained via a simulation with an even tree T having $Pt_e(T) = 0.1248$. The simulation involved the generation of about 20,000,000 random errors consisting of one to twenty corrupted bits.

We can observe the fast growth of the error detection probability, as the length of the message increases. The error detection probability is fairly large for $b \geq 100$. It can also be verified that experimental values of error detection probabilities are greater than the predicted ones by the model, specially for lower values of b . This indicates that the error detection probability model can be enhanced. But the differences between theory and practice are quite low for $b > 100$.

We conclude by remarking that there clearly exists a trade-off between the cost of an even tree and its error detection capability. Within certain limits, as the value of one of those variables is improved, the quality of the other one gets poorer. It would be interesting to extend this work by examining codes with a better error detection capability, even sacrificing cost.

Acknowledgment. The authors would like to thank Artur Alves Pessoa for the insightful suggestions, which improved the exact dynamic programming algorithm of Section 2.

References

1. Huffman, D.A.: A Method for the Construction of Minimum Redundancy Codes. In: Proceedings of the IRE, vol. 40, pp. 1098–1101 (1951)
2. Faller, N.: An adaptive Method for Data Compression. In: Record of the 7th Asilomar Conference on Circuits, Systems and Computers, Naval Postgraduate School, Monterey, Ca, pp. 593–597 (1973)
3. Gallager, R.G.: Variations on a Theme by Huffman. IEEE Transactions on Information Theory 24, 668–674 (1978)
4. Knuth, D.E.: Dynamic Huffman Coding. Journal of Algorithms 6, 163–180 (1985)

5. Milidiú, R.L., Laber, E.S., Pessoa, A.A.: Improved Analysis of the FGK Algorithm. *Journal of Algorithms* 28, 195–211 (1999)
6. Schwartz, E.S.: An Optimum Encoding with Minimal Longest Code and Total Number of Digits. *Information and Control* 7, 37–44 (1964)
7. Turpin, A., Moffat, A.: Practical length-limited coding for large alphabets. *Computer J.* 38(5), 339–347 (1995)
8. Larmore, L.L., Hirshberg, D.S.: A fast algorithm for optimal length-limited Huffman codes. *J. ACM* 37(3), 464–473 (1990)
9. Milidiú, R.L., Laber, E.S.: The Warm-up Algorithm: A Lagrangean Construction of Length Restricted Huffman Codes. *SIAM Journal on Computing* 30(5), 1405–1426 (2000)
10. Milidiú, R.L., Laber, E.S.: Improved Bounds on the Inefficiency of Length Restricted Codes. *Algorithmica* 31(4), 513–529 (2001)
11. Hamming, R.W.: *Coding And Information Theory*. Prentice Hall, Englewood Cliffs (1980)
12. Pinto, P.E.D., Protti, F., Szwarcfiter, J.L.: Parity codes. *RAIRO - Inf. Theor. Appl.* 39, 263–278 (2005)
13. Pinto, P.E.D., Protti, F., Szwarcfiter, J.L.: A Huffman-based Error detecting Code. In: Ribeiro, C.C., Martins, S.L. (eds.) *WEA 2004. LNCS*, vol. 3059, pp. 446–457. Springer, Heidelberg (2004)

Terminal Coalgebras for Measure-Polynomial Functors

Christoph Schubert

Chair for Software Technology
Technische Universität Dortmund
christoph.schubert@tu-dortmund.de

Abstract. We use the Kolmogorov Consistency Theorem from Measure Theory to construct terminal coalgebras for a large class of functors on the category of measurable spaces. In particular, we construct terminal stochastic relations and terminal labelled Markov processes. We use these constructions to provide extended expressivity results for canonical interpretations of modal and temporal logics in these structures.

1 Introduction

Since the beginning of the study of coalgebra, the construction of *terminal* (or *final*) *coalgebras* was in particular focus. The pronounced interest in terminal coalgebras comes from the fact that in a certain sense they incorporate all possible behavior a given class of coalgebras is able to exhibit.

In fact, there is a canonical construction for terminal coalgebras for well-behaved functors using the so-called terminal sequence of the functor; see [14,1]. This construction has the advantage that it gives quite a concrete description of the terminal coalgebra.

It seems that the subprobability functor S on the category of measurable spaces is not well-behaved in the above sense. This has led to several involved constructions [11,12,13,8] for terminal coalgebras for this functor and its variants.

By doing an analysis of the terminal sequence construction we are able to show that S , although it does not preserve limits of arbitrary sequences, preserves its terminal sequence, hence a terminal coalgebra for S exists. Moreover, this result holds for so-called *measure-polynomial* endofunctors on the category of measurable spaces. These measure polynomial functors are a convenient replacement for the Kripke-polynomial functors on the category of sets and functions: these are constructed from the identity, constant functors, and the (finite) powerset functor by closing under composition, finite products and sums. In the same manner, the measure-polynomial functors are constructed from the identity, constant functor for Standard Borel spaces, the subprobability functor, and closed under composition, countable products and sums.

We have to restrict ourselves to Standard Borel spaces since they permit using the Kolmogorov Consistency Theorem, the main technical tool for our observations. Using this classical result, we are able to construct terminal stochastic

relations [4] (when morphisms of stochastic relations are based on arbitrary, not necessarily surjective, measurable functions), labelled Markov processes [11], and models for CSL, a stochastic variant of CTL, as used in probabilistic model checking [3].

We then proceed to apply the existence of terminal coalgebras based on Standard Borel spaces to obtain stronger expressivity results for coalgebraic modal logic.

2 Preliminaries

For undefined categorical terminology we refer to [2].

Measurable Spaces. Recall that a measurable space X consists of a set $|X|$ and a σ -algebra $\mathcal{B}X$ on X , that is: a family of subsets of $|X|$ which is closed under complementation, countable intersections, and countable unions. For each family \mathcal{A} of subsets of a set M there is a smallest σ -algebra on M containing \mathcal{A} , which we denote by $\sigma(\mathcal{A})$. A *measurable function* $X \rightarrow Y$ is given by a function $f : |X| \rightarrow |Y|$ such that $f^{-1}[B] \in \mathcal{B}X$ for all $B \in \mathcal{B}(Y)$. In case $\mathcal{B}Y = \sigma(\mathcal{A})$, measurability of f is guaranteed by $f^{-1}[A] \in \mathcal{B}X$ for all $A \in \mathcal{A}$. The category of measurable spaces with measurable functions as morphisms is denoted by **Meas**. Observe that we do not notationally distinguish between a **Meas**-morphisms and its underlying function. Often we will just write X in place of $|X|$.

Special Morphisms. Given a family $(Y_i)_I$ of measurable spaces and family $(f_i : A \rightarrow |Y_i|)_I$ of functions with common domain, we define the *initial σ -algebra* with respect this data to be $\mathcal{A} = \sigma(\bigcup_I f_i^{-1}[\mathcal{B}(Y_i)])$. It has the following property: a function $g : |X| \rightarrow A$ is measurable with respect to $\mathcal{B}X$ and \mathcal{A} if and only if all $f_i \cdot g : X \rightarrow Y_i$ are measurable. In case $I = \{*\}$, we have $\mathcal{A} = f_*^{-1}[\mathcal{B}(Y_*)]$.

Dually, we define the *final σ -algebra* for $(|X_i| \xrightarrow{g_i} B)_I$ by $\{E \subseteq B \mid \forall i \in I : g_i^{-1}[E] \in \mathcal{B}(X_i)\}$. It has the property that measurability of $h : B \rightarrow |Y|$ is guaranteed by the measurability of all $h \cdot g_i$.

Thus, **Meas** is complete and cocomplete: limits are constructed as follows: construct the limit $(L, (l_i))$ of the underlying diagram in **Set** and equip L with the initial σ -algebra with respect to the projections l_i , thus, with the σ -algebra generated by $\bigcup l_i^{-1}[\mathcal{B}X_i]$. Colimits are formed dually using final σ -algebras.

Recall that a monomorphism m in a category **C** is called *strong* provided that whenever we have $m \cdot f = g \cdot e$ with e an epimorphism, then there exists a (uniquely determined) d such that

$$\begin{array}{ccc}
 A & \xrightarrow{e} & B \\
 f \downarrow & \swarrow d & \downarrow g \\
 C & \xrightarrow{m} & D
 \end{array}$$

commutes. This is called the *diagonalization property*. The epimorphisms in **Meas** are precisely the surjective measurable functions and the strong monomorphisms are precisely the injective, initial functions. Observe that we may factor every **Meas**-morphism f as $f = m \cdot e$ with e surjective and m a strong monomorphism. We say that **Meas** has (Epi, StrongMono)-factorizations.

Lemma 1. *The class of strong monomorphisms in **Meas** is closed under limits and coproducts.*

Proof. Closure under limits holds in any complete category. Closure under coproducts follows from the characterization as initial monomorphisms. \square

Standard Borel Spaces. A measurable space is called a Standard Borel space provided its measurable sets arise as the Borel sets induced by a complete, separable metric. The full subcategory of **Meas** spanned by the Standard Borel spaces is denoted by **SB**. The following two results are essential:

Fact 1. **SB** is closed under countable coproducts and countable limits in **Meas**.

Proof. It is well-known that countable coproducts and products of Standard Borel spaces are Standard Borel; see [7, 12.B]. If Y is Standard Borel, then $\Delta_Y = \{ (y, y) \mid y \in Y \}$ is measurable in $Y \times Y$. Hence, for any pair $f, g : X \rightarrow Y$ of measurable functions, the set $E = \{ x \in X \mid f(x) = g(x) \} = \langle f, g \rangle^{-1}[\Delta_Y]$, with $\langle f, g \rangle : X \rightarrow Y \times Y$ the induced function, is measurable in X , hence it is a Standard Borel space when equipped with the initial σ -algebra with respect to the embedding $E \rightarrow X$; see [7, 13.4]. \square

Fact 2 ([5]). Every surjective **SB**-morphism is final. In particular, every bijective **SB**-morphism is an isomorphism, that is: its inverse is measurable. \square

Subprobability Measures. A *subprobability measure* on a measurable space X is a σ -additive function $\mathcal{B}X \rightarrow [0, 1]$. The set of all subprobability measures on X becomes a measurable space SX when equipped with the initial σ -algebra with respect to $(\text{ev}_A)_{A \in \mathcal{B}X}$ with $\text{ev}_A : SX \rightarrow [0, 1], \mu \mapsto \mu(A)$. This subprobability construction gives rise to a functor $S : \mathbf{Meas} \rightarrow \mathbf{Meas}$ by setting

$$Sf(\mu)(B) = \mu(f^{-1}[B])$$

for $f : X \rightarrow Y$ in **Meas**, $\mu \in SX, B \in \mathcal{B}Y$. Using the so-called Prohorov metric it can be shown that SX is Standard Borel provided X is Standard Borel; see, for instance, [4, Proposition 1.78].

Lemma 2 ([12]). *Let $\mathcal{A} \subseteq \mathcal{B}X$ be a closed under finite intersections with $\sigma(\mathcal{A}) = \mathcal{B}X$. Then a function $f : T \rightarrow SX$ is measurable provided each $\text{ev}_A \cdot f$ for $A \in \mathcal{A}$ is measurable.* \square

Lemma 3. *If $f : X \rightarrow Y$ is initial in **Meas**, then Sf is initial and injective. In particular, S preserves strong monomorphisms.*

Proof. Assume that $Sf(\mu) = Sf(\nu)$ holds and take any $A \in \mathcal{B}(X)$. By initiality of f , there exists $B \in \mathcal{B}(Y)$ with $A = f^{-1}[B]$. Thus, $\mu(A) = \mu(f^{-1}[B]) = Sf(\mu)(B) = Sf(\nu)(B) = \nu(f^{-1}[B]) = \nu(A)$.

Observe that $\{ev_A : SX \rightarrow [0, 1] \mid A \in \mathcal{B}(X)\} = \{ev_B \cdot Sf : SX \rightarrow [0, 1] \mid B \in \mathcal{B}(Y)\}$ holds. Indeed, $ev_B \cdot Sf = ev_{f^{-1}[B]}$ so the claim follows from initiality of f . Hence with (ev_A) also $(ev_B \cdot Sf)$ is an initial source, and so Sf is initial; see, e.g., [2, 10.45]. □

Fact 3 ([4, Proposition 1.101]). If $f : X \rightarrow Y$ is a surjective **SB**-morphism, then also $Sf : SX \rightarrow SY$ is surjective. □

The proof of this fact is non-trivial and involves quite a bit of structure theory of Standard Borel spaces. Our main result hinges on the following result, known as the *Kolmogorov Consistency Theorem*:

Theorem 1. *Let*

$$A_0 \xleftarrow{f_0} A_1 \xleftarrow{f_1} A_2 \dots A_n \xleftarrow{f_n} A_{n+1} \dots$$

be a sequence of Standard Borel spaces and surjective measurable functions. If, for each $i \in \omega$, μ_i is a subprobability measure on A_i such that $Sf_i(\mu_{i+1}) = \mu_i$ holds, then there is a unique subprobability measure μ on the limit L of the above sequence such that $\mu_i = Sp_i(\mu)$, with p_i the projection. □

For a proof see, e.g., [6]. In Theorem 1 the restriction to Standard Borel spaces as well as to surjective functions is necessary: for counterexamples, see [13]. Let us write \mathcal{C} for the class of ω^{op} -chains of surjective **SB**-morphisms. A \mathcal{C} -limit is a limit of a diagram in \mathcal{C} .

Corollary 1. *The subprobability functor S preserves \mathcal{C} -limits.*

Proof. The Kolmogorov Consistency Theorem allows us to construct a bijection between the limit of the S -image of a chain (A_i) in \mathcal{C} and the S -image of the limit of the chain. Making essential use of Lemma 2, one shows in a straightforward but slightly involved calculation that this bijection is measurable. Hence it is a **Meas**-isomorphism by Fact 2. □

Arbitrary limits of ω^{op} -chains in **Meas** are not preserved by S ; see [13] for a discussion. As it turns out, S preserves limits of just the right type of ω^{op} -chains for our purposes.

3 Coalgebras

Fix a category \mathbf{C} and a functor $T : \mathbf{C} \rightarrow \mathbf{C}$. A T -coalgebra $\mathbb{A} = (A, d)$ is given by a \mathbf{C} -object A and a \mathbf{C} -morphism $d : A \rightarrow TA$, called the *dynamics*. A morphism of coalgebras $(A, d) \rightarrow (A', d')$ is given by a \mathbf{C} -morphism $f : A \rightarrow A'$ such that

$$\begin{array}{ccc}
 A & \xrightarrow{f} & A' \\
 d \downarrow & & \downarrow d' \\
 TA & \xrightarrow{Tf} & TA'
 \end{array}$$

commutes. This leads to the category $\mathbf{Coalg} T$ of T -coalgebras and morphisms.

Examples 1. We focus on $\mathbf{C} = \mathbf{Meas}$.

1. $\mathbf{Coalg} S$ is the category of stochastic relations. Observe that, in contrast to [4], we do not require morphisms of stochastic relations to be surjective.
2. From $T : X \mapsto (SX)^{\text{Act}}$ with Act a countable set we obtain the category of labelled Markov processes [11] as $\mathbf{Coalg} T$.
3. From $T : X \mapsto S((\mathbb{R}_+ \times X)^\omega)$ we obtain coalgebras which serve as a semantics for CSL [3]; see [4].

Lemma 4. *If \mathbf{C} has (Epi, StrongMono)-factorizations and T preserves strong monomorphisms, then $\mathbf{Coalg} T$ has (Epi, StrongMono)-factorizations.*

Proof. Take $f : (A, d) \rightarrow (A', d')$ and factor f in \mathbf{C} as $f = m \cdot e$ with m a strong monomorphism and e surjective. Since also Tm is a strong monomorphism, we obtain h as below:

$$\begin{array}{ccc}
 A & \xrightarrow{e} & B \\
 d \downarrow & \swarrow h & \downarrow m \\
 TA & \xrightarrow{h} & A' \\
 Te \downarrow & \swarrow & \downarrow d' \\
 TB & \xrightarrow{Tm} & TA'
 \end{array}$$

It remains to show that $e : (A, d) \rightarrow (B, h)$ is an epimorphism in $\mathbf{Coalg} T$ and that $m : (B, h) \rightarrow (A', d')$ is a strong monomorphisms in $\mathbf{Coalg} T$. This is straightforward, see [11, 4.6 and 4.11]. \square

Making use of Lemma 3, we see that we may factor every morphism f of stochastic relations as a surjection followed by an initial injection.

Terminal Coalgebras via Terminal Sequences. A *terminal object* in a category \mathbf{C} is an object 1 such that for any object C there exists a unique morphism $!_C : C \rightarrow 1$. Fix a category \mathbf{C} with terminal object 1 and a functor $T : \mathbf{C} \rightarrow \mathbf{C}$. The *terminal sequence* of T is the following ω^{op} -sequence:

$$1 \xleftarrow{!} T1 \xleftarrow{T!} T^2 1 \xleftarrow{T^2!} T^3 1 \xleftarrow{T^3!} T^4 1 \dots$$

where $! : T1 \rightarrow 1$ is the unique morphism. We recall [11, 3.18 and 3.12]:

Proposition 1. *If the terminal sequence of T has a limit in \mathbf{C} which is preserved by T , then $\mathbf{Coalg} T$ has a terminal object.* \square

Proposition 2. *Let \mathbf{C} be a category with binary products, C a \mathbf{C} -object and $T : \mathbf{C} \rightarrow \mathbf{C}$ a functor. The following statements are equivalent for a morphism $d : A \rightarrow C \times TA$ with components $d_1 : A \rightarrow C$ and $d_2 : A \rightarrow TA$:*

1. (A, d_2) is a cofree T -coalgebra over C with couniversal arrow d_1 .
2. (A, d) is a terminal $C \times T(-)$ -coalgebra. □

4 Terminal Coalgebras for Measurable Spaces

Let us write \mathcal{F} for the class of endofunctors T on **Meas** which

- preserve surjective **SB**-morphisms in the sense that whenever $f : A \rightarrow B$ is surjective with A, B in **SB**, then also TA, TB are in **SB** and Tf is surjective;
- preserve \mathcal{C} -limits.

In particular, every $T \in \mathcal{F}$ preserves \mathcal{C} , and thus restricts to the subcategory **SB**. For every $T \in \mathcal{F}$, we write $T_{\mathbf{SB}}$ for its restriction **SB** \rightarrow **SB**.

Proposition 3. *\mathcal{F} is closed under composition, countable coproducts, and countable limits.*

Fact 2 allows us to reduce the statement of the proposition to well-known closure properties of ω^{op} -sequence limit preserving endofunctors on **Set**; see, e.g., [14].

Proof. Clearly, \mathcal{F} is closed under composition. To show closure under countable coproducts, it suffices to show that the class of ω^{op} -sequence preserving functors on **SB** is closed under countable coproducts. Limits of ω^{op} -sequences commute with coproducts in **Set**, hence the underlying function of the unique connecting morphism is bijective, thus the result holds for Standard Borel spaces by Fact 2.

To show that \mathcal{F} is closed under countable limits, it suffices to recall that limits commute with limits in every category. □

Definition 1. *The class of measure-polynomial functors is the smallest class of endofunctors on **Meas** which contains the identity, the subprobability-functor S , the constant functor C_A for each $A \in \mathbf{SB}$, and is closed under composition, countable coproducts, and countable products.*

Examples of measure polynomial functors include the functors $A + (-)$ and $A \times (-)$ for each Standard Borel space A , as well as the functors presented in Examples 1. We remark that our notion of measure polynomial functor differs from the one in [8]; see Sect. 6 for a discussion.

Lemma 5. *Every measure-polynomial functor on **Meas** preserves strong monomorphisms.*

Proof. Obviously, every constant functor preserves strong monomorphisms, and the subprobability functor preserves them by Lemma 3. The claim follows from Lemma 1. □

Theorem 2. *Every measure-polynomial endofunctor on \mathbf{Meas} is in \mathcal{F} .*

Proof. We first show that every measure-polynomial functor preserves surjective **SB**-morphisms. Preservation of **SB** is obvious by Fact [1](#) and the fact that SX is Standard Borel provided X is. Preservation of surjectivity is obvious by Fact [3](#) and the fact that the class of surjectivity-preserving functors on \mathbf{Meas} is closed under countable limits and colimits.

Clearly, \mathcal{F} contains the identity and, by Corollary [1](#), the subprobability functor S as well as $A \times (-)$ and $A + (-)$ for each Standard Borel space A . Clearly, the latter functors preserve \mathcal{C} . To show that they also preserve limits of ω^{op} -sequences, it suffices to show that this holds for the corresponding functors on **Set** and invoke Fact [2](#). Finally, the closure properties outlined in Proposition [3](#) show that every measure polynomial functor lies in \mathcal{F} . \square

Theorem 3. *Every functor in \mathcal{F} has a terminal coalgebra whose carrier space is Standard Borel.*

Proof. Let $T \in \mathcal{F}$. We need to distinguish two cases. First, suppose that $T1 \neq \emptyset$, that is, $! : T1 \rightarrow 1$ is surjective. Hence, the terminal sequence of T is in \mathcal{C} , thus its limit is preserved by T , and hence a terminal coalgebra exists by Proposition [1](#). In case $T1 = \emptyset$ holds, we have $TX = \emptyset$ of all $X \in \mathbf{Meas}$. Indeed, indeed, $! : X \rightarrow 1$ gives rise to $T! : TX \rightarrow T1 = \emptyset$, and $TX = \emptyset$ follows. Thus, $(\emptyset, 1_\emptyset)$ is the unique T -coalgebra, thus necessarily terminal. \square

We call the constant endofunctor on \mathbf{Meas} with value \emptyset *trivial*, and every other endofunctor *non-trivial*.

Corollary 2. *Let T in \mathcal{F} be non-trivial. For each Standard Borel space A there exists a cofree T -coalgebra over A . In particular, the obvious forgetful functor $\mathbf{Coalg} T_{\mathbf{SB}} \rightarrow \mathbf{SB}$ has a right adjoint.* \square

Corollary 3. *Every measure-polynomial endofunctor on \mathbf{Meas} has a terminal coalgebra. For every non-trivial such endofunctor T there exists a cofree coalgebra over each Standard Borel space. In particular, the forgetful functor $\mathbf{Coalg} T_{\mathbf{SB}} \rightarrow \mathbf{SB}$ has a right adjoint.* \square

5 Applications to Modal Logic

We will now use properties of the final coalgebras to extend some results on expressivity of modal logics. Fix a countable set Act (of “actions”) and define a logic \mathcal{L} to consist of the following formulas

$$\varphi ::= \top \mid \varphi \wedge \varphi \mid \langle a \rangle_q \varphi$$

with $a \in \text{Act}$ and $q \in \mathbb{Q} \cap [0, 1]$. Let $T : \mathbf{Meas} \rightarrow \mathbf{Meas}$ be the functor given by $X \mapsto (SX)^{\text{Act}}$. Thus, T -coalgebras are precisely labelled Markov transition systems. We can interpret \mathcal{L} in any T -coalgebra $\mathbb{X} = (X, d)$ as follows:

$$\begin{aligned} \llbracket \top \rrbracket_{\mathbb{X}} &= X & \llbracket \varphi_1 \wedge \varphi_2 \rrbracket_{\mathbb{X}} &= \llbracket \varphi_1 \rrbracket_{\mathbb{X}} \cap \llbracket \varphi_2 \rrbracket_{\mathbb{X}} \\ \llbracket \langle a \rangle_q \varphi \rrbracket_{\mathbb{X}} &= \{ x \mid d(x)(a)(\llbracket \varphi \rrbracket_{\mathbb{X}}) \geq q \} \end{aligned}$$

Let us call states x and y of coalgebras \mathbb{X} and \mathbb{Y} , resp., *logically equivalent* if they satisfy exactly the same formulas, and behaviorally equivalent if there exists a coalgebra \mathbb{W} and morphism $\mathbb{X} \xrightarrow{f} \mathbb{W} \xleftarrow{g} \mathbb{Y}$ with $f(x) = g(y)$. In [9] the following result was established:

Fact 4. Let \mathbb{X} and \mathbb{Y} be T -coalgebras with X, Y Standard Borel. States x and y of \mathbb{X} and \mathbb{Y} , resp., are logically equivalent if and only if they are behaviorally equivalent. \square

In fact, in [9] the above result was established for a large class of functors T on **SB**. We will now alleviate the restriction to Standard Borel spaces and show that the conceptual equality

$$\text{logical equivalence} = \text{behavioral equivalence}$$

holds not only for labelled Markov transition systems based on Standard Borel spaces but for all labelled Markov transition systems. Note the following fact:

Lemma 6. Let $f : \mathbb{X} \rightarrow \mathbb{Y}$ be a **Coalg** T -morphism. Then $[\![\varphi]\!]_{\mathbb{X}} = f^{-1}[\![\varphi]\!]_{\mathbb{Y}}$ holds for all $\varphi \in \mathcal{L}$. \square

The logical equivalence relation on a final coalgebra is degenerate:

Lemma 7. Let \mathbb{F} be a terminal T -coalgebra. If states x and y of \mathbb{F} are logically equivalent, then they are equal.

Proof. Recall that \mathbb{F} is based on a Standard Borel space. By Fact 4 there exists \mathbb{W} in **Coalg** T and $f, g : \mathbb{F} \rightarrow \mathbb{W}$ with $f(x) = g(y)$. Hence, we have $!_{\mathbb{W}} \cdot f(x) = !_{\mathbb{W}} \cdot g(y)$ for the unique $!_{\mathbb{W}} : \mathbb{W} \rightarrow \mathbb{F}$. By terminality, we have $!_{\mathbb{W}} \cdot f = 1_{\mathbb{F}} = !_{\mathbb{W}} \cdot g$, hence $x = y$. \square

Theorem 4. States x and y of T -coalgebras \mathbb{X} and \mathbb{Y} , resp., are logically equivalent if, and only if, they are behaviorally equivalent.

Proof. Suppose x and y to be logically equivalent. $!_{\mathbb{X}}(x)$ and $!_{\mathbb{Y}}(y)$ are logically equivalent by Lemma 6 and hence equal by Lemma 7. The other implication is just Lemma 6. \square

Obviously, the proof of Theorem 4 makes no use of special properties of the logic \mathcal{L} nor the functor T besides the validity of Fact 4, Lemma 6, and the existence of a terminal T -coalgebra based on a Standard Borel space. Hence, the extended expressivity Theorem 4 is valid for any functor T on **Meas** which lies in \mathcal{F} and whose restriction $T_{\mathbf{SB}}$ satisfies the assumptions from [9].

Corollary 4. States x and x' of some T -coalgebra \mathbb{X} are logically equivalent if, and only if, $!_{\mathbb{X}}(x) = !_{\mathbb{X}}(x')$ holds. Hence, the kernel relation of $!_{\mathbb{X}}$ and the logical equivalence relation coincide. \square

Global Properties. We call coalgebras \mathbb{X} and \mathbb{Y} (*globally*) *logically equivalent* provided we may find, for each state x in of \mathbb{X} , a state y of \mathbb{Y} which satisfies exactly the same formulas, and vice versa. We call \mathbb{X} and \mathbb{Y} (*globally*) *behaviorally equivalent* if there exists a coalgebra \mathbb{W} and surjective morphisms $\mathbb{X} \xrightarrow{f} \mathbb{W} \xleftarrow{g} \mathbb{Y}$.

Theorem 5. *The following statements are equivalent for coalgebras \mathbb{X} and \mathbb{Y} based on arbitrary measurable spaces:*

1. \mathbb{X} and \mathbb{Y} are logically equivalent;
2. the images of $!_{\mathbb{X}}$ and $!_{\mathbb{Y}}$ coincide;
3. \mathbb{X} and \mathbb{Y} are behaviorally equivalent.

Proof. Assume \mathbb{X} and \mathbb{Y} to be logically equivalent. If $l = !_{\mathbb{X}}(x)$, then we may find y in \mathbb{Y} which satisfies exactly the same formulas. Hence, $l = !_{\mathbb{X}}(x) = !_{\mathbb{Y}}(y)$. The other inclusion is shown analogously. Now assume that the images of $!_{\mathbb{X}}$ and $!_{\mathbb{Y}}$ coincide. We obtain the following factorizations:

$$\begin{array}{ccccc}
 \mathbb{X} & \xrightarrow{e} & \text{Im } !_{\mathbb{X}} = \text{Im } !_{\mathbb{Y}} & \xleftarrow{e'} & \mathbb{Y} \\
 & \searrow & \downarrow i & \swarrow & \\
 & & \mathbb{F} & & \\
 & \searrow !_{\mathbb{X}} & & \swarrow !_{\mathbb{Y}} & \\
 & & & &
 \end{array}$$

with i the initial inclusion and e, e' surjective by Lemma 4

The implication (3) \Rightarrow (1) is obvious. □

6 Related Work

In [11], a terminal coalgebra for labelled Markov transition systems is constructed using domain-theoretic techniques. The connection between Standard Borel spaces and so-called ω -coherent domains is intriguing. It is not clear at the moment whether this construction can be adapted to deal with general measure-polynomial functors.

Moss and Viglizzo constructed in [8] terminal coalgebras for endofunctors on **Meas** of the following kind:

$$T ::= \text{Id} \mid \Delta T \mid C_X \mid T_1 + T_2 \mid T_1 \times T_2$$

where Id is the identity functor on **Meas**, Δ is the subfunctor of S given by the subprobabilities μ on X with $\mu(X) = 1$, and C_X is the constant functor for an *arbitrary* measurable space X . We denote the class of functors defined above by \mathcal{R} . They are called measure polynomial in [8]. While \mathcal{R} contains constant functor C_X for arbitrary measurable spaces X , it is not closed under countable products nor countable coproducts. In particular, labelled Markov processes for a countable set of actions do not arise from any functor in \mathcal{R} .

The construction in [8] uses so-called *satisfied theories* and is more involved than the classical construction based on terminal sequences. One might argue that the limit of the terminal sequence, which can be constructed as a subset of a countable product, is a quite simple object amenable to direct computation. This seems not to be the case for the space of satisfied theories.

The following strong result follows from [8] by using Proposition 2:

Proposition 4. *Coalg $T \rightarrow \text{Meas}$ has a right adjoint for every T in \mathcal{R} .* □

7 Conclusion

We have constructed terminal coalgebras for measure-polynomial functors on the category of measurable spaces. These functors form the smallest class of functors which contains the identity, the subprobability functor, constant functors for Standard Borel spaces, and is closed under countable coproducts and products. We have used this construction to extend expressivity results for modal logics previously obtained for Standard Borel spaces to general measurable spaces.

Acknowledgements. The author wants to thank Ernst-Erich Doberkat for numerous technical discussions and the anonymous referees for a number of suggestions which helped to improve this article.

References

1. Adamék, J.: Introduction to coalgebra. *Theory and Applications of Categories* 14(8), 157–199 (2005)
2. Adamek, J., Herrlich, H., Strecker, G.: *Abstract and Concrete Categories*. Wiley Interscience, Hoboken (1991), <http://www.tac.mta.ca/tac/reprints/articles/17/tr17.pdf>
3. Baier, C., Haverkort, B., Hermanns, H., Katoen, J.-P.: Model-checking algorithms for continuous time Markov chains. *IEEE Trans. Eng. 29(6)*, 524–541 (2003)
4. Doberkat, E.-E.: *Stochastic Relations. Foundations for Markov Transition Systems*. Chapman & Hall/CRC Press, Boca Raton (2007)
5. Doberkat, E.-E., Schubert, C.: Coalgebraic logic for stochastic right coalgebras. *Ann. Pure Appl. Logic* (2009), doi:10.1016/j.apal.2008/06/018
6. Doob, J.L.: *Measure theory*. Graduate Texts in Mathematics, vol. 143. Springer, New York (1994)
7. Kechris, A.S.: *Classical Descriptive Set Theory*. Graduate Texts in Mathematics. Springer, Heidelberg (1994)
8. Moss, L., Viglizzo, I.: Final coalgebras for functors on measurable spaces. *Information and Computation* 204, 610–636 (2006)
9. Schubert, C.: Coalgebraic logic over analytic spaces. Technical Report 170, Chair for Software Technology, Technische Universität Dortmund (January 2008)
10. Schubert, C.: Final coalgebras for measure-polynomial functors. Technical Report 175, Chair for Software Technology, Technische Universität Dortmund (December 2008)
11. van Breugel, F., Mislove, M., Ouaknine, J., Worrell, J.: Domain theory, testing and simulation for labelled Markov processes. *Theoretical Computer Science* 333, 171–197 (2005)
12. Viglizzo, I.D.: Coalgebras on measurable spaces. PhD thesis, Indiana University (2005)
13. Viglizzo, I.D.: Final sequences and final coalgebras for measurable spaces. In: Fiadeiro, J.L., Harman, N.A., Roggenbach, M., Rutten, J. (eds.) *CALCO 2005*. LNCS, vol. 3629, pp. 395–407. Springer, Heidelberg (2005)
14. Worrell, J.: Terminal coalgebras for accessible endofunctors. *Electronic Notes in Theoretical Computer Science* 19, 39–54 (1999)

High Minimal Pairs in the Enumeration Degrees

Andrea Sorbi¹, Guohua Wu^{2,*}, and Yue Yang^{3,**}

¹ University of Siena, 53100 Siena, Italy

sorbi@unisi.it

<http://www.dsmi.unisi.it/~sorbi/>

² Nanyang Technological University, Singapore

guohua@ntu.edu.sg

<http://www3.ntu.edu.sg/home/guohua/>

³ National University of Singapore, Singapore

matyangy@nus.edu.sg

<http://ww1.math.math.nus.edu.sg/~matyangy/>

Abstract. The natural embedding of the Turing degrees into the enumeration degrees preserves the jump operation, and maps isomorphically the computably enumerable Turing degrees onto the Π_1^0 enumeration degrees. The embedding does not preserve minimal pairs, though, unless one of the two sides is low. In particular it is known that there exist high minimal pairs of c.e. Turing degrees that do not embed to minimal pairs of e-degrees. We show however that high minimal pairs of Π_1^0 e-degrees do exist.

1 Introduction

In the structure of the computably enumerable Turing degrees, the existence of minimal pairs of c.e. Turing degrees was established independently by Lachlan [2] and Yates [6] in 1966. In the same 1966 paper [2], Lachlan also showed that the c.e. sets whose degrees form the minimal pair can be maximal, which implies the existence of high minimal pairs of c.e. Turing degrees. Using the tree method, one can easily combine the high strategy with the minimal pair strategy to produce directly a high minimal pair. Individually, the highness strategy and the minimal pair strategy are Π_2 -strategies, and they can be implemented by infinite injury arguments. Combining these two kinds of strategies does not require much extra effort, and it still results in an infinite injury argument. One would expect the same phenomenon in the enumeration degrees (for short: e-degrees). But it is not so. Although constructing a minimal pair of Σ_2^0 e-degrees requires only a Sacks' type of finite injury argument and constructing a high e-degree requires infinite injury, the combination of these two kinds of strategies requires extra

* Wu is partially supported by NTU grant RG58/06, M52110023.

** Yang is partially supported by NUS grant R 146-000-114-112; part of this research was carried out by Yang, while he was an INDAM-GNSAGA Visiting Professor at Siena.

effort as demonstrated in this paper — we need a $\mathbf{0}'''$ -argument to figure out how the minimal pair requirements are satisfied.

A natural starting point is to explore the embedding ι from the Turing degrees into the enumeration degrees; after all, ι also preserves the jump, so high Turing degrees are mapped to high e-degrees. Unfortunately, ι does not necessarily preserve the infimum. In fact, McEvoy in his thesis [3] (see also McEvoy and Cooper [4]) produced a high minimal pair of c.e. Turing degrees which, under the embedding ι , does not form a minimal pair in the e-degrees. This dashed the hope of directly translating results concerning high minimal pairs in the Turing degrees to e-degrees by ι .

In his investigation, McEvoy isolated a notion on Σ_2^0 sets A and B called “meshing”; and he proved that if A and B mesh then they will not form a minimal pair in the e-degrees. Sacrificing some accuracy, the idea of meshing can be crudely rephrased as follows: some Σ_2^0 -approximations of A and B can synchronize their changes so that for some e-operators Θ and Ψ ,

$$\exists x [\exists s \forall t > s (x \in \Theta^A[t] \vee x \in \Psi^B[t]) \wedge x \notin \Theta^A \cup \Psi^B].$$

As argued by McEvoy in [3], one could use this x to “diagonalize against” some minimal pair strategy in the e-degrees. Hence to build a minimal pair in the Σ_2^0 e-degrees, we must break this meshing. In other words, we have to prevent a certain x from leaving Θ^A and Ψ^B in a Π_2^0 -way, yet at any moment the same x appears to be in either Θ^A or Ψ^B .

The easiest way to do this is to make one of them, say A , low, thus no elements are able to leave Θ^A in a Π_2^0 -way. One can build an e-minimal pair with A low and B high without too much trouble. McEvoy and Cooper even pointed out in [4] that for any set A with low degree, there is a set B such that the e-degrees of A and B form a minimal pair.

For our purpose, to make both A and B high, we need new strategies, where breaking meshing becomes the main issue of the construction.

2 Building a High Minimal Pair

We state and prove the theorem.

Theorem 1. *There are Π_1^0 -sets A and B , both of high enumeration degrees, such that for all pairs of enumeration operators Θ and Ψ , if $\Theta^A = \Psi^B$ then Θ^A is c.e..*

2.1 Requirements and Strategies

We begin with describing each strategy in isolation. In the next subsection, we revise the strategies to overcome their conflicts.

Fix a computable enumeration of all pairs of e-operators Θ and Ψ . The minimal pair requirements are:

- $M_{\Theta, \Psi}$: If $\Theta^A = \Psi^B$ then there is a c.e. set W such that $\Theta^A = W$.

The strategy μ for a minimal pair requirement M , say $M = M_{\Theta, \Psi}$, is very similar to the one in the Turing degrees. We build a c.e. set W at μ . When we see an element $v \in \Theta^A \cap \Psi^B$ we enumerate v into W and guarantee that for any future stage t , v is staying in either $\Theta^A[t]$ or $\Psi^B[t]$. Whenever we extract some elements out of A to cause v out of Θ^A , we will hold the B -side to keep v in Ψ^B . If v never returns to Θ^A , then we have an easy disagreement at v between Θ^A and Ψ^B . This is a Σ_2 -outcome which is quite similar to the one in the Turing degrees. The net effect is to have a finite restraint on B forever. The more interesting possibility is that v returns to Θ^A , and we see $W \subseteq \Theta^A \cap \Psi^B$ again. This cycle can repeat forever, in which case we have a Π_2^0 -outcome ∞ . A similar argument applies if we extract some elements out of B to cause v out of Ψ^B .

We say that s is a μ -expansionary stage if, roughly, $W[s] \subseteq \Theta^A \cap \Psi^B[s]$. Assume that μ is able to achieve that for all elements x and for all stages s , if x in $W[s]$ then $x \in \Theta^A[s] \vee x \in \Psi^B[s]$. Under this assumption, if the μ -strategy sees only finitely many μ -expansionary stages, then M is satisfied, since $\Theta^A \neq \Psi^B$. If the μ -strategy sees infinitely many μ -expansionary stages, we can conclude that W appears to be correct at infinitely many stages. However, we have no idea if $W = \Theta^A$ or not, as in the discussion of meshing. We will come back to the satisfaction of M in the next subsection.

To satisfy the highness requirements, we use the highness strategy, due to [11], as implemented in [5]. Recall, [5], that a sufficient condition for a set A to be of high enumeration degree, is that for some Σ_2^0 -approximation A_s of A , the function $C_A(x)$ is total and eventually dominates all total computable functions $\varphi_e(x)$, where

$$C_A(x) = \mu s > x[A_s \upharpoonright x \subseteq A].$$

Thus we aim at building Π_1^0 -approximations to A and B , satisfying the requirements

- P_e^A : If $\varphi_e(x)$ is a total computable function then $C_A(x)$ eventually dominates $\varphi_e(x)$.
- P_e^B : If $\varphi_e(x)$ is a total computable function then $C_B(x)$ eventually dominates $\varphi_e(x)$.

Let α be a strategy working for P_e^A . In isolation, α has a fixed infinite decidable set $E_\alpha = \{x_0 < x_1 < \dots\}$ for its use. We assume that the predicate “ $x \in E_\sigma$ ” is decidable, and $E_\sigma \cap E_\tau = \emptyset$ for all $\sigma \neq \tau$. We start with $E_\alpha \subseteq A$. The strategy waits for a stage s at which $\varphi_e(y) \downarrow [s]$ for all y such that $y < x_1$. Here we assume the convention that if $\varphi_e(y) \downarrow = z[s]$ then $y, z, e < s$. If no such s exists then φ_e is not total. Suppose we see such an s , then α extracts x_0 out of A forever at the first stage $t \geq s$ when we visit α again. Thus for all y with $x_0 \leq y < x_1$, $C_A(y) \geq t > s > \varphi_e(y)$. Then we work on x_1 in the same way, and so on. Thus if α has a Π_2^0 -outcome, i.e., φ_e is total, then α would extract the decidable set E_α out of A , and for any $y \geq x_0$, $C_A(y) > \varphi_e(y)$.

The strategy β to make B high is symmetric.

2.2 Coordinations of Strategies

We now put things together, discuss possible conflicts and modify our strategies to overcome the conflicts.

From simple to complex, we discuss the following cases. Let us ignore the Σ_2 -outcomes as their impact on the construction is finite.

Case 1. A high strategy α having higher priority than a minimal pair strategy μ . For example, α is making A high, and $\alpha \hat{\infty} \subset \mu$. In this case, μ knows the decidable set E_α that α is going to extract. μ will only use μ -believable computations (i.e. computations that do not use any element in E_α) when it decides which elements to enumerate in W . In conclusion, the difficulty is solved in the same way as in the c.e. Turing degrees.

Case 2. A minimal pair strategy having higher priority than a high strategy. For example, $\mu \hat{\infty} \subseteq \alpha$, where α is making A high: μ is enumerating its c.e. set $W = \Theta^A \cap \Psi^B$ and α is extracting the decidable set E_α .

In this case, the extraction of E_α by α could interfere with μ by making some v out of Θ^A in a Π_2^0 -way. One possible scenario would be: μ has enumerated v into W ; then α extracts x_0 out of E_α to cause v out of Θ^A . Now we modify the high strategy by adding a restraint on B and let α have outcome v (on top of its own Π_2 -outcome ∞). If v never comes back into Θ^A then μ has a Σ_2 -outcome, and thus we win easily. Suppose v comes back via another axiom, say $\langle v, D \rangle \in \Theta$ and $D \subseteq A$. There are two possibilities. One is $D \cap E_\alpha = \emptyset$, in other words D has nothing to do with α : we delay the discussion to Case 3, when we deal with the interference of D with another A -high strategy α' . The other possibility is $D \cap E_\alpha \neq \emptyset$, say $x_1 \in D \cap E_\alpha$. When α extracts x_1 , v is out of Θ^A again. This cycle can continue forever. However the modified strategy works: so long as we keep the B -restraint (which is of a fixed amount, one axiom is sufficient to keep a number in Ψ^B), every time we visit α , v is out of Θ^A . Consequently, we see a disagreement at α for μ : v is in Ψ^B in a Σ_2 -way and out of Θ^A in a Π_2 -way. We have satisfied both α and μ . Notice that this disagreement v cannot be detected at μ , thus μ 's Σ_2 -outcome v is different from α 's outcome v for μ , where the latter one is a Σ_3 outcome of μ , detected at α .

The outcomes of the modified high strategy (on top of its own Π_2 -outcome ∞) are $\infty < 0 < 1 < \dots < v < \dots$, ordered as described. After α extracts a certain x out of E_α , α examines if there is any v out of Θ_μ^A for some $\mu \hat{\infty} \subset \alpha$: if yes, then α lets the outcome be the least v ; if no, it lets the outcome be ∞ . Later, whenever we visit α , if Θ_μ offers a new axiom to put v back into Θ_μ^A without using numbers in E_α , i.e., the current least disagreement disappears, then we have outcome ∞ .

Case 3. A minimal pair strategy having higher priority than more than one high strategy of the same type. For example, $\mu \hat{\infty} \subseteq \alpha_1 \hat{\infty} \infty_\mu \subseteq \alpha_2$, where μ is enumerating its c.e. set $W = \Theta^A \cap \Psi^B$, α_1 and α_2 are two A -high strategies, and for $i = 1, 2$, α_i extracts the decidable set E_{α_i} respectively. Here the (abused) notation ∞_μ indicates that α_1 not only has the Π_2 -outcome ∞ for itself (highness),

it also has ∞_μ for μ , i.e., α_1 will see more and more $v \in W$ having an axiom $\langle v, D \rangle \in \Theta$ such that $D \cap E_{\alpha_1} = \emptyset$.

We analyze what happens with α_2 considered, which acts as the α in Case 2. When α_2 extracts elements from E_{α_2} to satisfy its own high requirement, it could extract the same v out of Θ^A again. If this v never returns via an axiom $\langle v, C \rangle \in \Theta$ for some $C \cap E_{\alpha_2} = \emptyset$, we have Σ_3 -outcome v for μ at α_2 (v could be a Σ_2 -outcome v for μ , which means that no more axiom in Θ can enumerate v into Θ^A in the remainder of the construction. In this case, μ will notice this, and will have a Σ_2 -outcome at μ). Suppose v returns via $\langle v, C \rangle$ such that $C \cap E_{\alpha_2} = \emptyset$. We have two new concerns.

The first concern is that $C \cap E_{\alpha_1} \neq \emptyset$. In fact, we could have two sequences of disjoint finite sets D_i and C_i such that $\langle v, D_i \rangle \in \Theta$ and $\langle v, C_i \rangle \in \Theta$, $D_i \cap E_{\alpha_1} = \emptyset$, $C_i \cap E_{\alpha_2} = \emptyset$, $D_i \cap E_{\alpha_2} \neq \emptyset$, and $C_i \cap E_{\alpha_1} \neq \emptyset$. Thus v could escape from Θ^A in a Π_2^0 -way due to the combination of α_1 and α_2 .

This worry can be repelled as follows. Notice that this worry materializes only when $\alpha_1 \hat{\infty}_\mu \subseteq \alpha_2$. Thus α_2 “knows” the set E_{α_1} which α_1 wants to extract. Thus when α_2 sees v returning to Θ^A via an axiom $\langle v, C \rangle$ with $C \cap E_{\alpha_1} \neq \emptyset$, it should not “believe” this computation, hence α_2 would still have outcome v and keep its restraint on the B -side to keep v in Ψ^B . The strategy α_2 only drops this B -restraint and has outcome ∞_μ when it sees an axiom $\langle v, G \rangle$ such that $G \cap (E_{\alpha_1} \cup E_{\alpha_2}) = \emptyset$. This is another place where we use *believable computations*. Thus a computation $\langle v, G \rangle \in \Theta$ is α_2 -believable if $\langle v, G \rangle \in \Theta$ is μ -believable and for all α such that $\mu \hat{\infty} \subset \alpha \hat{\infty}_\mu \subset \alpha_2$, $G \cap E_\alpha = \emptyset$.

The second new concern is that $C \cap E_\alpha \neq \emptyset$ and α goes further and further down the tree, where $\langle v, C \rangle$ is the axiom putting v back into Θ^A . To be more precise, imagine the following scenario. The same $v \in W$ was extracted out of Θ^A finitely many times by α_1 before we see the axiom $\langle v, D \rangle \in \Theta$ showing up; after this, v has nothing to do with α_1 . Then α_2 may do the same, i.e., extract v out of Θ^A finitely often, after which α_1 and α_2 have nothing to do with v . We have to prevent the same phenomenon from repeating infinitely often, in other words, we do not want a sequence of α_i such that each individual α_i extracts v out of Θ^A finitely many times, but the accumulative effect makes v out of Θ^A in a Π_2 -way. Thus we modify the high strategy α again by adding a threshold $k = k_\alpha$; and require that α only extracts numbers from E_α without causing $v < k$ out of Θ^A . A better way to implement this is to truncate the set E_α so that the minimum element x_0 in E_α is sufficiently large (the exact amount of how sufficiently large will be explained in more detail later). Since k would be fixed, the effect on α is equivalent to an extra finite restraint.

Case 4. More than one minimal pair strategy having higher priority than a high strategy. For example, $\mu_1 \hat{\infty} \subseteq \mu_2 \hat{\infty} \subseteq \alpha$.

In this case, the extraction at α could make v_1 out of $\Theta_{\mu_1}^A$ and at the same time make v_2 out of $\Theta_{\mu_2}^A$. Thus we may use a subtree of height two to explicitly indicate the impact. The outcome of α can be visualized as having two levels; the first level is $\infty < 0 < 1 < \dots$ indicating the outcomes for μ_1 ; for each outcome $o \in \omega \cup \{\infty\}$, we further have the second level $\infty < 0 < 1 < \dots$ indicating

the outcomes for μ_2 . It has to be said that the particular order of levels is not essential, as α will not injure μ_2 for the sake of satisfaction of μ_1 . In theory, one could have a binary tree, but spelling the outcomes out would make the later description of the construction easier.

Naturally, if there are n minimal pair strategies above α , then the Π_2 -outcome of α would be something like an ω -branching tree of height n . To implement the idea and to keep the structure of the priority tree simple, let us crush the subtree and use $\langle \infty, \sigma \rangle$ to denote the outcome of a high strategy α or β . The first component ∞ indicates the original outcome of the high strategy; and the second component σ is a string of length n over the alphabet $\{\infty < 0 < 1 < 2 < \dots\}$ where $<$ is the left-to-right order. We extend the left-to-right order to strings in a natural way: $\sigma_1 < \sigma_2$ if and only if $\sigma_1(j) < \sigma_2(j)$ where j is least position at which they disagree.

Finally the discussion relative to P^B -strategies β and their interactions with minimal pair strategies is the same.

2.3 Construction

First let us describe the priority tree. We have three types of strategies: μ for M -requirements, α for P^A -requirements, and β for P^B -requirements. The outcomes of μ are $\infty < 0 < 1 < \dots$, whereas the outcomes of α and β are:

$$\langle \infty, \sigma_0 \rangle < \langle \infty, \sigma_1 \rangle < \dots < 0 < 1 < \dots$$

where each σ_i is a string of length n as described at the end of the previous subsection.

We now define the priority tree together with an ordered list L of *active* minimal pair strategies by simultaneous induction. Suppose that we have defined everything for all nodes $\subset \tau$.

Case 1. $|\tau| = 3e$ for some $e \in \omega$. Then we assign the minimal pair strategy $\mu_{\Theta, \Psi}$ to τ , where Θ and Ψ form the e -th pair of e -operators under our fixed computable enumeration of all pairs of e -operators. Append μ to the list $L_{\tau \hat{\ } \infty}$ of $\tau \hat{\ } \infty$; and keep the list unchanged at $\tau \hat{\ } v$ for all $v \in \omega$.

Case 2. $|\tau| = 3e + 1$ for some $e \in \omega$. Then we assign the A -high strategy α working for P_e^A to τ . The outcome of α will be of the form $\langle \infty, \sigma \rangle$ or $i \in \omega$, where σ is of the same length as the list L_τ . Modify the list at extensions of τ as follows. No change for $\tau \hat{\ } s$. At $\tau \hat{\ } \langle \infty, \sigma \rangle$, delete the j -th item from L_τ if $\sigma(j) \neq \infty$.

Case 3. $|\tau| = 3e + 2$ for some $e \in \omega$. Just replace A by B in Case 2.

We next describe the environments and parameters associated with each node on the priority tree, in addition to the list L_τ inherited from the definition of the tree.

For a minimal pair strategy μ , the environment has one c.e. set W_μ . When μ is *initialized*, W_μ is discarded and is replaced by the empty set.

For each A -high strategy α , the environment has a threshold k_α and an infinite decidable set E_α . When α is initialized, the parameter k_α is discarded; and although, in some sense, we do not change E_α , the change of threshold has the effect of effectively pushing the minimal element in E_α to a larger number.

The environment and parameters for a B -high strategy β are defined similarly.

We are now ready to describe the construction. At stage $s = 0$ we start up with $A = B = \omega$, and initialize all strategies.

At stage $s > 0$, we define an accessible string δ_s by recursion, and we describe its action if it acts. A stage t is called σ -true if $\sigma \subseteq \delta_t$.

$\delta_s(0)$ is the root of the priority tree.

Suppose that we have defined $\delta_s(i)$ for some $i < s$. We define $\delta_s(i + 1)$ as follows.

Case 1. $\delta_s(i)$ is a minimal pair strategy μ , relatively to the pair of e-operators Θ and Ψ , and building the c.e. set W .

First we define the meaning of a computation being μ -believable. Let T^A be the set of all high strategies α such that $\alpha \hat{\ } \langle \infty, \sigma \rangle \subseteq \mu$. A computation $\langle v, D \rangle \in \Theta$ is μ -believable if $D \cap (\bigcup_{\alpha \in T^A} E_\alpha) = \emptyset$. Similarly we can define a μ -believable computation for Ψ^B .

Suppose that v was the outcome at μ when we last visited μ , after last initialization, at stage, say, s^- ; if μ has not been visited yet after last initialization, or no outcome was defined at s^- , then let $v = \infty$.

- (1.1) Suppose $v \neq \infty$, and, say, v is A -related (i.e., at s^- we had $v \in \Psi^B \setminus \Theta^A$; the notion of B related is defined accordingly). If currently $v \in \Theta^A$ (i.e., due to a new axiom), then let ∞ be the outcome at μ , and drop the existing B -restraint. If v is B -related then the action is symmetric, interchanging A with B ;
- (1.2) Suppose $v = \infty$. Take the least $w \in W$ such that $w \in \Theta^A \cup \Psi^B$, but $\Theta^A(w) \neq \Psi^B(w)$. Let w be the outcome and set up an A -restraint or a B -restraint, according to whether $w \in \Theta^A$ or $w \in \Psi^B$. (To set up an A -restraint amounts as usual to request that extractions performed by lower priority strategies do not interfere with keeping $w \in \Theta^A$; similarly for a B -restraint.) If there is no such w then let ∞ be the outcome.

If the stage is μ -expansionary, i.e. the outcome is ∞ , then we enumerate into W the least element in $\Theta^A \cap \Psi^B \setminus W$.

Case 2. $\delta_s(i)$ is an A -high strategy α .

If the threshold $k = k_\alpha$ is not defined, then define it big. Let x^* be the least number in $E_\alpha \cap A$ such that x^* is bigger than the amount of restraint that has been set up to keep all necessary $v < k$ into Θ_μ^A , for all relevant μ as explained later. Reset $E_\alpha = E_\alpha \cap \{x : x \geq x^*\}$.

If the threshold k is defined, then keep the same E_α . Suppose that $E_\alpha = \{x_0 < x_1 < \dots\}$.

If s is the first α -true stage after last initialization of α then we simply let $\alpha \hat{\ } 0$ be accessible. Otherwise, suppose that j is the largest number such that there has

been an $\alpha^{\wedge}j$ -true stage after last initialization. We say that s is α -expansive if for all y such that $y < x_{j+1}$, $\varphi_e(y) \downarrow [s]$.

- (2.1) If s is α -expansive, then let $\langle \infty, \sigma \rangle$ be accessible where σ is determined as follows. Suppose that $\langle \infty, \tau \rangle$ was the outcome when we visited α for the last time: if α has never been visited after it was initialized last time, or the outcome was not of this type, then let $\tau = \langle \infty, \infty, \dots, \infty \rangle$. For $j = 1$ to the length of σ , let μ_j be the j -th element in the active list L_α . We say that a computation $\langle v, D \rangle \in \Theta_{\mu_j}$ is α -believable if $D \cap E_{\alpha'} = \emptyset$ for all A -high strategies α' such that $\alpha' = \alpha$ or $\mu_j \hat{\infty} \subseteq \alpha' \hat{\infty} \subseteq \alpha$ where $\sigma'(j) = \infty$.
- If $\tau(j) = \infty$ and v is the least number in $W_{\mu_j} \setminus \Theta_{\mu_j}^A$ (due to the absence of α -believable computations), then let $\sigma(j) = v$. (A restraint is imposed by this bit of σ on B to keep v in $\Psi_{\mu_j}^B$.) If no such v exists, let $\sigma(j) = \infty$ (No restraint for this bit of σ).
 - If $\tau(j) = v$ for some v , and there is $\langle v, D \rangle \in \Theta_{\mu_j}$ which is α -believable, then let $\sigma(j) = \infty$ and drop the restraint imposed by this bit of σ . Otherwise, let $\sigma(j) = v$ and keep the restraint imposed earlier on B to keep v in $\Psi_{\mu_j}^B$.
- (2.2) Otherwise, s is not α expansive. If the previous α -true stage was not α -expansive then let $\alpha^{\wedge}j$ be accessible; otherwise let $\alpha^{\wedge}(j + 1)$ be accessible.

The total restraint at the end of $\alpha^{\wedge} \langle \infty, \sigma \rangle$ is the union of all restraints imposed at each bit of σ .

Case 3. $\delta_s(i)$ is a B -high strategy β .

Symmetric.

When $i = s$ we stop defining the accessible string and initialize all nodes to the right of δ_s .

Choose the least x ,

$$x \in \bigcup \{ (E_\alpha \cap A) \cup (E_\beta \cap B) : s \text{ is } \alpha\text{-expansive or } \beta\text{-expansive} \},$$

extract x out of A or B , respectively.

This finishes the construction.

2.4 Verification

First of all it is clear that A and B are both Π_1^0 sets, since we never enumerate back elements after extraction.

Lemma 1. *The true path exists.*

Proof. We must show that there exists an infinite path f through the tree of strategies such that for every n , $\sigma_n = f \upharpoonright n$ is the leftmost string of length n which is accessible infinitely many times. This is done by routine induction. Assuming that σ_n exists, and the relative parameters reach a limit, one shows

that claim is true of $n + 1$ as well. If σ_n is an M -strategy, then there is a leftmost outcome, since every time we abandon an outcome v we give outcome ∞ . If σ_n is a high strategy then the proof is a bit subtler. If σ_n is assigned, say, requirement P_e^A and $E_{\sigma_n} = \{x_0 < x_1 < \dots\}$ is the final value of the parameter, then the outcome is j where j is the least number such that for some y with $y < x_{j+1}$ we have $\varphi_e(y) \uparrow$. Otherwise the outcome is of the form $\langle \infty, \sigma \rangle$. In this case, the claim follows from the fact that if we have a correct guess of the first i many active minimal pair strategies, then the $i + 1$ -th active minimal pair strategy will have outcome ∞ after seeing that more and more numbers are in Θ^A via believable computations.

Let f denote the true path.

Lemma 2. *Every high requirement is satisfied.*

Proof. We limit ourselves to an A -high strategy α on the true path. The case of B -high requirements is dealt with similarly.

Let s_0 be the stage after which α is never initialized. Thus the set $E_\alpha = \{x_0 < x_1 < \dots\}$ is fixed after s_0 . If $\alpha \hat{\ } j \subset f$, then φ_e is partial at some point $y < x_{j+1}$. Thus P_e^A is satisfied. If $\alpha \hat{\ } \langle \infty, \sigma \rangle \subset f$ for some σ , then eventually α is able to extract every element $x_i \in E_\alpha$ above the threshold, out of A . To see this, suppose by induction that at some $\alpha \hat{\ } \langle \infty, \sigma \rangle$ -true stage all $x_j \in E_\alpha$, with $j < i$, have been already extracted from A , but x_i is still in A ; at each future $\alpha \hat{\ } \langle \infty, \sigma \rangle$ -true stage, if not already extracted, x_i is a candidate for extraction, unless there is a smaller element that qualifies for extraction too; but eventually x_i becomes the smallest one. Note that when α has a Σ_3 -outcome, it imposes a restraint on B , not on A . As argued in the description of a single requirement, we have that $C_A(x) > \varphi_e(x)$ for all x larger than the minimal element in E_α .

Lemma 3. *Every minimal pair requirement $M_{\Theta, \Psi}$ is satisfied.*

Proof. Let μ be the strategy on true path which is working on $M_{\Theta, \Psi}$.

Suppose $\mu \hat{\ } v \subset f$. Let s_0 be the least stage at which $\mu \hat{\ } v$ is accessible, and after which $\mu \hat{\ } v$ is never initialized. Then the restraint imposed by μ in (1.1) of the construction is permanent. All nodes α and β to the right or extending $\mu \hat{\ } v$ obey this restraint; strategies α and $\beta \subseteq \mu$ will either not act or will not affect μ , as μ used believable computations. This makes $\Theta^A(v) \neq \Psi^B(v)$.

Suppose $\mu \hat{\ } \infty \subset f$. Here is the case where we need to know the whole true path in order to know how $M_{\Theta, \Psi}$ is satisfied.

Case 1. There is an A -high strategy α on the true path, such that $\alpha \hat{\ } \langle \infty, \sigma \rangle$ is on true path, and μ is the j -th item μ_j in the list L_α and $\sigma(j) = v$ for some $v \in \omega$. Let s_1 be the stage after which $\alpha \hat{\ } \langle \infty, \sigma \rangle$ is never initialized. Then after stage s_1 the restraint imposed on B at $\alpha \hat{\ } \langle \infty, \sigma \rangle$ is permanent. So $v \in \Psi^B$. On the A -side, by (2.1) in the construction, we will not see any α -believable computation $\langle v, C \rangle \in \Theta^A$. Thus $C \cap E_{\alpha'} \neq \emptyset$ for some A -high strategies α' such that either $\alpha = \alpha'$ or $\mu_j \hat{\ } \infty \subseteq \alpha' \hat{\ } \langle \infty, \sigma' \rangle \subseteq \alpha$ where $\sigma'(j) = \infty$. Hence $v \notin \Theta^A$ since we extract all such $E_{\alpha'}$ out of A .

Case 2. The case of a B -high strategy β on the true path is symmetric.

Case 3. For every high strategy α on true path, if $\alpha \hat{\langle \infty, \sigma \rangle}$ is on true path, and for some j , μ is the j -th item μ_j in the list of L_α , then $\sigma(j) = \infty$.

Assume that $\Theta^A = \Psi^B$. We claim that $W \subseteq \Theta^A$. For the sake of a contradiction, pick the least $v \in W \setminus \Theta^A$. By the device of the threshold, only finitely many nodes may extract v out of Θ^A . Since $\mu \hat{\infty}$ is on the true path, one of these nodes must extract v infinitely often, and this node has to be on the true path, say the least such one is α . Then $\alpha \hat{\langle \infty, \sigma \rangle}$ would be on true path for some σ having $\sigma(j) = v$. Contradiction.

Similarly we can show that $W \subseteq \Psi^B$. Thus $W = \Theta^A \cap \Psi^B$. Clearly, by the construction, each element in $\Psi^B \cap \Theta^A$ is enumerated into W sooner or later, and hence $W \supseteq \Theta^A \cap \Psi^B$. Therefore, $W = \Theta^A \cap \Psi^B = \Psi^B = \Theta^A$.

Finally, notice that we successfully avoid meshing as we never allow any $v \in W$ escape both Θ^A and Ψ^B in a Π_2 -fashion.

References

1. Cooper, S.B., Copstake, C.S.: Properly Σ_2 enumeration degrees. *Z. Math. Logik Grundlag. Math.* 34, 491–522 (1988)
2. Lachlan, A.H.: Lower bounds for pairs of recursively enumerable degrees. *Proc. London Math. Soc.* 16, 537–569 (1966)
3. McEvoy, K.: The Structure of the Enumeration Degrees. PhD thesis, School of Mathematics, University of Leeds (1984)
4. McEvoy, K., Cooper, S.B.: On minimal pairs of enumeration degrees. *J. Symbolic Logic* 50, 983–1001 (1985)
5. Shore, R., Sorbi, A.: Jumps of Σ_2^0 high e-degrees and properly Σ_2^0 e-degrees. In: Arslanov, M., Lempp, S. (eds.) *Recursion Theory and Complexity*. De Gruyter Series in Logic and Its Applications, pp. 157–172. W. De Gruyter, Berlin (1999)
6. Yates, C.E.M.: A minimal pair of recursively enumerable degrees. *J. Symbolic Logic* 31, 159–168 (1966)

Searching a Circular Corridor with Two Flashlights

Bo Jiang¹ and Xuehou Tan^{1,2,*}

¹ School of Inform. Sci. and Tech., Dalian Maritime University, China

² Tokai University, 317 Nishino, Numazu 410-0395, Japan

`tan@wing.ncc.u-tokai.ac.jp`

Abstract. We consider the problem of searching for a mobile intruder in a *circular corridor* (a polygon with one polygonal hole) by two searchers, who hold a flashlight. Both searchers move on the outer boundary, directing their flashlights at the inner boundary. The objective is to decide whether there exists a *search schedule* for the searchers to detect the intruder, no matter how fast he moves. We give a characterization of the class of circular corridors, which are searchable with two flashlights. Based on our characterization, an $O(n \log n)$ time algorithm is then presented to determine the searchability of a circular corridor with two flashlights, where n denotes the total number of vertices of the outer and inner boundaries. Moreover, a search schedule can be output in time linear in its size, if it exists. Our result gives the first efficient solution to the polygon search problem for two searchers.

1 Introduction

Motivated by the relation to the well-known *Art Gallery* problem, much attention has recently been devoted to the problem of searching for a mobile intruder in a polygonal region P by a mobile searcher [6, 7, 9, 10, 11]. Both the searcher and the intruder are modeled by points that can continuously move in P , and the intruder is assumed to be able to move arbitrarily faster than the searcher. The intruder is said to be detected if he is ever within the vision of the searcher. The polygon is said to be *searchable* if there exists a schedule for the searcher to detect the intruder.

The visibility of a searcher can be defined by the flashlights he holds. The so-called *1-searcher* has a flashlight and can see only along the ray of the flashlight emanating from his position. The direction of the flashlight can be changed continuously with bounded angular rotation speed. The 1-searcher has to move on the polygon boundary continuously, but the endpoint of the ray emanating from the 1-searcher may not be continuous [6, 10]. If the endpoint of the ray emanating from the 1-searcher is also required to move on the polygon boundary continuously, it introduces a slightly different type of 1-searchers, which is usually termed *two guards* [3, 4].

* Corresponding author.

A large number of papers on the polygon search problem has been published in the computational geometry and robotics literature, since the polygon search problem was first introduced in 1992 [9]. It is mainly because this problem captures the key issues in various robotics applications. Efficient algorithms that compute search strategies can be embedded in various types of robotics systems that detect intruders using lasers or cameras. Mobile robots can also be used in a high-risk military action that requires to systematically search a building or an area in enemy territory.

Most researches focus on the problem of searching a simple polygon (without holes) by a single mobile searcher [6, 7, 9, 10, 11]. For example, a characterization of the class of the polygons, which are searchable by a *boundary searcher* (who always moves on the polygon boundary), and a linear time algorithm for determining the searchability of simple polygons have been given in [11]. For searching a simple polygon by two 1-searchers, an $O(n^4)$ time algorithm was reported in [8]. Because of its high complexity, developing an efficient solution for two 1-searchers is left as an interesting open problem. Only very preliminary results on the problem of searching a polygonal region with holes by multiple searchers were known [2].

In this paper, we study the problem of searching a *circular corridor* (i.e., a polygon with one hole) by two 1-searchers [5]. Both 1-searchers move on the outer boundary ∂P , directing their flashlights at the inner boundary ∂H . If there exists a search schedule for the 1-searchers to detect the intruder, we say that the circular corridor is *searchable with two flashlights*, or simply *searchable*. Since two searchers have to cooperatively search the circular corridor, the problem is much difficult and challenging. A systematical study on the cooperative motions of two searchers may open the door to more efficient solutions to the polygon search problem for multiple searchers.

The main contributions of this paper are the followings. First, we give a characterization of the class of circular corridors searchable with two flashlights, in terms of weak visibility and deadlocks (which are not searchable with only one flashlight [4]). It is obtained by a through study on the structures of the deadlocks which restrict the motions of two flashlights. Based on our characterization, an $O(n \log n)$ time algorithm is then presented to determine the searchability of a circular corridor with two flashlights, where n denotes the total number of vertices of the outer and inner boundaries. Moreover, a search schedule can be output in time linear in its size, if it exists. Our result gives the first efficient solution to the polygon search problem for two searchers.

2 Preliminaries

A circular corridor, denoted by CC , is defined as a polygon with one polygonal hole. The outer boundary of CC is denoted by ∂P , and the inner boundary is denoted by ∂H . For ease of presentation, we denote by $P[x, y]$ ($H[x, y]$) the clockwise closed chain of ∂P (∂H) from x to y , and $P(x, y)$ ($H(x, y)$) the open chain of ∂P (∂H) from x to y . For two points $p \in \partial P$, $h \in \partial H$, we say that

they are mutually *visible* if every interior point of the line segment \overline{ph} lies in the interior of CC , except for two endpoints p and h . For two chains $P[x, y]$ and $H[x', y']$, we say that $P[x, y]$ is *weakly visible* from $H[x', y']$ if every point of $P[x, y]$ is visible from some point of $H[x', y']$.

For a vertex v of CC , we denote by $Pred(v)$ (resp. $Succ(v)$) the vertex of CC immediately preceding (resp. succeeding) v on the boundary ∂P or ∂H clockwise. A vertex of CC is *reflex* if its interior angle is strictly larger than 180° ; otherwise, it is *convex*. An important definition for reflex vertices is that of *ray shots*: the backward ray shot from a reflex vertex r of ∂P (∂H), denoted by $B(r)$, is the first point of ∂H (∂P), if it exists, hit by a “bullet” shot at r in the direction from $Succ(r)$ to r , and the forward ray shot $F(r)$ is the first point of CC hit by the bullet shot at r in the direction from $Pred(r)$ to r . The vertex r is called the *origin* of the shots $B(r)$ and $F(r)$.

A pair of vertices $p \in \partial P, h \in \partial H$ is said to form a *backward deadlock* if both the three points $p, Succ(p), B(h) \in \partial P$ and the three point $h, Succ(h), B(p) \in \partial H$ are in clockwise order (Fig. 1(a)), or a *forward deadlock* if both the three points $F(h) \in \partial P, Pred(p), p$ and the three points $F(p) \in \partial H, Pred(h), h$ are in clockwise order (Fig. 1(b)). Two vertices p and h are called the *defining vertices* of the deadlock. Also, the edges $\overline{pSucc(p)}$ and $\overline{hSucc(h)}$ in Fig. 1(a) (resp. $\overline{pPred(p)}$ and $\overline{hPred(h)}$ in Fig. 1(b)) are called the *defining edges* of the deadlock.

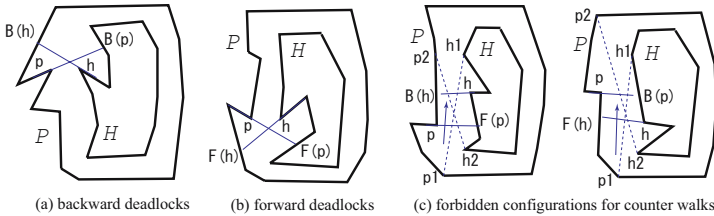


Fig. 1. Illustration of ray shots and deadlocks

2.1 Problem Definition

Let $s_i(t) \in \partial P$ and $f_i(t) \in \partial H$ denote the position of the 1-searcher i ($i = 1, 2$) and the endpoint of his flashlight at a time $t > 0$, respectively. A point $x \in CC$ is said to be *detected* or *illuminated* at time t if x is on the line segment $\overline{s_1(t)f_1(t)}$ or $\overline{s_2(t)f_2(t)}$. Any region that might contain the intruder at a time is said to be *contaminated*; otherwise, it is *clear*. The given CC is said to be *searchable* with two flashlights if there exists a search schedule that finally clears CC (i.e., the intruder is eventually on the ray emanating from either 1-searcher, no matter how fast he moves).

A search schedule of two 1-searchers consists of the following basic actions [6, 10]: The searcher $s_i(t)$ ($i = 1, 2$) and the endpoint $f_i(t)$ of his flashlight move along segments of single edges such that (i) no intersections occur among all line segments $\overline{s_i(t)f_i(t)}$ during the movement or (ii) any two of segments $\overline{s_i(t)f_i(t)}$

intersect each other, and (iii) $f_i(t)$ jumps from a reflex vertex x to the other point y or from the point y to the vertex x . Note that except for the movement of f_i performed by instructions (iii), all other movements of s_i and f_i are continuous on ∂P and ∂H , respectively.

In the rest of this paper, we denote by S_a (resp. S_b) the 1-searcher whose flashlight moves in clockwise (resp. counterclockwise) direction. Fig. 2 illustrates an example for clearing a circular corridor, where the shaded region denotes the clear portion of CC and the dotted arrows give the directions in which the flashlights move. The starting positions of two flashlights are shown in Fig. 2(a). Note that the ray of the flashlight emanating from the 1-searcher s_a separates the defining edges of several deadlocks on ∂P from those on ∂H . The 1-searcher s_b first uses his flashlight to clear these defining edges on ∂P . See Fig. 2(b). Next, s_b aims his flashlight at the vertex r of ∂H (Fig. 2(c)), and s_a moves his flashlight clockwise over r (Fig. 2(d)). Two searchers can then move clockwise and counterclockwise, respectively (Fig. 2(e)). Finally, they work together to clear CC (Fig. 2(f)).

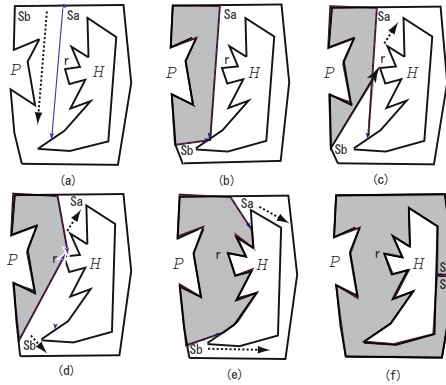


Fig. 2. Snapshots of a search schedule

As shown in Fig. 2, two 1-searchers may cooperatively clear a group of deadlocks in the beginning or ending of a search schedule. Specifically, while a 1-searcher uses the ray of the flashlight to separate the defining edges of the deadlocks on ∂P from those on ∂H , the other moves to clear the defining edges on ∂P and some part of ∂H as well. We will use the "start phase" and the "end phase" to represent two different time periods, in which the 1-searchers cooperatively clear a group of deadlocks (if it exists). To be exact, two 1-searchers start at the same position and then move to clear a group of deadlocks in the start phase. Analogously, after they move to clear a group of deadlocks, two 1-searchers finish the search at the same position in the end phase.

2.2 Related Work

We briefly review the solution to the two-guard problem [3, 4], which will be used in our proof and the search algorithm as well. A *corridor* is a simple polygon

Q with a point u on the boundary called the 'entrance' and the other v on the boundary called the 'exit'. We denote it by (Q, u, v) . Also, denote by L (resp. R) the clockwise (resp. counterclockwise) chain Q from u to v . The deadlocks between L and R can be defined with a slight modification that the three points of a deadlock on R are in counterclockwise order [4]. The *two-guard problem* for (Q, u, v) asks whether there is a walk from u to v such that two guards move along L and R , one clockwise and the other counterclockwise. Note that the instructions (i) and (ii) are allowed for two guards [4].

Lemma 1. (See [3, 4].) *A corridor (Q, u, v) is walkable by two guards if and only if L and R are mutually weakly visible and no deadlocks occur between L and R . It takes $\Theta(n)$ time to test the two-guard walkability of a corridor, and $O(n \log n + m)$ time to output an optimal walk schedule, where $m (\leq n^2)$ is the number of the instructions reported.*

A *counter walk* is a walk in which one guard moves clockwise on L from u to v and the other moves clockwise on R from v to u , in such a way that they are always mutually visible. (Clearly, u and v are mutually visible.) In order for (Q, u, v) to have a counter walk, some configurations of non-crossing ray shots from different chains are prohibited (see Fig. 12(i)) of [4]. For either example shown in Fig. 1(c), the counter walk from p_1h_1 to p_2h_2 is not allowed, because of the forbidden configuration of non-crossing ray shots from p and h .

Lemma 2. (See [3, 4].) *A corridor (Q, u, v) is counter-walkable by two guards if and only if L and R are mutually weakly visible and the configurations of non-crossing ray shots shown in Fig. 12(i) of [4] do not exist. It takes $\Theta(n)$ time to test the counter-walkability of a corridor, and $O(n \log n + m)$ time to output an optimal walk schedule, where $m (\leq n^2)$ is the number of the instructions reported.*

It is clear that the above results also hold for the 1-searcher. Note that u or v can be defined as an edge of Q . In this case, only the parts of L and R walked by the guards are considered [4].

3 Necessary Conditions

In this section, we present necessary conditions for circular corridors to be searchable with two flashlights. The first condition simply comes from weak visibility between ∂P and ∂H .

Lemma 3. *If ∂P and ∂H are not mutually weakly visible, then CC is not searchable.*

Proof. Simple and omitted. □

From now on, assume that ∂P and ∂H are mutually weakly visible. Assume also that all points on ∂P (resp. ∂H) are ordered with respect to a point $p_0 \in \partial P$

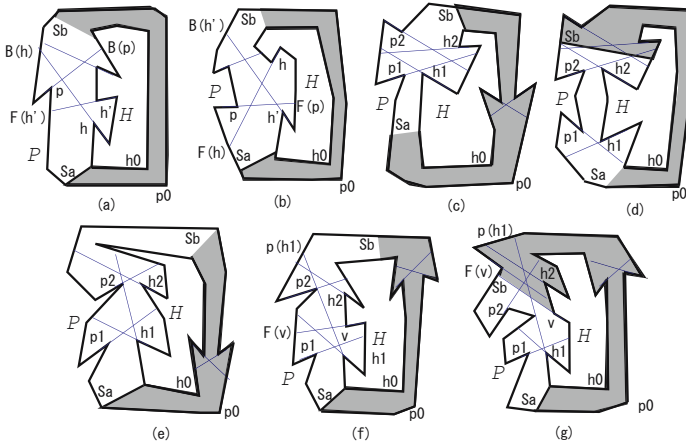


Fig. 3. Forbidden configurations

(resp. $h_0 \in \partial H$). So the inequality $a < b$ implies that a is encountered before b by a clockwise walker on the boundary, starting at p_0 or h_0 . Usually, we assume that p_0 and h_0 are contained in the clear region. See Fig. 3. (The exact position of p_0 or h_0 is not important, as it is used only to give the order of boundary points.)

Let us now study the structures of deadlocks, which cannot be cleared in the end (or start) phase. The simplest forbidden configuration consists of a deadlock and a reflex vertex. Three vertices p , h and h' are said to form a *strong backward* (resp. *forward*) deadlock if p and h form a backward (resp. forward) deadlock, and $F(h') < p$ and $h < h'$ (resp. $p < B(h')$ and $h' < h$) hold. See Figs. 3(a) and 3(b). The vertices p , h and h' (resp. their corresponding edges) are called the *defining vertices* (resp. *defining edges*) of the strong deadlock.

Lemma 4. *A strong deadlock cannot be cleared in the end phase.*

Proof. Denote by p , h and h' the three vertices of a strong backward (resp. forward) deadlock in the contaminated region. See Fig. 3(a) (resp. Fig. 3(b)). The lemma simply follows from a claim that the edge $hSucc(h)$ (resp. $hPred(h)$) cannot be cleared in the end phase. Since our claim is rather simple, its rigorous proof is omitted in this extended abstract. \square

There are more other configurations of deadlocks, which cannot be cleared in the end phase, either. Informally, whether a group of deadlocks can be cleared in the end phase depends on whether the defining edges of the deadlocks on ∂P can be separated from those on ∂H by the ray of a flashlight, and whether the whole chain ∂H can be cleared by two 1-searchers cooperatively. To be more precise, we introduce below the concept of "non-separated" deadlocks.

Denote by $\langle p_1, h_1 \rangle$ and $\langle p_2, h_2 \rangle$, $p_1 < p_2$ and $h_1 < h_2$, two pairs of the defining vertices of deadlocks. Let $p(h_1)$ denote the first point succeeding p_2 (with respect to the starting point p_0), which is visible from h_1 , if it exists. Probably, the

point $p(h_1)$ is undefined (see Fig. 3(e)). Similarly, let $p(h_2)$ denote the last point preceding p_1 , which is visible from h_2 , if it exists. Moreover, let v (resp. u) be the vertex of $H(h_1, h_2)$ (resp. $H(h_1, h_2)$) such that $F(v) \in \partial P$ (resp. $B(u) \in \partial P$) is the smallest (resp. largest) among those forward (resp. backward) shots, if it exists.

Assume that no strong deadlocks occur among p_1, h_1, p_2 and h_2 . The deadlocks formed by $\langle p_1, h_1 \rangle$ and $\langle p_2, h_2 \rangle$ are said to be *non-separated* if one of the followings is true.

- (i) The deadlock formed by $\langle p_1, h_1 \rangle$ is the backward one and the deadlock formed by $\langle p_2, h_2 \rangle$ is the forward one (Fig. 3(c)). Note that $p_1 < F(h_2)$ and $B(h_1) < p_2$ hold; otherwise, a strong deadlock occurs. For simplicity, we call them a pair of *BF-deadlocks*.
- (ii) The deadlocks formed by $\langle p_1, h_1 \rangle$ and $\langle p_2, h_2 \rangle$ are respectively the forward and backward ones, and the chain $H[h_1, h_2]$ is not weakly visible from $P[p_2, p_1]$ or there are no internal line segments \overline{ph} such that \overline{ph} separates the defining edges of them on ∂P from those on ∂H (Fig. 3(d)). We call them a pair of *FB-deadlocks*.
- (iii) For two backward deadlocks, the point $p(h_1)$ is undefined (Fig. 3(e)), $p_1 < F(v) < p_2$ (Fig. 3(f)) or the condition for the counter-walk from $\overline{F(v)v}$ to $\overline{p(h_1)h_1}$ is not satisfied when $p_2 < F(v) < p(h_1)$ (Fig. 3(g)). Note that $p_1 < F(v)$ holds; otherwise, the vertices p_1, h_1 and v form a strong backward deadlock. We call them a pair of *BB-deadlocks*.
- (iv) For two forward deadlocks, the point $p(h_2)$ is undefined, $p_1 < B(u) < p_2$ holds, or the condition for the counter-walk from $\overline{B(u)u}$ to $\overline{p(h_2)h_2}$ is not satisfied when $p(h_2) < B(u) < p_1$. Also, $B(u) < p_2, h_1 < u$ holds. We call them a pair of *FF-deadlocks*.

The definition of non-separated deadlocks is relatively complicated. This is because there are so many different configurations of deadlocks in CC . Note that the vertices p_1, h_1, p_2 and h_2 giving a pair of non-separated deadlocks are not unique. The concepts of non-separated deadlocks and strong deadlocks are important, as they not only help give a characterization of searchable circular corridors but also make it possible to develop an efficient solution to this difficult and challenging problem. Note also that if there are no internal segments \overline{ph} such that \overline{ph} separates the defining edges of two backward (resp. forward) deadlocks in CC , then a pair of *BB-deadlocks* (resp. *FF-deadlocks*) occurs.

Lemma 5. *If two deadlocks are non-separated, they cannot be cleared in the end phase.*

Proof. Suppose first that there is a pair of *BF-deadlocks* in the contaminated region. See Fig. 3(c) for an example. In this case, neither S_a nor S_b can further move to any point of $P(p_1, p_2)$; otherwise, CC becomes contaminated, except for two line segments illuminated by the flashlights. For the same reason, neither endpoint of the flashlights can be moved to any point of $H(h_1, h_2)$. Hence, the pair of *BF-deadlocks* cannot be cleared in the end phase.

Assume now that a pair of FB -deadlocks occurs in the contaminated region. If there are no internal segments \overline{ph} such that \overline{ph} separates the defining edges of the deadlocks in CC , neither flashlight can be used to separate the defining edges of the deadlocks, and thus, the pair of FB -deadlocks cannot be cleared in the end phase. See Fig. 3(d). Otherwise, $H[h_1, h_2]$ is not weakly visible from $P[p_2, p_1]$. So the chain $H[h_1, h_2]$ cannot be cleared in the end phase. This is because S_a (resp. S_b) cannot move over p_1 clockwise (resp. p_2 counterclockwise), before all defining edges of uncleared deadlocks on ∂H are cleared. Hence, the pair of FB -deadlocks cannot be cleared.

Finally, consider the situation in which a pair of BB -deadlocks occurs in the contaminated region. (The situation where a pair of FF -deadlocks can be dealt with analogously.) If $p(h_1)$ is undefined, then no point succeeding p_2 is visible from h_1 and thus the movement of S_b is blocked by the deadlock formed by p_2 and h_2 . See Fig. 3(e) for an example. Hence, S_b cannot clear the whole edge $\overline{h_1 Succ(h_1)}$ using his flashlight. On the other hand, since p_1 and h_1 form a backward deadlock, S_a cannot move the endpoint of his flashlight over the vertex h_1 clockwise. Therefore, the chain $H[h_1, h_2]$ cannot be cleared in the end phase, and the pair of BB -deadlocks can never be cleared. For the case $p_1 < F(v) < p_2$, the discussion is the same as above, because $Pred(v)$ cannot be cleared by the flashlight of S_a or S_b . See Fig. 3(f). Consider the situation in which $p_2 < F(v) < p(h_1)$ holds. To clear the vertex $Pred(v)$, the ray of the flashlight of S_b has to move to $vF(v)$ at least once. Since $F(v) < p(h_1)$ holds, the vertex h_1 is contaminated at that time. See Fig. 3(g). Because of the deadlock formed by p_1 and h_1 , it is impossible for S_a to clear the chain $\overline{H[h_1, v]}$. In order for S_b to clear $\overline{H[h_1, v]}$, the counter-walk from $\overline{F(v)v}$ to $\overline{p(h_1)h_1}$ is then required. Hence, if the required counter-walk is not allowed, the pair of BB -deadlocks cannot be cleared. See Fig. 3(g). This completes the proof. \square

By symmetry, a strong deadlock or a pair of non-separated deadlocks cannot be cleared in the start phase, either. The following result shows that all polygons in Fig. 3 are not searchable with two flashlights. Note first that any two of the deadlocks shown in Figs. 3(c) to 3(g) are non-separated. With respect to a strong backward (resp. forward) deadlock formed by p , h and h' , a deadlock is *simple* if its defining vertex on ∂H is contained in $H[h', h]$ (resp. $H(h, h')$). That is, h cannot be the defining vertex of the simple deadlock. But, a defining vertex of the simple deadlock may be identical to the vertex p or h' . See Figs. 3(a) and 3(b) for some examples.

Lemma 6. *Suppose that ∂P and ∂H of the given circular corridor are mutually weakly visible. Then, CC is not searchable with two flashlights if there are a strong deadlock and a simple deadlock, or there are three deadlocks such that any two of them are non-separated.*

Proof. Suppose first that CC contains, say, a strong backward deadlock and a simple deadlock. In order to clear CC , a 1-searcher has to initially use the ray of his flashlight to separate the defining edge of the strong deadlock on ∂P from two other defining edges on ∂H . From the proof of Lemma 4, we can assume

that $H(h, h')$ is contained in the contaminated region and two flashlights cross over the chain $H(h, h')$, at a time $t > 0$. See Figs. 4(a) and 4(b). Consider only the situation in which a defining vertex of the simple deadlock is identical to the vertex h' , as all other situations can be dealt with analogously. Then, two 1-searchers may move to meet, say, at a point $p' < p$ (Fig. 4(a)) or $p' > p$ (Fig. 4(b)), depending on the initial position of the flashlight used to separate the defining edges of the strong deadlock (and the simple deadlock as well). To clear the chain $H[h, h']$, either 1-searcher has further to move over p , to the other side of p . But, it is impossible, because either the deadlock formed by $\langle p, h \rangle$ (Fig. 4(a)) or the simple deadlock (Fig. 4(b)) is encountered. Hence, the chain $H[h, h']$ cannot be cleared, and CC is not searchable.

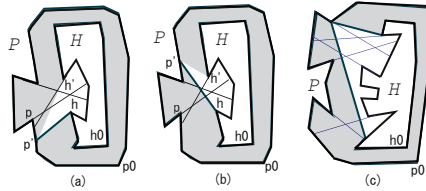


Fig. 4. Illustrating the proof of Lemma 6

Assume now that there are three deadlocks such that any two of them are non-separated. If one of these three deadlocks is cleared in the start phase, a pair of non-separated deadlocks has to be cleared in the end phase, and thus CC is not searchable. See Figs. 3(c) to 3(g) for some examples. A 1-searcher may initially use the ray of his flashlight to separate all defining edges of three deadlocks. See Fig. 4(c). But, the position of this flashlight with respect to these deadlocks cannot be changed after the start phase. The other 1-searcher may walk along ∂P to clear all points of ∂P . When two searchers meet on ∂P again, at least two defining edges of these deadlocks on ∂H are still contaminated (Fig. 4(c)). Since any two of the considered deadlocks are non-separated, two 1-searchers cannot use their flashlights to clear all defining edges on ∂H , while keeping the whole chain ∂P to be clear. Hence, CC is not searchable with two flashlights. \square

4 Sufficiency

In this section, we show that the absence of the configurations described in Lemma 6 ensures that CC is searchable with two flashlights.

Lemma 7. *Suppose that ∂P and ∂H of the given circular corridor are mutually weakly visible. Then, CC is searchable with two flashlights if there are neither three deadlocks such that any two of them are non-separated, nor a strong deadlock and a simple deadlock in CC .*

Proof. Assume first that CC does not contain any strong deadlock. Since there are no three deadlocks such that any two of them are non-separated, we can find

the disjoint line segments \overline{ph} and $\overline{p'h'}$ in CC such that two elements of any pair of non-separated deadlocks are not contained in $P[p, p'] \cup H[h, h']$ nor in $P[p', p] \cup H[h', h]$. See Fig. 5. (Probably, the defining edges of a considered deadlock belong to $P[p, p'] \cup H[h, h']$, but their ray shots are contained in $P[p', p] \cup H[h', h]$.) Next, we show that the deadlocks appeared in between $P[p, p']$ and $H[h, h']$ (resp. $P[p', p]$ and $H[h', h]$) can be cleared in the end (resp. start) phase. Again, we only discuss how to clear the deadlocks in the end phase, assuming that all points of $P[p', p] \cup H[h', h]$ have been cleared. Note also that a 1-searcher (resp. the ray of his flashlight) can move on $P[p', p]$ (resp. $H[h', h]$) in the end phase.

Assume that p_1 and p_2 (resp. h_1 and h_2) are the defining vertices of some deadlocks such that all defining vertices of uncleared deadlocks on ∂P (resp. ∂H) belong to $P[p_1, p_2]$ (resp. $H[h_1, h_2]$). We discuss only the situation in which p_1 and h_1 differ from p_2 and h_2 , respectively. (Other situations are simpler and can be dealt with analogously.) Assume first that $\langle p_1, h_1 \rangle$ and $\langle p_2, h_2 \rangle$ form two backward deadlocks. If there is no vertex v in $H[h_1, h_2]$ such that the shot $F(v) \in \partial P$ is defined, S_b can simply use his flashlight to clear $H[h_1, h']$ and then aim his flashlight at h_1 , which separates all defining edges of uncleared deadlocks on ∂H from those on ∂P . Next, the searcher S_a can move to clear the rest of the contaminated region. Hence, CC is searchable with two flashlights. Suppose below that v is the vertex of $H[h_1, h_2]$ such that the shot $F(v) \in \partial P$ is the smallest among these forward shots. If $p_2 < F(v) < p(h_1)$ holds, the chain of ∂H from v to the current endpoint of the ray emanating from S_b is weakly visible from $P[F(v), p']$. Also, the chain of ∂P from $F(v)$ to the current position of S_b is weakly visible from $H[v, h']$; otherwise, there is a vertex r in $P[F(v), p']$ such that $F(r) < v$ holds on ∂H and thus r and v form a forward deadlock, which is the case to be handled later. Since $p_2 < F(v)$, no defining vertices of uncleared deadlocks on ∂P are contained in $P[F(v), p']$, and thus S_b can move his flashlight to $\overline{F(v)v}$ using a walk [4]. Since the counter-walk from $\overline{F(v)v}$ to $\overline{p(h_1)h_1}$ is allowed, the flashlight of S_b can further be moved to $\overline{p(h_1)h_1}$. See Fig. 5(a). At that moment, all defining edges of uncleared deadlocks on ∂H are separated from those by the flashlight of S_b . As described above, CC can then be cleared. The remaining case is $p(h_1) < F(v)$. If there are no reflex vertices r in $P[p(h_1), F(v)]$ such that $F(r) < h_1$ holds, the flashlight of S_b can simply be moved to $\overline{p(h_1)h_1}$, which separates all defining edges of uncleared deadlocks on ∂P from those on ∂H , and thus CC is searchable. Otherwise, let r be the largest vertex of $P[p(h_1), F(v)]$ such that $F(r) < h_1$ holds. See Fig. 5(b). As discussed above, the flashlight of S_b can be moved to $\overline{rF(r)}$ using a walk. Again, CC can be cleared. The situation where $\langle p_1, h_1 \rangle$ and $\langle p_2, h_2 \rangle$ form the forward deadlocks can be dealt with analogously.

Consider the situation in which p_1 and p_2 contribute to the different types of deadlocks. Assume first that p_1 (resp. p_2) contributes to a backward (resp. forward) deadlock, but neither $\langle p_1, h_1 \rangle$ nor $\langle p_2, h_2 \rangle$ forms a deadlock. (The symmetric case can be handled analogously.) Since all defining vertices of uncleared deadlocks are contained in $P[p_1, p_2] \cup H[h_1, h_2]$, there is a vertex w in $H[h_1, h_2]$ such that p_1 and w form the backward deadlock. See Fig. 5(c). If $B(w) < p_2$

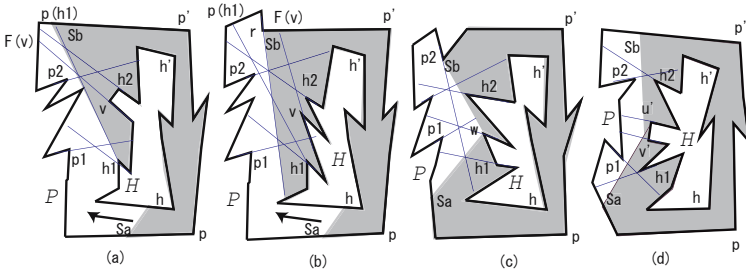


Fig. 5. Illustrating the proof of Lemma 7

holds, then $P[p_2, p']$ is weakly visible from $H[w, h']$. Also, $H[w, h']$ is weakly visible from $P[p_2, p']$; otherwise, there is a vertex v in $H[w, h']$ (to be exact, in $H[w, h_2)$) such that $F(v) < p_2$; in this case, p_2 and v form a forward deadlock, and two deadlocks of $\langle p_1, w \rangle$ and $\langle p_2, v \rangle$ form a pair of BF -deadlocks, a contradiction. As discussed above, the flashlight of S_b can then be moved to $\overline{p_2 w}$ using a walk. Next, S_a moves to aim his flashlight at w , and then, S_b can further move his flashlight, e.g., to $\overline{p_2 F(p_2)}$ in Fig. 5(c), so as to separate the defining edges of uncleared deadlocks. Hence, CC can be cleared with two flashlights. In the case that $B(w) > p_2$, as discussed above, the flashlight of S_b can first be moved to $\overline{B(w)w}$ using a walk. Next, S_a moves to aim his flashlight at w , and S_b can then use his flashlight to separate the defining edges of uncleared deadlocks. Again, CC can be cleared with two flashlights.

The remaining case is that $\langle p_1, h_1 \rangle$ forms a forward deadlock and $\langle p_2, h_2 \rangle$ forms a backward deadlock, with $p_1 < p_2$ and $h_1 < h_2$. (Recall that no pair of BF -deadlocks occurs in CC .) For ease of presentation, assume that $P[F(h_1), B(h_2)]$ is also contained in $P[p, p']$. Let v' denote the smallest vertex of $H[h_1, h']$ such that $F(v') < p_2$. If the vertex v' does not exist, then we let $v' = h_1$. Also, let u' denote the largest vertex of $H[h, h_2]$ such that $p_1 < B(u')$, and if u' does not exist, let $u' = h_2$. See Fig. 5(d) for an example. If both u' and v' exist, then $v' < u'$ holds; otherwise, $H[h_1, h_2]$ is not weakly visible from $P[p_2, p_1]$ and thus the deadlocks formed by $\langle p_1, h_1 \rangle$ and $\langle p_2, h_2 \rangle$ form a pair of FB -deadlocks, a contradiction. Then, $H[h_1, u']$ and $H[v', h_2]$ are weakly visible from $P[p, p_1]$ and $P[p_2, p']$, respectively. By considering v' (resp. u') as h_1 (resp. h_2), we can also define the point $p(v')$ (resp. $p(u')$) on ∂P . The flashlight of S_b (resp. S_a) can then be moved to aim at v' (resp. u'), when $p(v')$ (resp. $p(u')$) is well defined. See Fig. 5(d). Since no pair of FB -deadlocks occurs in CC , at least one of S_a and S_b can further use the ray of his flashlight to separate the defining edges of these deadlocks (Fig. 5(d)). The other 1-searcher (e.g., S_a in Fig. 5(d)) can then move to clear the rest of the contaminated region. Hence, CC can be cleared.

Finally, let us show how to clear, say, a strong backward deadlock in CC . Denote by p, h and h' the defining vertices of the strong backward deadlock. By fixing p , we can assume that the point $B(h)$ (resp. $F(h')$) is the largest (resp. smallest) among the backward (resp. forward) shots from all possible defining vertices h (resp. h'). Suppose that S_a is initially located at $B(h)$ and aims his

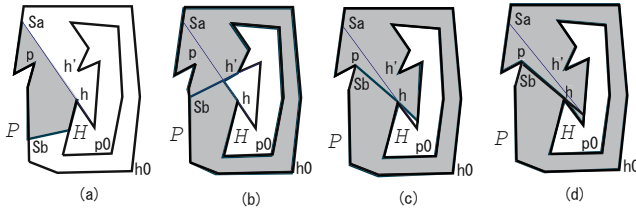


Fig. 6. Illustration for clearing the strong deadlock

flashlight at h , so as to separate $\overline{pSucc(p)}$ from $\overline{hSucc(h)}$ (Fig. 6(a)). The searcher S_b can then walk along ∂P in counterclockwise direction, starting at the same position of S_a , till $F(h')$ is reached for the second time (Fig. 6(b)). Assume that S_b has aimed his flashlight at h' . Since $F(h')$ is the smallest forward shot from all the defining vertices h' , S_b can move to the vertex p so as to clear a part of $H[h, h']$, as large as possible (Fig. 6(c)). Finally, since S_a is located at the largest shot $B(h)$, he can use (or rotate) his flashlight to clear CC (Fig. 6(d)); otherwise, ∂H is not weakly visible from ∂P , a contradiction. This completes the proof. \square

By now, we obtain the main result of this paper.

Theorem 1. *Suppose that ∂P and ∂H of the given circular corridor are mutually weakly visible. Then CC is searchable with two flashlights if and only if there are neither three deadlocks such that any two of them are non-separated, nor a strong deadlock and a simple deadlock in CC .*

5 Algorithms

The following observation immediately follows from the definition of non-separated deadlocks.

Observation 1. *Suppose that $\langle p_2, h_2 \rangle$ and $\langle p_3, h_3 \rangle$ form a pair of non-separated deadlocks, with $p_2 < p_3$ and $h_2 < h_3$. If $\langle p_1, h_1 \rangle$ forms the deadlock of the same type as $\langle p_2, h_2 \rangle$, with $p_1 < p_2$ and $h_1 < h_2$, then $\langle p_1, h_1 \rangle$ and $\langle p_3, h_3 \rangle$ form a pair of non-separated deadlocks, too.*

Assume that $\langle p_1, h_1 \rangle$ and $\langle p_2, h_2 \rangle$ form a pair of BB -deadlocks, with $p_1 < p_2$ and $h_1 < h_2$. If there are no vertices $p \in P(p_1, p_2)$, $h \in H(h_1, h_2)$ such that $\langle p_1, h_1 \rangle$ and $\langle p, h \rangle$ form a pair of BB -deadlocks, the deadlocks formed by $\langle p_1, h_1 \rangle$ and $\langle p_2, h_2 \rangle$ are said to be BB -adjacent. Analogously, we can define the FF -adjacent, BF -adjacent and FB -adjacent deadlocks.

We have by now employed the symbol " $<$ " to indicate the order of boundary points, with respect to a starting point p_0 or h_0 , say, in the clear region. It works well because p_0 or h_0 is not changed in our discussion made in the end (or start) phase. However, the following algorithm for determining the searchability of a

circular corridor requires to traverse ∂P and ∂H several times. More caution is thus needed. For example, a backward deadlock and a forward deadlock may form a pair of BF -deadlocks or a pair of FB -deadlocks, depending on where the starting points are set. To avoid confusions, we do not employ the symbol " $<$ " in the description of our decision algorithm.

Theorem 2. *The searchability of a circular corridor with two flashlights can be determined in $O(n \log n)$ time, where n denotes the total number of vertices of the outer and inner boundaries.*

Proof. Let L denote an internal line segment in CC , which can simply be found in linear time. We then compute in $O(n \log n)$ time all ray shots from reflex vertices of ∂P and ∂H . This can be done by inserting L into CC as two isometric edges and performing the ray shooting queries in the resulting simple polygon [1]. A simple exception is that some ray shots may cross with L in CC , but it can easily be handled. After all ray shots are obtained, the weak visibility between the outer boundary ∂P and the inner boundary ∂H can be verified in linear time [4].

Assume below that ∂P and ∂H are mutually weakly visible. Given two points on ∂P and ∂H respectively, we can find the first deadlock, say, in clockwise direction, if it exists [3]. The time required is linear in the total number of the vertices and ray shots, which are traversed.

Let us now consider how to determine whether a strong deadlock and a simple deadlock exist in CC . First, we find the first backward deadlock, starting at two endpoints of L . Let p and h denote its two defining vertices. Then, continue to traverse on ∂P and ∂H , starting at $B(h)$ and $Succ(h)$, till the next backward deadlock is found. During the traversal, we also determine, for each reflex vertex v encountered on ∂H , whether v , p and h form a strong backward deadlock. It can be done in constant time, as all ray shots and vertices on ∂P have been ordered. If *yes*, we are done. Otherwise, we obtain a new backward deadlock adjacent to that of $\langle p, h \rangle$, and then, perform the same operation for it again. This procedure stops as soon as a strong deadlock is reported, or either of p and h is met again. In this way, a strong backward deadlock can eventually be reported, if it exists. In the case that a strong deadlock is found, we further determine whether a simple deadlock exists. Whether a strong forward deadlock and a simple deadlock exist can be determined analogously. Since it suffices to traverse on ∂H and ∂P a constant time, the time required is $O(n)$.

Consider how to determine whether there are three deadlocks such that any two of them are non-separated, provided that CC does not contain any strong deadlock. (If a strong deadlock occurs, we can then determine whether CC is searchable with two flashlights.) First, we determine whether there are three backward (resp. forward) deadlocks such that any two of them form a pair of BB -deadlocks (resp. FF -deadlocks). If *yes*, CC is not searchable and we are done. Otherwise, as described above, we find a pair of BF -deadlocks such that its two elements are BF -adjacent. Suppose that the two deadlocks giving the pair of BF -deadlocks are found (otherwise, there are no three deadlocks consisting of both backward and forward ones such that any two of them are non-separated,

and CC is thus searchable). Next, we further determine whether two found deadlocks also form a pair of FB -deadlocks. If *no*, all deadlocks can then be cleared in the start phase, and thus CC is searchable. Otherwise, we determine whether there exists the other deadlock such that it together with either found deadlock forms a pair of BB -deadlocks with two BB -adjacent elements or a pair of FF -deadlocks with two FF -adjacent elements. In the case that the pair of BB -deadlocks or FF -deadlocks is found, we further determine whether the third combination among three found deadlocks also gives a pair of FB -deadlocks. It follows from Observation 1 that CC is searchable if and only if either answer is *no*.

The procedure described above can be performed in $O(n)$ time. First, whether two deadlocks, say, given by $\langle p_1, h_1 \rangle$ and $\langle p_2, h_2 \rangle$, form a pair of FB -deadlocks can be determined in $O(n)$ time. This is because whether $H[h_1, h_2]$ is weakly visible from $P[p_2, p_1]$ can be determined in linear time, and whether there are internal segments ph such that \overline{ph} separates the defining edges of two deadlocks can also be determined, after the regions visible from h_1 and h_2 are computed. Note that the operation of determining a pair of FB -deadlocks is performed at most twice in the above procedure. The remaining task is to show that a pair of BB -deadlocks (a pair of FF -deadlocks, or a pair of BF -deadlocks) with two adjacent elements can be computed in linear time. Also, they are performed only a constant time. Similar to the work of computing a strong deadlock, we can make a traversal on ∂P and ∂H to find the required pair of non-separated deadlocks. Due to space limit, we omit the detail.

In conclusion, whether there are three deadlocks such that any two of them are non-separated can be tested in $O(n)$ time, after all ray shots in CC are computed. The proof is complete. \square

Theorem 3. *A search schedule can be reported in time linear in its size, which is $O(n^2)$ in the worst case, if it exists*

Proof. Omitted in this extended abstract. \square

Acknowledgement

The authors would like to thank Prof. Tsunehiko Kameda of Simon Fraser University for valuable comments on a preliminary version of the paper.

References

- [1] Chazelle, B., Guibas, L.: Visibility and intersection problem in plane geometry. *Discrete Comput. Geom.* 4, 551–581 (1989)
- [2] Guibas, L.J., Latombe, J.C., Lavalley, S.M., Lin, D., Motwani, R.: Visibility-based pursuit-evasion in a polygonal environment. *Int. J. Comput. Geom. & Appl.* 9, 471–493 (1999)
- [3] Heffernan, P.J.: An optimal algorithm for the two-guard problem. *IJCGA* 6, 15–44 (1996)

- [4] Icking, C., Klein, R.: The two guards problem. *IJCGA* 2, 257–285 (1992)
- [5] Kameda, T., Zhang, J.Z., Yamashita, M.: Searching a circular corridor by two boundary 1-searchers. In: *Proc. KyotoCGGT 2007* (2007)
- [6] LaValle, S.M., Simov, B., Slutzki, G.: An algorithm for searching a polygonal region with a flashlight. *IJCGA* 12, 87–113 (2002)
- [7] Lee, J.H., Park, S.M., Chwa, K.Y.: Searching a polygonal room with one door by a 1-searcher. *IJCGA* 10, 201–220 (2000)
- [8] Simov, B.H., LaValle, S.M., Slutzki, G.: A complete pursuit-evasion algorithm for two pursuers using beam detection. In: *Proc. IEEE Int'l. Conf. Robotics Automation*, pp. 618–623 (2002)
- [9] Suzuki, I., Yamashita, M.: Searching for mobile intruders in a polygonal region. *SIAM J. Comp.* 21, 863–888 (1992)
- [10] Tan, X.: A unified and efficient solution to the room search problem. *Comput. Geom. Theory Appl.* 40, 45–60 (2008)
- [11] Tan, X.: Searching a polygonal region by a boundary searcher. *J. Comput. Sci. Tech.* (to appear)

On the Complexity of the Multiple Stack TSP, k STSP

Sophie Toulouse and Roberto Wolfler Calvo

LIPN (UMR CNRS 7030) - Institut Galilée, Université Paris 13
99 av. Jean-Baptiste Clément, 93430 Villetaneuse, France
sophie.toulouse@lipn.univ-paris13.fr, wolfler@lipn.univ-paris13.fr

Abstract. Given a universal constant k , the multiple Stack Travelling Salesman Problem (k STSP in short) consists in finding a pickup tour T^1 and a delivery tour T^2 of n items on two distinct graphs. The pickup tour successively stores the items at the top of k containers, whereas the delivery tour successively picks the items at the current top of the containers: thus, the couple of tours are subject to LIFO (“*Last In First Out*”) constraints. This paper aims at finely characterizing the complexity of k STSP in regards to the complexity of TSP. First, we exhibit tractable sub-problems: on the one hand, given two tours T^1 and T^2 , deciding whether T^1 and T^2 are compatible can be done within polynomial time; on the other hand, given an ordering of the n items into the k containers, the optimal tours can also be computed within polynomial time. Note that, to the best of our knowledge, the only family of combinatorial precedence constraints for which constrained TSP has been proven to be in \mathbf{P} is the one of PQ-trees, [2]. Finally, in a more prospective way and having in mind the design of approximation algorithms, we study the relationship between optimal value of different TSP problems and the optimal value of k STSP.

1 Introduction

1.1 The Problem Specification

Assume that a postal operator has to pick up n items in some city 1, and then to deliver the same items in some city 2. Such a situation can be modeled by means of two TSP instances $I^1 = (G^1, d^1)$ and $I^2 = (G^2, d^2)$, where the two graphs $G^1 = (V^1, E^1)$ and $G^2 = (V^2, E^2)$ have the same order $n + 1$ (vertex 0 represents the depot, whereas vertices $[n]$ represent the location where the items have to be picked up or delivered), and the distance functions $d^1 : E^1 \rightarrow \mathbb{N}$, $d^2 : E^2 \rightarrow \mathbb{N}$ associate integer values to the edges of G^1, G^2 . The two TSP instances I^1, I^2 thus represent the search of an optimal pickup tour in city 1, and the search of an optimal delivery tour in city 2, respectively. If no constraint occurs between the two tours, then the problem is equivalent to the resolution of two independent TSP. In k STSP, one assumes that the tours are subject to LIFO constraints, namely: the pickup tour stacks the items into k containers, so that the delivery tour must deliver *at first* the items that have been stored

at last by the pickup tour. Hence, a solution of k STSP consists of a couple of tours (T^1, T^2) , together with a stacking order \mathcal{P} on the k containers that is compatible with both the pickup and the delivery tours. Here, a stacking order is defined as a set $\{P^1, \dots, P^k\}$ of k q^ℓ -uples $P^\ell = (v_1^\ell, \dots, v_{q^\ell}^\ell)$ that partitions $[n]$, where vertices v_1^ℓ and $v_{q^\ell}^\ell$ respectively represent the bottom and the top of the ℓ th stack. A feasible solution (T^1, T^2, \mathcal{P}) is optimal if it is of optimal distance, where the distance is given by the sum $d^1(T^1) + d^2(T^2)$ of the distances of the two tours. For sake of simplicity, we will always consider that G^1 and G^2 are the complete directed graph \mathbf{K}_{n+1} on $V = \{0\} \cup [n]$. In the case of symmetric distance functions, one just has to consider $d^\alpha(u, v) = d^\alpha(v, u)$ for $u, v \in V$ and $\alpha \in \{1, 2\}$; moreover, one could recognize unexisting arcs by associating to each couple of vertices $u, v \in V$ such that $(u, v) \notin E^\alpha$, e.g., the distance $d^\alpha(u, v) = d_{max}^\alpha + 1$, where $d_{max}^\alpha = \max\{d^\alpha(e) \mid e \in E^\alpha\}$.

1.2 Previous Work, Problematic and Outline

By contrast to other TSP problems, only a few literature exists on this problem (see, for example, [3,8] for some heuristic approaches), and none (to the best of our knowledge) about its complexity. Anyway, the problem we address naturally is **NP – hard**, from TSP.

Nevertheless, one could wonder about what combinatorial structure of k STSP impacts more on its complexity: the stacks (and the LIFO constraints they induce on the tours), or the permutations themselves? The answer is not clear and this paper shows why. First of all we prove that, given two tours T^1 and T^2 , deciding whether T^1 and T^2 are compatible or not is tractable, since the decision reduces to k -coloring in comparability graphs. Moreover, given an ordering of the n items into the k stacks, the optimal tours can also be computed within polynomial time, by means of dynamic programming. Another interesting question concerns the relative complexity of k STSP in regards to TSP: how much its trickier combinatorial structure makes k STSP harder to solve (exactly as well as approximatively) than TSP? Although we do not provide formal answers to this latter question, we give some intuition that k STSP is globally harder than general TSP to optimize: it is obvious that efficient algorithms for k STSP can be derived in order to solve TSP; by contrast, we establish that tours of good quality for TSP may lead to arbitrary low quality solutions for STSP.

The paper is organized as follows: we first expose in section 2 some notations and properties that will be useful for the next sections; section 3 exhibits two tractable sub-problems (one of decision when the tours are given, 3.2, one of optimization when the stacks are given, 3.3); section 4 compares the behaviour of solution values in STSP instances and some related TSP instances, bringing to the fore that good resolution of TSP may not lead to good resolution of STSP; finally, section 5 concludes with some perspectives.

2 Preliminaries

Three strict orders $<^1, <^2, <^3$ on $[n]$ are associated, respectively, to the pickup tour $T^1 = (0, u_1^1, \dots, u_n^1, 0)$, to the delivery tour $T^2 = (0, u_1^2, \dots, u_n^2, 0)$ and to a stacking order $\mathcal{P} = \{P^1, \dots, P^k\}$. The two orders $<^1, <^2$ are complete whereas $<^3$ is partial. It means that $\forall a \neq b \in [n]$ we can write:

$<^1:$	$a <^1 b$	$\Leftrightarrow T^1$ picks up a before b
$<^2:$	$a <^2 b$	$\Leftrightarrow T^2$ delivers b before a
$<^3:$	$a <^3 b$	$\Leftrightarrow \exists \ell \in [1, k] / a, b \in P^\ell, a$ is stacked before b in P^ℓ
	$\neg(a <^3 b) \wedge \neg(a >^3 b)$	$\Leftrightarrow a, b$ are stacked into two distinct stacks

Lemma 1. *A solution (T^1, T^2, \mathcal{P}) is feasible iff the three orders $<^1, <^2, <^3$ it induces on $[n]$ satisfy the following conditions:*

$$\forall a \neq b \in [n], a <^1 b \Rightarrow \neg(a >^3 b) \tag{1}$$

$$\forall a \neq b \in [n], a <^2 b \Rightarrow \neg(a >^3 b) \tag{2}$$

Proof. The necessary condition is obvious. For the sufficient condition, let consider a pickup tour T^1 and a stacking order \mathcal{P} (the argument is rather similar for the delivery tour). For any $i \in [n]$, T^1 has to pick up the item u_i^1 , that is of index j in some stack P^ℓ ; this is possible iff the previous item that T^1 has picked up in P^ℓ is the item u_{j-1}^ℓ , or there is no such index and $j = 1$, what is always true if T^1 and \mathcal{P} verify condition **(1)**. □

3 Complexity Classes and Properties

3.1 Global Complexity

The problem obviously is **NP – hard**, for arbitrary instances $(G^1, d^1; G^2, d^2)$ of k STSP (where by “arbitrary”, we mean that we do not make any assumption, neither on the graph completeness, nor on the symmetry of the distance functions). When the distance functions d^1 and d^2 are the same, up to the arc direction (that is, $d^1(a, b) = d^2(b, a)$ for all $a, b \in \{0\} \cup [n]$), it is equivalent to the regular TSP (consider on the one hand that $T^1 = (0, u_1, \dots, u_n, 0)$ is an optimal pickup tour iff $T^2 = (0, u_n, \dots, u_1, 0)$ is an optimal delivery tour, on the other hand that every stacking order that is feasible for T^1 also is feasible for T^2). Second, for a given triple (T^1, T^2, \mathcal{P}) where T^1, T^2 are two tours on $\{0\} \cup [n]$ and \mathcal{P} is a stacking order of $[n]$ using k stacks, checking whether (T^1, T^2, \mathcal{P}) is feasible or not can be done within linear time (quite immediate from Lemma **(1)**).

3.2 Deciding Feasibility for a Couple of Tours

Let us denote by $G^\neq = (V^\neq, E^\neq)$ the graph induced by the set of pairs $\{a, b\}$ such that the two orders $<^1$ and $<^2$ are discordant:

$$E^\neq = \{ \{a, b\} \mid a \neq b \in [n], a <^1 b \wedge a >^2 b \}, \quad V^\neq = \bigcup_{\{a, b\} \in E^\neq} \{a, b\}$$

Lemma 2. *Given two tours T^1, T^2 , a compatible stacking order \mathcal{P} exists iff $\chi(G^\neq) \leq k$, where $\chi(G^\neq)$ denotes the chromatic number on G^\neq .*

Proof. For the necessary condition, consider a feasible solution (T^1, T^2, \mathcal{P}) and two items $a \neq b$ such that $\{a, b\} \in E^\neq$, iff $a <^1 \wedge a >^2 b$, or $a >^1 b \wedge a <^2 b$. In both cases, we know from Lemma 1 that $\neg(a >^3 b) \wedge \neg(a >^3 b)$; thus, the k stacks in \mathcal{P} correspond to k independent sets in G^\neq . For the sufficient condition, we build from a k -coloring on V^\neq a stacking order \mathcal{P} that fulfills, together with T^1 and T^2 , conditions (1) and (2) of Lemma 1. We first fill each stack P^ℓ with the items of the ℓ th color set $\{v_1^\ell, \dots, v_{q^\ell}^\ell\}$, by considering on P^ℓ the order induced by the relation $<^{1,2}$ defined as: $<^{1,2} = <^1 \wedge <^2$ (the two orders do coincide on each color ℓ). It remains to insert into the stacks the items from $[n] \setminus V^\neq$. The orders $<^1$ and $<^2$ also coincide on $([n] \setminus V^\neq) \times [n]$; we can therefore write $[n] \setminus V^\neq = (v_1, \dots, v_r)$ with $v_1 <^{1,2} \dots <^{1,2} v_r$. For index i from 1 to r , we insert v_i in position $j(v_i) + 1$ in P^1 iff $j(v_i)$ is the current maximum index j such that $v_j^1 <^{1,2} v_i$, if such an index exists; otherwise, $j(v_i) = 0$ (in any case, indices in P^1 are updated after each insertion). We finally obtain a partition of $[n]$ within a set of k stacks such that in every stack, the elements are ordered with respect to $<^{1,2}$, what fulfills conditions (1) and (2). \square

Graph coloring problems (in general, but also k -coloring for a universal constant $k \geq 3$) are known to be **NP - c** (see, e.g., [4]); nevertheless, it turns out that G^\neq belongs to the class of perfect graphs, for which determining $\chi(G)$ is in **P**, [6]. Hence, the considered decision problem is tractable, for any k (and this even if k is not any longer considered as a universal constant, but as being part of the input).

Theorem 1. *The STSP sub-problem that consists, given a couple (T^1, T^2) , in deciding whether there exists or not a compatible stacking order, is in **P**.*

Proof. E^\neq represents the pairs $\{a, b\}$ of $[n]$ such that $(a <^1 b \wedge a >^2 b)$ or $(a >^1 b \wedge a <^2 b)$, where we recall that $<^1$ and $<^2$ both totally order $[n]$. Therefore, G^\neq is a comparability graph. Indeed, consider the set of arcs $F^\neq = \{(a, b) \in [n] \times [n] \mid a <^1 b \wedge a >^2 b\}$: (i) for all $a \neq b \in V^\neq$, $(a, b) \in F^\neq \vee (b, a) \in F^\neq$ iff $\{a, b\} \in E^\neq$; (ii) for all $a \neq b \in V^\neq$, $(a, b) \in F^\neq \Rightarrow (a, b) \notin F^\neq$; (iii) for all distinct $a, b, c \in V^\neq$, $(a, b) \in F^\neq$ and $(b, c) \in F^\neq$ iff $a <^1 b <^1 c \wedge a >^2 b >^2 c$ and thus, $(a, c) \in F^\neq$. Then, F^\neq defines a transitive orientation of the edge set E^\neq , and G^\neq is a comparability graph. By the way, note that a comparability graph $G = (V, E)$ may represent the conflict graph of some couple or orders $(<^1, <^2)$ iff its complementary graph $\overline{G} = (V, \overline{E})$ also is a comparability graph (representing $<^1 \wedge <^2$).

Algorithm 1 is a polynomial time procedure that, given a couple of tours (T^1, T^2) , responses NO if this couple is unfeasible, returns a compatible stacking order \mathcal{P} otherwise. \square

3.3 Optimizing the Tours When the Stacks Are Given

In this section, we prove that it is a tractable problem to compute the optimal tours, when the stacks are fixed.

Algorithm 1. STACKING FROM T^1 AND T^2

Input. T^1 a pickup tour, T^2 a delivery tour, k the number of stacks.
Output. A compatible stacking order \mathcal{P} iff (T^1, T^2) is feasible.

for $\ell = 1$ **to** k **do** $P^\ell \leftarrow \emptyset$;
 build $G^\# = (V^\#, E^\#)$ from T^1, T^2 ;
 // Coloring stage
for each $v \in [n] \setminus V^\#$ **do** $C(v) \leftarrow 1$;
 compute a minimum coloring $(C : V^\# \rightarrow [\chi(G^\#)], v \mapsto C(v))$ on $G^\#$;
if $\chi(G^\#) > k$ **then return** NO;
 // Stacking stage (done according to \prec^1)
for $i = 1$ **to** n **do** $\{ \ell \leftarrow C(u_i^1); \text{stack } u_i^1 \text{ into } P^\ell \}$;
return $\mathcal{P} = \{P^1, \dots, P^k\}$;

Theorem 2. For a given stacking order \mathcal{P} , one can find an optimal pickup tour and an optimal delivery tour within polynomial time (but exponential in k).

Proof. We only present the argument for the pickup tour (the proof being rather similar for the delivery tour). Let $\mathcal{P} = P^1, \dots, P^k$ where $P^\ell = (v_1^\ell, \dots, v_{q^\ell}^\ell)$ for any ℓ be a stacking order. A pickup tour that is compatible with \mathcal{P} starts by picking up the items which must be placed at the bottom of each stack, until all the stacks have been completely read. Hence, we will consider the space of states $\mathcal{S} = \times_{\ell=1}^k [0, q^\ell]$, where a state $e = (e^1, \dots, e^k) \in \mathcal{S}$ represents the set of items on each stack that have already been picked up. Therefore, $e^\ell = h$ means that items $v_1^\ell, \dots, v_h^\ell$ have been collected, and that the current bottom (that is, its $(h + 1)$ th element v_{h+1}^ℓ) is the next item in P^ℓ that has to be picked up. Let denote with $W(e) = \cup_{\ell=1}^k \{v_1^\ell, \dots, v_{e^\ell}^\ell\}$ the set of collected items once the state e has been reached. Although there are (in general) an exponential number of paths to reach a given state e , there are only (at most) k possible preceding states, depending on which stack has been considered at last. Hence, we associate to each state e its list of possible predecessors $p(e, 1), \dots, p(e, k)$, where $p(e, \ell) = (e^1, \dots, e^{\ell-1}, e^\ell - 1, e^{\ell+1}, \dots, e^k)$, for e and ℓ such that $e^\ell \geq 1$.

In order to build an optimal tour, we associate to each state e a collection of k labels that correspond to its cost. For any $e \neq (0, \dots, 0) \in \mathcal{S}$, and for any $\ell \in [k]$, the label $\mathcal{E}(e, \ell)$ gives the minimum cost for picking up all the items of $W(e)$ starting from 0, compatible with \mathcal{P} , and that end with P^ℓ . According to this definition, $\mathcal{E}(e, \ell)$ may only depend on $\mathcal{E}(p(e, \ell), \ell')$, for $\ell' \in [k]$: the current sub-tour T^1 may reach e after having picked up $v_{e^\ell}^\ell$ iff $v_{e^\ell}^\ell$ has not been picked up yet. Then, for any $e \neq (0, \dots, 0), (q^1, \dots, q^k) \in \times_{\ell=1}^k [0, q^\ell]$ and for any $\ell \in [k]$, we have the following recurrence relation:

$$\mathcal{E}(e, \ell) = \begin{cases} +\infty & \text{if } e^\ell = 0 \\ \min_{\ell'=1}^k \{ \mathcal{E}(p(e, \ell), \ell') + d^1(v_{p(e, \ell)\ell'}^{\ell'}, v_{e^\ell}^\ell) \mid p(e, \ell')^\ell \geq 1 \} & \text{if } e^\ell \geq 1 \end{cases}$$

Note that item $v_{p(e,\ell)\ell'}^{\ell'}$ differs from $v_{e\ell'}^{\ell'}$ (that is, $p(e,\ell)^{\ell'}$ differs from $e^{\ell'}$) iff $\ell' = \ell$. The initial conditions are given by the k states $f(\ell) = (0, \dots, 0, 1, 0, \dots, 0)$ that correspond to the ℓ th canonical vectors:

$$\mathcal{E}(f(\ell), \ell') = \{ +\infty \text{ if } \ell' \neq \ell, d^1(0, v_1^\ell) \text{ if } \ell' = \ell \}$$

Finally, the expression of the labels on the final state $F = (q^1, \dots, q^k)$ is the following (for ℓ such that $F^\ell \geq 1$):

$$\mathcal{E}(F, \ell) = \min_{\ell'=1}^k \{ \mathcal{E}(p(F, \ell), \ell') + d^1(v_{p(F,\ell)\ell'}^{\ell'}, v_{q^\ell}^\ell) + d^1(v_{q^\ell}^\ell, 0) \mid p(F, \ell)^{\ell'} \geq 1 \}$$

Algorithm 2. optimal PICKUP TOUR(\mathcal{P})

Input. $I = (d^1, d^2)$ an instance of k STSP, \mathcal{P} a stacking order on I .

Output. An optimal pickup tour T^1 that is compatible with \mathcal{P} .

// Initialization stage

for $e \in \mathcal{S}$, $\ell \in [k]$ s.t. $e^\ell = 0$ do $\mathcal{E}(e, \ell) \leftarrow +\infty$;

for $\ell \in [k]$ do $\mathcal{E}(f(\ell), \ell) \leftarrow d^1(0, v_1^\ell)$;

// Dynamic procedure

for $p = 2$ to $n - 1$ do

for $e \in \mathcal{S}$ s.t. $|e| = p$ do

for $\ell = 1$ to k s.t. $e^\ell \geq 1$ do

$\mathcal{E}(e, \ell) \leftarrow \min_{\ell'=1}^k \{ \mathcal{E}(p(e, \ell), \ell') + d^1(v_{p(e,\ell)\ell'}^{\ell'}, v_{e^\ell}^\ell) \mid p(e, \ell)^{\ell'} \geq 1 \}$;

// Termination

for $\ell = 1$ to k s.t. $F^\ell \geq 1$ do

$\mathcal{E}(F, \ell) \leftarrow \min_{\ell'=1}^k \{ \mathcal{E}(p(F, \ell), \ell') + d^1(v_{p(F,\ell)\ell'}^{\ell'}, v_{F^\ell}^\ell) + d^1(v_{F^\ell}^\ell, 0) \mid p(F, \ell)^{\ell'} \geq 1 \}$;

return T^1 the tour associated with the label $\arg \min \{ \mathcal{E}(F, q^\ell) \mid \ell = 1, \dots, k \}$;

The optimal value is the minimal cost among $\mathcal{E}(F, 1), \dots, \mathcal{E}(F, k)$, since any feasible pickup tour must end with the top of some stack. Furthermore, the recurrence relation indicates that the labels of a state e (including F) such that $|e| = p$ (where $|\cdot|$ denotes the Hamming norm) only depend on a subset of the states e' such that $|e'| = p - 1$. Based on these observations, Algorithm 2 computes an optimal pickup tour within polynomial time. The number of states to consider is upper bounded by $(n+1)^k - 1$ (the worst configuration occurs when the items are fairly distributed in the k stacks). The computation of the k labels of a given state requires at worst k^2 comparisons and the global complexity is $\mathcal{O}((n+1)^k)$. Note that the computation of an optimal delivery tour is perfectly symmetric, considering the reverse order on each stack. □

4 Evaluating Optimal k STSP vs. Optimal TSP

In this section, we discuss relationships between solution values of the two TSP tours and the optimal value of the k STSP. For a given instance $I = (d^1, d^2)$ of the

k STSP, we define the two instances $I^1 = (K_{n+1}, d^1)$ and $I^2 = (K_{n+1}, d^2)$ of the TSP. The optimal values (*resp.*, the worst solution values) on I^1, I^2 (for the TSP) and I (for the k STSP) are respectively denoted by $\text{opt}_{TSP}(I^1), \text{opt}_{TSP}(I^2)$ and $\text{opt}_{kSTSP}(I)$ (*resp.*, by $\text{wor}_{TSP}(I^1), \text{wor}_{TSP}(I^2)$ and $\text{wor}_{kSTSP}(I)$). For any I , these extremal values obviously verify relations (3) and (4). Any feasible couple (T^1, T^2) for the k STSP is feasible for the couple of TSP instances (I^1, I^2) . For any tour T^1 on I^1 (*resp.*, T^2 on I^2), there exists a compatible tour T^2 (*resp.*, T^1). Note that for this latter fact (and thus, for relation (5)), we must assume that the underlying graphs are complete.

$$\text{opt}_{TSP}(I^1) + \text{opt}_{TSP}(I^2) \leq \text{opt}_{kSTSP}(I) \tag{3}$$

$$\text{wor}_{kSTSP}(I) \leq \text{wor}_{TSP}(I^1) + \text{wor}_{TSP}(I^2) \tag{4}$$

$$\text{opt}_{kSTSP}(I) \leq \left\{ \begin{array}{l} \text{opt}_{TSP}(I^1) + \text{wor}_{TSP}(I^2) \\ \text{wor}_{TSP}(I^1) + \text{opt}_{TSP}(I^2) \end{array} \right\} \leq \text{wor}_{kTSP}(I) \tag{5}$$

In particular we discuss the results obtained by two simple heuristics denoted here after TWS and TWD and based on the idea of solving to optimality a single TSP. TWS builds a solution of the k STSP, by solving to optimality the delivery tour, while the pickup tour is fixed (or the reverse). TWD determines a solution for the k STSP, by solving to optimality a single stack TSP on a graph whose distance function is obtained by summing up the original distance functions. We prove that both TWS and TWD give an unbounded error, when particular instances families are considered. Let's introduce some more notations. Given the TSP solutions T'^1, T'^2 for I^1, I^2 , we denote by $\text{opt}_{2STSP|T'^1}$ (*resp.* $\text{opt}_{2STSP|T'^2}$) the best solution value for the k STSP on I , among the solutions (T^1, T^2, \mathcal{P}) where $T^1 = T'^1$ (*resp.*, $T^2 = T'^2$). The optimal TSP tours on I^1, I^2 are denoted by $T^{1,*}, T^{2,*}$, respectively. Moreover, for a given $\alpha \in]0, 1[$, we denote by I_α the TSP instance on K_{n+1} with distance function $d_\alpha = 2(\alpha d^1 + (1 - \alpha)(d^2)^{-1})$, where $(d^2)^{-1}$ is defined as $(d^2)^{-1}(a, b) = d^2(a, b)$ for any couple (a, b) of items. The optimal tour on I_α will be denoted by T_α^* .

Lemma 3. *We consider arbitrary distance functions d^1, d^2 (symmetric or not).*

1. *For $a \in \{1, 2\}$, the optimal value $\text{opt}_{2STSP|T^{a,*}}(I)$ verifies:*

$$\inf_{I \in I_{2STSP}} \text{opt}_{2STSP|T^{a,*}}(I) / \text{opt}_{2STSP}(I) = +\infty$$

2. *For $\alpha = 1/2$, the quantities $d^1(T_\alpha^*) + d^2(T_\alpha^*)$ and $\text{opt}_{2STSP}(I)$ verify:*

$$\inf_{I \in I_{2STSP}} (d^1(T_\alpha^*) + d^2(T_\alpha^*)) / \text{opt}_{2STSP}(I) = +\infty$$

Proof. (3),(4) the asymmetric case. Consider the instance family $(I_n)_{n \geq 3}$, $I_n = (d_n^1, d_n^2)$, defined as (indexes are taken modulo $n + 1$):

$$d_n^1(u, v) = \begin{cases} 1 & \text{if } v = u + 1 \\ 1 + \varepsilon & \text{otherwise} \end{cases} \quad d_n^2(u, v) = \begin{cases} 1 & \text{if } v = u + 1 \\ n & \text{otherwise} \end{cases}$$

The optimal values for TSP on I_n^1 and I_n^2 both are $n + 1$, reached by the tours $T_n^{1,*} = T_n^{2,*} = (0, 1, 2, \dots, n, 0)$. If we fix $T_n^{1'} = T_n^{1,*}$, then any stacking order $\mathcal{P}_n = \{P_n^1, P_n^2\}$ that is compatible with $T_n^{1'}$ will order the items in such a way that contradicts the order induced by the optimal delivery tour. In order to evaluate the best possible compatible T_n^2 , we consider three cases:

- Case $(0, 1) \in T_n^2$: $1 >^2 u \forall u \neq 1 \in [n], 1 <^1 u \forall u \neq 1$ and thus, item 1 is non comparable to any other item under $<^3$; hence, $P_n^a = (1), P_n^{3-a} = (2, \dots, n)$ (for $a \in \{1, 2\}$), $T_n^2 = (0, 1, n, n - 1, \dots, 3, 2, 0)$ and $d_n^2(T_n^2) = 1 + n^2$.
- Case $(n, 0) \in T_n^2$: by means of a similar argument, we obtain $P_n^a = (n), P_n^{3-a} = (1, \dots, n - 1), T_n^2 = (0, n - 1, n - 2, \dots, 2, 1, n, 0)$ and $d_n^2(T_n^2) = 1 + n^2$.
- Case $(0, 1), (n, 0) \notin T_n^2$: consider any item $u \in [n] \setminus \{1, n\}$, and assume $(u - 1, u), (u, u + 1) \in T_n^2$; we then deduce from $u - 1 >^2 u >^2 u + 1$ and $u - 1 <^1 u <^1 u + 1$ that items $u - 1, u$ and $u + 1$ are pairwise non comparable under $<^3$, what is not possible for $k = 2$. Hence, T_n^2 uses at least one arc (*resp.*, exactly 2 arcs) of distance n per item $v \in [n]$ (*resp.*, for the depot 0) and thus, $d_n^2(T_n^2) \geq 1/2(n(n + 1) + 2n) = n(n + 3)/2$.

Moreover, the following solution is optimal for I_n , of value $2(n + 1) + (\lfloor (n + 1)/2 \rfloor + 1)\varepsilon$; it consists in stacking items of odd and even values by decreasing order into separated containers, which enables to use the delivery tour $T_n^2 = T_n^{2,*}$, while putting into T_n^1 a maximum number of arcs of kind $(u, u + 1)$ (figure 3 illustrates instance I_n for even value of n):

$$\begin{aligned} P_n^1 &= (n, n - 2, \dots, n - 2i, \dots, n - 2\lfloor (n - 1)/2 \rfloor) \\ P_n^2 &= (n - 1, n - 3, \dots, n - (2i + 1), \dots, n - (2\lfloor (n - 2)/2 \rfloor + 1)) \\ T_n &= \{(0, n - 1)\} \cup \{(n - (2i + 1), n - 2i, n - (2i + 3)) \mid i = 0, \dots, \lfloor (n - 4)/2 \rfloor\} \\ T_n^1 &= \mathcal{T} \cup \{(2, 0)\} \text{ if } n \text{ even, } \mathcal{T} \cup \{(3, 1), (1, 0)\} \text{ if } n \text{ odd} \\ T_n^2 &= (0, 1, \dots, n, 0) \end{aligned}$$

We thus get the expected result (note by the way that, since $\text{opt}_{TSP}(I_n^1) + \text{wor}_{TSP}(I_n^2) = (n + 1)^2$, the ratio $\text{opt}_{2STSP|T_n^1}(I_n)/(\text{opt}_{TSP}(I_n^1) + \text{wor}_{TSP}(I_n^2))$ is asymptotically $1/2$):

$$\left. \begin{aligned} \text{opt}_{2STSP|T_n^1}(I_n) &\geq (n + 1) + n(n + 3)/2 \\ \text{opt}_{2STSP}(I_n) &\leq (n + 1)(2 + \varepsilon) \end{aligned} \right\} \Rightarrow \frac{\text{opt}_{2STSP|T_n^1}(I_n)}{\text{opt}_{2STSP}(I_n)} \xrightarrow{n \rightarrow +\infty} +\infty$$

3.1 the symmetric case. Consider $(J_n)_{n \geq 6}, J_n = (d_n^1, d_n^2)$, defined as (see figure 2 for an illustration):

$$d_n^1(u, v) = \begin{cases} 1 & \text{if } v = u \pm 1 \\ 1 + \varepsilon & \text{otherwise} \end{cases} \quad d_n^2(u, v) = \begin{cases} 1 & \text{if } v + u \in \{n, n + 1\} \\ n & \text{otherwise} \end{cases}$$

The tours $T_n^{1,*} = (0, 1, 2, \dots, n, 0)$ and $T_n^{2,*} = (0, n, 1, n - 1, 2, n - 2, 3, \dots, \lceil n/2 \rceil, 0)$ are optimal on J_n^1, J_n^2 , of value $n + 1$ and $2n$, respectively. Similarly to the asymmetric case, we show that, for items $u = 2, \dots, n - 1$ such that $2u \notin \{n - 1, n, n + 1, n + 2\}$, any tour T_n^2 that is compatible with $T_n^{1'} = T_n^{1,*}$ cannot use the whole (undirected) sequence 1:

¹ The proof is not provided here, please contact the authors for further information.

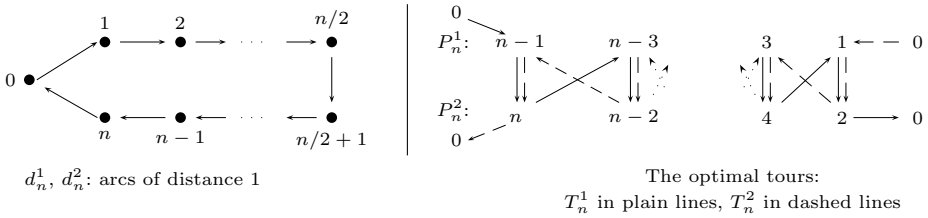


Fig. 1. Instance I_n for n even

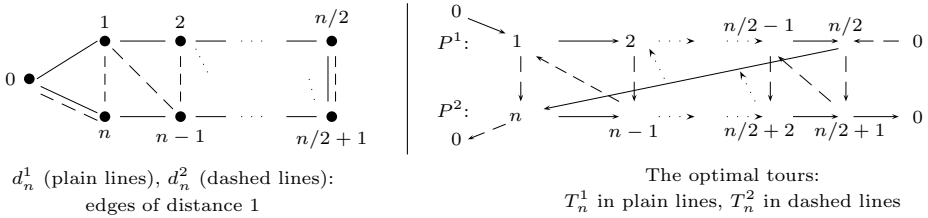


Fig. 2. Instance J_n for n even

$$\{u + 1, n - u, u, n + 1 - u, u - 1\}$$

Hence, $\text{opt}_{2STSP|T_n^1}(J_n) \geq (n + 1) + ((n - 4)(3 + n) + 5 * 4) / 4 \geq n^2 / 4$, whereas the following solution is of value $3n + 2\varepsilon - 1$ (case n even):

$$\begin{aligned} P_n^1 &= (1, 2, 3, \dots, n/2), & P_n^2 &= (n, n - 1, n - 2, \dots, n/2 + 1) \\ T_n^1 &= (0, 1, 2, \dots, n/2; n, n - 1, \dots, n/2 + 2, n/2 + 1; 0) \\ T_n^2 &= (0, n, 1, n - 1, 2, \dots, n/2 - 1, n/2 + 1, n/2; 0) \end{aligned}$$

3.2 Consider the following symmetric instance family $(H_n)_{n \geq 3}$:

condition	$d_n^1(u, v)$	$d_n^2(u, v)$	$d_n^1(u, v) + d_n^2(u, v)$
if $v = u \pm 1$	1	n	$n + 1$
else if $v = u \pm 2$	1	$n + 1$	$n + 2$
else if $u + v \in \{n + 1, n + 3\}$	$n + 1$	1	$n + 2$
else	$n + 1$	$n + 1$	$2n + 2$

The tour $T_{n,1/2}^* = (0, 1, 2, \dots, n, 0)$ that is optimal for the aggregate distance function is of value $(n + 1)^2$, whereas there exists a couple $(T_n^{1,*}, T_n^{2,*})$ of compatible optimal tours with $d_n^1(T_n^{1,*}) = n + 1$ and $d_n^2(T_n^{2,*}) = (n - 3) + 5(n + 1)$ (for n even) or $d_n^2(T_n^{2,*}) = (n - 4) + 6(n + 1)$ (for n odd); hence, the following solution of 2STSP is optimal, of value in $\{7n + 2, 8n + 2\}$ (case n even):

$$\begin{aligned} P_n^1 &= (1, 3, 5, \dots, n - 3, n - 1), & P_n^2 &= (n, n - 2, n - 4, \dots, 4, 2) \\ T_n^1 &= (0; 1, 3, \dots, n - 3, n - 1; n, n - 2, \dots, 4, 2, 0) \\ T_n^2 &= (0; 1, n, 3, n - 2, 5, \dots, 6, n - 3, 4, n - 1, 2; 0) \end{aligned}$$

Note that there exist simpler instance families verifying that $d_{n,1/2}(T_{n,1/2}^*)$ is arbitrarily large *vs.* opt_{2STSP} ; however, for more relevancy, we built $(H_n)_{(n)}$ in such a way that $T_{n,1/2}^*$ and the couple $(T_n^{1,*}, T_n^{2,*})$ are not compatible. \square

5 Conclusion

This paper address the time complexity of k STSP, whose highly combinatorial structure suggests the search of approximation algorithms (may be the most likely for the differential ratio, [7]). The good complexity of its sub-problems makes relevant the design of exact methods based on constraints decomposition of k STSP. Finally, it would be interesting to better characterize the shape of precedence constraints for which TSP/sequencing under precedence constraints are tractable. Indeed, we deduce from Theorem 2 that TSP under “*stack precedence constraints*” is in \mathbb{P} . Equivalently, the single machine scheduling with sequence-dependent time or cost setup under the same shape of constraints, that we denote by $(1/k\text{-stack}, p, ST_{sd}/C_{max}) \equiv (1/k\text{-stack}, p, ST_{sd}/TST)$ and $(1/k\text{-stack}, p, SC_{sd}/TSC)$, are tractable (for the notations used in order to represent the $\alpha/\beta/\gamma$ -classification, [5] of scheduling problems, we refer to [1]). Here, by “*stack precedence constraints*”, we mean that the constraints define a partial order on $[n]$ within at most k ordered subsets, where k is a universal constant.

References

1. Allahverdi, A., Gupta, J.N.D., Aldowaisan, T.: A review of scheduling research involving setup considerations. *Omega* 27(2), 219–239 (1999)
2. Burkard, R.E., Deineko, V.G., Woeginger, G.J.: The Travelling Salesman and the PQ-Tree. In: Cunningham, W.H., Queyranne, M., McCormick, S.T. (eds.) IPCO 1996. LNCS, vol. 1084, pp. 490–504. Springer, Heidelberg (1996)
3. Felipe, A., Ortuño, M., Tirado, G.: Neighborhood structures to solve the double TSP with multiple stacks using local search. In: Proc. of FLINS 2008 (2008)
4. Garey, M.R., Johnson, D.S.: Computers and intractability: a guide to the theory of NP-completeness. Freeman, CA (1979)
5. Graham, R.L., Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G.: Optimization and approximation in deterministic sequencing and scheduling: a survey. *Ann. of Discrete Math.* 5, 287–326 (1979)
6. Grötschel, M., Lovász, L., Schrijver, A.: The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica* 1(2), 169–197 (1981)
7. Monnot, J.: Differential approximation results for the Traveling Salesman and related problems. *Information Processing Letters* 82(5), 229–235 (2002)
8. Petersen, H.L., Madsen, O.B.G.: The double travelling salesman problem with multiple stacks - Formulation and heuristic solution approaches. *EJOR* (2008) (in press)

Linear Programming Based Approximation Algorithms for Feedback Set Problems in Bipartite Tournaments

Anke van Zuylen

Institute for Theoretical Computer Science, Tsinghua University, Beijing, China

Abstract. We consider the feedback vertex set and feedback arc set problems in bipartite tournaments. We improve on recent results by giving a 2-approximation algorithm for the feedback vertex set problem. We show that this result is the best we can attain when using a certain linear program as the lower bound on the optimal value. For the feedback arc set problem in bipartite tournaments, we show that a recent 4-approximation algorithm proposed by Gupta [5,6] is incorrect. We give an alternative 4-approximation algorithm based on an algorithm for feedback arc set in (regular) tournaments in [10,11].

1 Introduction

We consider the feedback vertex set problem and the feedback arc set problem on bipartite tournaments. The feedback vertex set problem on a directed graph $G = (V, A)$ asks for a set of vertices V' of minimum size such that the subgraph of G induced by $V \setminus V'$ is acyclic. The feedback arc set problem on G asks for a set of arcs A' of minimum size such that $(V, A \setminus A')$ is acyclic. A bipartite tournament is an orientation of a complete bipartite graph.

The feedback vertex set problem and the feedback arc set problem are equivalent on general directed graphs: given a directed graph $G = (V, A)$ we can create a graph G' which has a vertex for every arc in A , and an arc from vertex (u, v) to vertex (v, w) . A directed cycle in G corresponds to a directed cycle in G' and vice versa; hence a feedback vertex set in G' corresponds to a feedback arc set in G , and a feedback arc set in G' corresponds to a feedback vertex set in G . The feedback arc/vertex set problem in general graphs is APX-hard [7], and can be approximated to within $\log |V| \log \log |V|$ [3,9]. On bipartite tournaments, the problems are no longer equivalent, since if G is a bipartite tournament, then the graph G' , as defined above, is bipartite but not necessarily a bipartite tournament and vice versa. However, both problems were shown to be NP-hard on bipartite tournaments as well [4,2].

Cai, Deng and Zang [2] study a certain linear programming relaxation of the feedback vertex set problem in bipartite tournaments. They characterize certain small “forbidden subgraphs”, and show that for an instance which does not contain such a subgraph, the linear program is totally dual integral: both the linear program and its dual have integer optimal solutions. Their work also

implies a 3.5-approximation algorithm for the feedback vertex set problem in bipartite tournaments. Prashant [8] recently improved this result by giving a 3-approximation algorithm for feedback vertex set in bipartite tournaments. He uses the optimal solution to a linear programming relaxation of the feedback vertex set problem, and shows that one can iteratively round variables that are $\geq \frac{1}{3}$, until one obtains a feasible integer solution of cost at most 3 times the cost of the linear program. Gupta [5,6] claims a (randomized) 4-approximation algorithm for feedback arc set in bipartite tournaments, by adapting the approach of Ailon, Charikar and Newman [1]. She shows that one can obtain a deterministic algorithm with the same guarantee, by using the optimal solution to a linear programming relaxation and the ideas in [10].

In this paper, we start by giving an alternative method for rounding the linear program for feedback vertex set in bipartite tournaments used by Prashant [8] which also gives an integer solution that costs at most 3 times the optimal value of the respective linear program. Our algorithm simply rounds up the variables that are at least $\frac{1}{2}$ plus all variables that are strictly greater than 0 and that correspond to vertices “on the left” in the bipartite tournament. Our algorithm and its analysis immediately suggest two improvements. First, we could also round up the variables that are strictly greater than 0 that correspond to vertices “on the right”. We show that taking the better of these two rounded solutions yields a $\frac{5}{2}$ -approximation algorithm. Our second improvement uses iterated rounding, where we solve the linear program, round up the variables that are greater than $\frac{1}{2}$, formulate a new linear program, and repeat. At some point, all variables are either 1 or less than $\frac{1}{2}$. Once this condition is reached, we show how to round the remaining solution and bound the cost against the dual solution to get a 2-approximation algorithm. We show that this result is tight for the linear program under consideration: we demonstrate an example with integrality gap 2, hence one cannot obtain a better approximation algorithm by using the lower bound given by the linear program.

Next, we consider the feedback arc set problem in bipartite tournaments. We point out a problem in the analysis of the algorithm used by Gupta [5,6] and show that it does not give a constant factor approximation algorithm. However, we give another algorithm that does indeed obtain the result claimed by Gupta.

2 Feedback Vertex Set in Bipartite Tournaments

We are given a bipartite tournament $G = (V, A)$, and want to find a set of vertices $V' \subseteq V$ such that the subgraph of G induced by $V \setminus V'$ is acyclic, and $|V'|$ is minimal. We consider here a more general problem, in which for each $i \in V$, we are given a weight $w_i \geq 0$, and the goal is to find a feedback vertex set V' of minimum weight $\sum_{i \in V'} w_i$.

We use the following well known lemma [2,8].

Lemma 1. *A bipartite tournament is acyclic if and only if it contains no cycle of length 4.*

Given a bipartite tournament $G = (V, A)$, let \mathcal{C} be the set of cycles of length 4, i.e. $C \in \mathcal{C}$ is given by $\{i_1, (i_1, i_2), i_2, (i_2, i_3), i_3, (i_3, i_4), i_4, (i_4, i_1)\}$ with $i_1, \dots, i_4 \in V$ and $(i_1, i_2), (i_2, i_3), (i_3, i_4), (i_4, i_1) \in A$. By Lemma [11](#), we have the following integer program for the feedback vertex set problem in a bipartite tournament:

$$\begin{aligned}
 & \min \sum_{i \in V} w_i x_i \\
 \text{(FVS - BT)} \quad & \text{s.t.} \quad \sum_{i \in C \cap V} x_i \geq 1, \forall C \in \mathcal{C} \\
 & x_i \in \{0, 1\}, \forall i \in V.
 \end{aligned}$$

By solving the linear programming (LP) relaxation of this integer program, and rounding the values that are at least $\frac{1}{4}$, we can construct a solution with objective value of at most 4 times the optimal value. Prashant showed that in fact one can always find an optimal solution to the LP relaxation where some variable is at least $\frac{1}{3}$. Hence repeatedly rounding up these variables gives a 3-approximation algorithm.

We will begin by demonstrating another 3-approximation algorithm, where we bound the value of the solution against the dual of the LP relaxation, rather than the primal. Based on the ideas of this algorithm, we then show how to obtain an improved approximation algorithm.

The dual of the LP relaxation of (FVS-BT) is given by

$$\begin{aligned}
 & \max \sum_{C \in \mathcal{C}} y_C \\
 & \text{s.t.} \quad \sum_{C \in \mathcal{C}: i \in C} y_C \leq w_i, \forall i \in V \\
 & y_C \geq 0, \forall C \in \mathcal{C}.
 \end{aligned}$$

Let $\{x_i\}_{i \in V}$ be an optimal solution to the linear relaxation. Let L, R be the partition of the vertices, so that all arcs in the bipartite tournament have one endpoint in L and one endpoint in R .

Lemma 2. *There exists a 3-approximation algorithm for feedback vertex set in bipartite tournaments.*

Proof. We create an integer solution \hat{x}_i as follows: If $x_i \geq \frac{1}{2}$, or if $x_i > 0$ and $i \in L$ then $\hat{x}_i = 1$, else $\hat{x}_i = 0$. Note that $\{\hat{x}_i\}_{i \in V}$ is a feasible integer solution, since every cycle C has either some $i \in C$ such that $x_i \geq \frac{1}{2}$, or $|\{i \in C : x_i > 0\}| \geq 3$, in which case $\{i \in C : x_i > 0\} \cap L \neq \emptyset$.

Let $\{y_C\}_{C \in \mathcal{C}}$ be an optimal solution to the dual. We will need the following claim in our analysis:

Claim. Let $\{x_i\}_{i \in V}, \{y_C\}_{C \in \mathcal{C}}$ be optimal primal and dual solutions, and let \hat{x}_i be given as above. Then for every $C \in \mathcal{C}$ either $|\{i \in C : \hat{x}_i = 1\}| \leq 3$ or $y_C = 0$.

Consider any $C \in \mathcal{C}$. If $|\{i \in C : \hat{x}_i = 1\}| > 3$, then every vertex in C has $\hat{x}_i = 1$. This means that $x_i > 0$ for $i \in C \cap L$, and $x_i \geq \frac{1}{2}$ for $i \in C \cap R$. But then $\sum_{i \in C} x_i > 1$ and by complementary slackness we know that $y_C = 0$.

Note that if $\hat{x}_i = 1$, then $x_i > 0$, and by complementary slackness, we know that $\sum_{C \in \mathcal{C}: i \in C} y_C = w_i$. Therefore we get that

$$\begin{aligned} \sum_{i \in V: \hat{x}_i = 1} w_i &= \sum_{i \in V: \hat{x}_i = 1} \sum_{C: i \in C} y_C \\ &= \sum_{C \in \mathcal{C}} y_C |\{i \in C : \hat{x}_i = 1\}| \\ &\leq 3 \sum_{C \in \mathcal{C}} y_C \\ &= 3 \sum_{i \in V} w_i x_i. \end{aligned}$$

where the inequality follows from the claim. □

The algorithm and analysis in the proof of Lemma 2 suggest two ways of getting improved approximation guarantees. First of all, note that for i such that $0 < x_i < \frac{1}{2}$, the integer solution we created arbitrarily chose to set $\hat{x}_i = 1$, if $i \in L$; we could also have chosen to set $\hat{x}_i = 1$, if $i \in R$. Indeed, taking the better of these two solutions gives an improved approximation factor of 2.5, as we prove in Lemma 3. Secondly, instead of rounding up all variables on one side of the partition, we could only round up the variables that are at least $\frac{1}{2}$, and then resolve the linear program. In Lemma 4 we show that this gives a 2-approximation algorithm. Although we thus immediately improve the result from Lemma 3, we include Lemma 3 because it does not require us to solve linear programs repeatedly.

Lemma 3. *There exists a 2.5-approximation algorithm for feedback vertex set in bipartite tournaments.*

Proof. We define two solutions $\hat{x}_i^{(L)}$ and $\hat{x}_i^{(R)}$, where for $Z \in \{L, R\}$, we define \hat{x}_i^Z to be 1 if $x_i \geq \frac{1}{2}$, or if $x_i > 0$ and $i \in Z$. By the arguments in the proof of Lemma 2, both $\{\hat{x}_i^{(L)}\}_{i \in V}$ and $\{\hat{x}_i^{(R)}\}_{i \in V}$ are feasible integer solutions.

Claim. Let $\{x_i\}_{i \in V}, \{y_C\}_{C \in \mathcal{C}}$ be optimal primal and dual solutions, and let $\hat{x}_i^{(Z)}$ for $Z = L, R$ be defined as above. Then for every $C \in \mathcal{C}$

$$|\{i \in C : \hat{x}_i^{(L)} = 1\}| + |\{i \in C : \hat{x}_i^{(R)} = 1\}| \leq 5 \text{ or } y_C = 0.$$

Consider any $C \in \mathcal{C}$. If $y_C > 0$, then $|\{i \in C : x_i \geq \frac{1}{2}\}| \leq 2$. We consider three cases:

- (i) If $|\{i \in C : x_i \geq \frac{1}{2}\}| = 0$, then $|\{i \in C : \hat{x}_i^{(L)} = 1\}| \leq 2$ and $|\{i \in C : \hat{x}_i^{(R)} = 1\}| \leq 2$.

- (ii) If $|\{i \in C : x_i \geq \frac{1}{2}\}| = 1$, suppose without loss of generality that there exists $i \in C \cap L$ such that $x_i \geq \frac{1}{2}$. Then $|\{i \in C : \hat{x}_i^{(L)} = 1\}| \leq 2$ and $|\{i \in C : \hat{x}_i^{(R)} = 1\}| \leq 3$.
- (iii) If $|\{i \in C : x_i \geq \frac{1}{2}\}| = 2$, then by the fact that $y_C > 0$ and complementary slackness, we know that $\sum_{i \in C} x_i = 1$ and hence $|\{i \in C : x_i > 0\}| = 2$, so $|\{i \in C : \hat{x}_i^{(L)} = 1\}| = 2$ and $|\{i \in C : \hat{x}_i^{(R)} = 1\}| = 2$.

◇

As before, if $\hat{x}_i^{(Z)} = 1$, then $x_i > 0$, and by complementary slackness, we know that $\sum_{C \in \mathcal{C}: i \in C} y_C = w_i$. So now we get that

$$\begin{aligned} \sum_{i \in V: \hat{x}_i^{(L)} = 1} w_i + \sum_{i \in V: \hat{x}_i^{(R)} = 1} w_i &= \sum_{i \in V: \hat{x}_i^{(L)} = 1} \sum_{C: i \in C} y_C + \sum_{i \in V: \hat{x}_i^{(R)} = 1} \sum_{C: i \in C} y_C \\ &= \sum_{C \in \mathcal{C}} y_C (|\{i \in C : \hat{x}_i^{(L)} = 1\}| + |\{i \in C : \hat{x}_i^{(R)} = 1\}|) \\ &\leq 5 \sum_{C \in \mathcal{C}} y_C \\ &= 5 \sum_{i \in V} w_i x_i, \end{aligned}$$

where the inequality follows from the claim. □

Lemma 4. *There exists a 2-approximation algorithm for feedback vertex set in bipartite tournaments.*

Proof. Our algorithm solves the linear program (FVS – BT), rounds up the variables that are $\geq \frac{1}{2}$, remove the corresponding vertices from the graph and then resolves the linear program. If no variables $\geq \frac{1}{2}$ exist, we use the algorithm in the proof of Lemma 2 to complete the solution to a feasible integer solution.

Let V_k be the vertex set at the beginning of the k -th iteration of the algorithm, i.e. $V_1 = V$, and $V_k \subset V_{k-1}$ for $k \geq 2$. Let $G(V_k)$ be the induced bipartite tournament on V_k , let $OPT(V_k)$ be the optimal value of the LP on $G(V_k)$, and let $ALG(V_k)$ be the weight of the algorithm’s solution restricted to V_k . Let ℓ be the total number of iterations of the algorithm.

We prove by backward induction on the algorithm that the variables rounded to 1 in iterations k, \dots, ℓ give a feasible feedback vertex set on $G(V_k)$ of weight at most $2OPT(V_k)$. Since $OPT(V_1)$ is a lower bound on the value of the optimal feedback vertex set, this implies the lemma.

At the start of the last iteration, let $\mathcal{C}(V_\ell)$ be the 4-cycles in $G(V_\ell)$. Let $\{x_i^{(\ell)}\}_{i \in V_\ell}$ be an optimal primal, and $\{y_C^{(\ell)}\}_{C \in \mathcal{C}(V_\ell)}$ be an optimal dual for the LP on this instance. Since $x_i^{(\ell)} < \frac{1}{2}$ for each $i \in V_\ell$, every $C \in \mathcal{C}(V_\ell)$ contains at least 3 vertices with strictly positive value $x_i^{(\ell)}$. Hence if we round up the variables for $i \in L$ with $x_i^{(\ell)} > 0$, we hit every cycle in $\mathcal{C}(V_\ell)$ at least once, and at most twice.

It follows that the solution we create is feasible on $G(V_\ell)$, and following the proof of Lemma 2, its weight is at most $2 \sum_{C \in \mathcal{C}(V_\ell)} y_C = 2 \sum_{i \in V_\ell} w_i x_i^{(\ell)} = 2OPT(V_\ell)$.

Now consider the beginning of iteration $k < \ell$. We solve the LP on $G(V_k)$, and let $\{x_i^{(k)}\}_{i \in V_k}$ be the optimal primal solution. The algorithm returns a feasible solution on $G(V_k)$: every 4-cycle either has a vertex i such that $x_i^{(k)} \geq \frac{1}{2}$, or it is a 4-cycle also in $G(V_{k+1})$, and by induction we know that our solution hits every 4-cycle in $G(V_{k+1})$.

By induction, $ALG(V_{k+1}) \leq 2OPT(V_{k+1})$. Note that

$$\begin{aligned} ALG(V_k) &= \sum_{i \in V_k : x_i^{(k)} \geq \frac{1}{2}} w_i + ALG(V_{k+1}) \\ &\leq 2 \sum_{i \in V_k : x_i^{(k)} \geq \frac{1}{2}} w_i x_i^{(k)} + 2OPT(V_{k+1}) \\ &= 2 \sum_{i \in V_k : x_i^{(k)} \geq \frac{1}{2}} w_i x_i^{(k)} + 2 \sum_{i \in V_{k+1}} w_i x_i^{(k+1)}. \end{aligned}$$

We note that $\{x_i^{(k)}\}_{i \in V_{k+1}}$ (the optimal LP solution on $G(V_k)$ restricted to V_{k+1}) is a feasible solution to the LP on $G(V_{k+1})$. Therefore $2 \sum_{i \in V_{k+1}} w_i x_i^{(k+1)} \leq 2 \sum_{i \in V_{k+1}} w_i x_i^{(k)}$, and since every vertex is either in V_{k+1} , or has $x_i^{(k)} \geq \frac{1}{2}$, we get that $ALG(V_k) \leq 2 \sum_{i \in V_k} w_i x_i^{(k)} = 2OPT(V_k)$. \square

We conclude this section by showing that the result in Lemma 4 is the best one can hope for if using the optimal value of the LP relaxation of (FVS – BT) as a lower bound. The *integrality gap* of an integer linear program is the worst case ratio between the optimal value of the integer program and the optimal value of its LP relaxation, and hence a lower bound on the integrality gap implies a lower bound on the approximation ratio of an algorithm that bounds the cost of the algorithm’s solution against the optimal value of the LP relaxation.

Lemma 5. *The integrality gap of (FVS – BT) is 2.*

Proof. By Lemma 4 the integrality gap is at most 2. We construct an example in which the integrality gap approaches 2. In particular, we will show that there exists an instance on $2n$ vertices for which the minimum feedback vertex set has size at least $n - 1$. It remains to note that setting $x_i = \frac{1}{4}$ for all $i \in V$ always gives a feasible solution to (FVS – BT).

Let G_{2n} be a bipartite tournament with the following properties. We have vertices $L = \{\ell_1, \dots, \ell_n\}$ and $R = \{r_1, \dots, r_n\}$ and all arcs have one endpoint in L and one endpoint in R . In addition, we require that the arc between ℓ_i and r_i is directed from ℓ_i to r_i . We show by induction that there exists such a graph G_{2n} with minimum feedback vertex set of size at least $n - 1$ for all $n \geq 2$.

For $n = 2$, G_4 is just a cycle of length 4. Given a graph G_{2n} , we construct $G_{2(n+1)}$ by adding a vertex ℓ_{n+1} to L and a vertex r_{n+1} to R , and we add the arc (ℓ_{n+1}, r_{n+1}) , plus arcs (r_i, ℓ_{n+1}) and (r_{n+1}, ℓ_i) for every $i \leq n$. Note that

the new arc (ℓ_{n+1}, r_{n+1}) is in a directed 4-cycle with every pair (ℓ_i, r_i) . Hence a feedback vertex set in $G_{2(n+1)}$ either removes one of the new vertices ℓ_{n+1}, r_{n+1} , plus a minimum feedback vertex set in G_{2n} (thus removing at least n vertices) or it must remove one of ℓ_i, r_i for every $i \leq n$. Hence the size of the feedback vertex set on $G_{2(n+1)}$ is at least n . \square

3 Feedback Arc Set in Bipartite Tournaments

We now consider the feedback arc set problem in bipartite tournaments. Gupta [5,6] recently gave an algorithm for this problem, and claimed it was a 4-approximation algorithm. We believe however that there is an error in the analysis. The algorithm is similar to that proposed by Ailon et al. [1] for the feedback arc set problem in tournaments and it recursively constructs an ordering of the vertices. The feedback arc set then consists of the *backward arcs*, i.e. the arcs going from right to left in the final ordering. To order the vertices, the proposed algorithm chooses an arc (i, j) as “pivot”, and orders a vertex u to the left of the arc if either $(u, i) \in A$ or $(u, j) \in A$, and to the right of (i, j) otherwise. It then recurses on the two instances induced by the vertices on the left and on the right respectively. The key to their analysis, which we believe to be incorrect, is the claim that “an arc $(u, v) \in A$ becomes a backward arc if and only if $\exists(i, j) \in A$ such that (i, j, u, v) forms a directed 4-cycle in G and (i, j) was chosen as the pivot when all 4 were part of the same recursive call”. Note, however, that an arc (u, v) may also become backward if $(i, u) \in A$ and $(v, j) \in A$, and (i, j) is chosen as the pivot when i, j, u, v were in the same recursive call. In that case, i, j, u, v are not in a directed 4-cycle, since we have $(i, u), (u, v), (v, j), (i, j) \in A$.

As an example, in the following instance, the optimal feedback arc set has size 1, and the expected number of backward arcs created by Gupta’s algorithm is $O(n^2)$. We have vertices $L = \{\ell_1, \dots, \ell_n\}$ and $R = \{r_1, \dots, r_n\}$. We think of the vertices as being ordered as $\ell_1, r_1, \ell_2, r_2, \dots, \ell_n, r_n$. All arcs have one endpoint in R and one endpoint in L and go from left to right except for the arc (r_n, ℓ_1) . The number of backward arcs created in the first iteration of Gupta’s algorithm is $O(n^2)$ with constant probability: choosing an arc at random is the same as choosing a left vertex and a right vertex independently at random. Hence the probability that we choose (ℓ_i, r_j) with $1 < i \leq \frac{1}{4}n$ and $\frac{3}{4}n \leq j < n$ is $\frac{n/4-1}{n} \frac{n/4-1}{n} \approx \frac{1}{16}$. By pivoting on this arc, the arcs $(r_k, \ell_{k'})$ for $i \leq k \leq k' \leq j$ become backward, and since $i \leq \frac{1}{4}n, j \geq \frac{3}{4}n$, there are $O(n^2)$ such arcs.

We propose a more direct extension of the algorithm of Ailon et al. [1], or more precisely, we directly apply its derandomization by Van Zuylen et al. [11] (see also [10]) to the case of the feedback arc set problem in bipartite tournament. This allows us to obtain a 4-approximation algorithm.

We will use the following linear program. Let $x_{(i,j)} = 1$ denote that i is ordered before j . For an arc $(i, j) \in A$, let $w_{(i,j)} = 1$. If $(i, j) \notin A$, we let $w_{(i,j)} = 0$. Note that if i and j are not both in L or R , then $w_{(i,j)} + w_{(j,i)} = 1$, otherwise $w_{(i,j)} = w_{(j,i)} = 0$.

$$\begin{aligned}
 & \min \sum_{i < j} (w_{(j,i)}x_{(i,j)} + w_{(i,j)}x_{(j,i)}) \\
 \text{(FAS) s.t. } & x_{(i,j)} + x_{(j,k)} + x_{(k,i)} \geq 1, \forall \text{ distinct } i, j, k \\
 & x_{(i,j)} + x_{(j,i)} = 1, \forall \text{ distinct } i, j \\
 & x_{(i,j)} \geq 0, \forall \text{ distinct } i, j.
 \end{aligned}$$

The algorithm proposed in [1110] for the feedback arc set problem in *tournaments* (rather than bipartite tournaments) starts by solving the linear program (FAS). Based on the optimal solution, they form a tournament $T = (V, A_T)$, which has $(i, j) \in A_T$ if $x_{(i,j)} \geq \frac{1}{2}$ (where ties are broken arbitrarily if $x_{(i,j)} = x_{(j,i)} = \frac{1}{2}$).

The algorithm recursively constructs an ordering of the vertices, by choosing a pivot *vertex* k , ordering vertex i to the left of k if $(i, k) \in A_T$, and to the right of k if $(k, i) \in A_T$. It then recurses on the instances induced by the vertices on the left and right.

We can directly apply this algorithm to the feedback arc set problem in bipartite tournaments. Note that there are two types of backward arcs (from the original bipartite tournament) in the ordering constructed: arcs $(i, j) \in A$ for which $(j, i) \in A_T$ (i.e. $x_{(j,i)} \geq \frac{1}{2}$) and one of i, j is chosen as pivot when i, j are in the same recursive call, and arcs $(i, j) \in A$ for which there exists k such that $(j, k) \in A_T, (k, i) \in A_T$ and k is chosen as pivot when i, j, k are in the same recursive call.

Clearly, we can bound the cost of the first type of backward arcs against twice the contribution of (i, j) to the linear program’s objective value. In order to bound the cost of the second type of backward arcs, [1110] chooses a pivot carefully. Let $T_k(V)$ denote the pairs (i, j) such that $(j, k) \in A_T$ and $(k, i) \in A_T$. In a recursive call with vertex set V , the pivot k is the vertex that minimizes

$$\frac{\sum_{(i,j) \in T_k(V)} w_{(i,j)}}{\sum_{(i,j) \in T_k(V)} (w_{(j,i)}x_{(i,j)} + w_{(i,j)}x_{(j,i)})}.$$

It follows from the Theorem 2.1 in Van Zuylen and Williamson [11] that if the following condition holds for every $(i, j), (j, k), (k, i) \in A_T$, then it is always possible to choose a pivot k such that the above ratio is at most 4.

$$\begin{aligned}
 w_{(i,j)} + w_{(j,k)} + w_{(k,i)} \leq 4 & \left(w_{(j,i)}x_{(i,j)} + w_{(i,j)}x_{(j,i)} + \right. \\
 & \left. + w_{(k,j)}x_{(j,k)} + w_{(j,k)}x_{(k,j)} + w_{(i,k)}x_{(k,i)} + w_{(k,i)}x_{(i,k)} \right). \tag{1}
 \end{aligned}$$

Hence we can bound the cost of the second type of backward arcs against 4 times their contribution to the linear program’s objective value. It thus follows the algorithm is a 4-approximation algorithm.

Lemma 6. *There exists a 4-approximation algorithm for feedback arc set in bipartite tournaments.*

Proof. We need to show that (II) holds. Note that for any triple such that $(i, j), (j, k), (k, i) \in A_T$, it must either be the case that all three vertices were on the same side of the bipartite tournament G , or exactly two were on one side, and the other vertex was on the other side. In the first case, the left hand side of (II) is 0 and there is nothing to prove. In the second case, suppose without loss of generality that $w_{(i,j)} = w_{(j,i)} = 0$.

We need to show that

$$w_{(j,k)} + w_{(k,i)} \leq 4 \left(w_{(k,j)}x_{(j,k)} + w_{(j,k)}x_{(k,j)} + w_{(i,k)}x_{(k,i)} + w_{(k,i)}x_{(i,k)} \right).$$

We rewrite the right hand side as $4 \left((1 - w_{(j,k)})x_{(j,k)} + w_{(j,k)}(1 - x_{(j,k)}) + (1 - w_{(k,i)})x_{(k,i)} + w_{(k,i)}(1 - x_{(k,i)}) \right) = 4 \left(w_{(j,k)} + w_{(k,i)} - x_{(j,k)}(2w_{(j,k)} - 1) - x_{(k,i)}(2w_{(k,i)} - 1) \right)$.

Note that $x_{(j,k)} \geq \frac{1}{2}, x_{(k,i)} \geq \frac{1}{2}$ and $x_{(i,j)} \geq \frac{1}{2}$ by the fact that $(j, k), (k, i), (i, j) \in A_T$. Hence the right hand side is non-increasing in $w_{(k,i)}$ and $w_{(j,k)}$, and since the left hand side is increasing in $w_{(k,i)}$ and $w_{(j,k)}$, it is enough to consider the case when $w_{(k,i)} = w_{(j,k)} = 1$. It thus remains to show that $4(2 - x_{(j,k)} - x_{(k,i)}) \geq 2$.

By the second set of constraints of (FAS), $4(2 - x_{(j,k)} - x_{(k,i)}) = 4(x_{(k,j)} + x_{(i,k)})$, and by the first set of constraints, $x_{(i,k)} + x_{(k,j)} \geq 1 - x_{(j,i)} = x_{(i,j)} \geq \frac{1}{2}$, which directly gives the desired inequality. \square

We leave open the question of whether there exists a combinatorial algorithm that achieves the same guarantee. The idea of Gupta’s algorithm to pivot on an *arc* of the graph, rather than a vertex as in Ailon et al. [1] is interesting, and it may be possible to modify the algorithm so that it does achieve a constant approximation guarantee.

References

1. Ailon, N., Charikar, M., Newman, A.: Aggregating inconsistent information: ranking and clustering. In: STOC 2005: Proceedings of the 37th Annual ACM Symposium on Theory of Computing, pp. 684–693 (2005)
2. Cai, M.-C., Deng, X., Zang, W.: A min-max theorem on feedback vertex sets. *Math. Oper. Res.* 27, 361–371 (2002)
3. Even, G., Naor, J., Schieber, B., Sudan, M.: Approximating minimum feedback sets and multicuts in directed graphs. *Algorithmica* 20(2), 151–174 (1998)
4. Guo, J., Hüffner, F., Moser, H.: Feedback arc set in bipartite tournaments is NP-complete. *Inf. Process. Lett.* 102(2-3), 62–65 (2007)
5. Gupta, S.: Feedback arc set problem in bipartite tournaments. In: Cai, J.-Y., Cooper, S.B., Zhu, H. (eds.) TAMC 2007. LNCS, vol. 4484, pp. 354–361. Springer, Heidelberg (2007)
6. Gupta, S.: Feedback arc set problem in bipartite tournaments. *Inf. Process. Lett.* 105(4), 150–154 (2008)
7. Kann, V.: On the approximability of NP-complete optimization problems, Ph.D. thesis, Department of Numerical Analysis and Computing Science, Royal Institute of Technology, Stockholm (1992)

8. Sasatte, P.: Improved approximation algorithm for the feedback set problem in a bipartite tournament. *Operations Research Letters* 36(5), 602–604 (2008)
9. Seymour, P.D.: Packing directed circuits fractionally. *Combinatorica* 15(2), 281–288 (1995)
10. van Zuylen, A., Hegde, R., Jain, K., Williamson, D.P.: Deterministic pivoting algorithms for constrained ranking and clustering problems. In: *SODA 2007: Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 405–414 (2007)
11. van Zuylen, A., Williamson, D.P.: Deterministic pivoting algorithms for constrained ranking and clustering problems. *Math. Oper. Res.* (to appear), <http://www.itcs.tsinghua.edu.cn/~anke/MOR.pdf>

An Online Algorithm for Applying Reinforcement Learning to Handle Ambiguity in Spoken Dialogues

Fangju Wang and Kyle Swegles

University of Guelph, Guelph, Ontario, Canada N1G 2W1

Abstract. Spoken dialogue systems (SDSs) have been widely used in human-computer communications, including database querying, online trouble shooting advising, etc. A major challenge in building an SDS is to handle ambiguity in natural languages. User queries, questions, descriptions in a natural language may be ambiguous. To be effective in practical applications, an SDS must be able to disambiguate input from its user(s). In our research, we develop an online algorithm for applying reinforcement learning to handle ambiguity in SDSs. We introduce a new user dialogue policy into the framework of reinforcement learning to model user dialogue behavior. Also, differing from the current reinforcement learning algorithms in speech and language processing that are characterized by offline training, our algorithm conducts both offline and online detection of user dialogue behavior. In this paper, we present the online algorithm for reinforcement learning, emphasizing the detection of user dialogue behavior. We also describe the initial implementation and experiments.

1 Introduction

A spoken dialogue system (SDS) converses with the user for answering questions, providing advice or help in a given domain, or for other purposes. Application examples include database querying, online trouble-shooting advising, educational tutoring, and so on. In a dialogue, an SDS speaks to the user, and listens to and understands the user's speech. A major component in an SDS is a dialogue manager. It applies a dialogue strategy to control dialogue flows.

The dialogue strategy is used to make human-computer conversations correct and meaningful in serving the purpose of the SDS. When an SDS needs to speak to the user, its dialogue strategy guides the selection of words. To make right selections, a dialogue strategy must first interpret user's voice input correctly.

User voice input in a natural language may be ambiguous. Ambiguity may be caused by noises, homonyms, different word tags, multiple parsing options, and so on. Because of ambiguity, a user question or description may be interpreted by an SDS in different ways. If the dialogue strategy takes a wrong interpretation, it may select the wrong words to say, which would make the dialogue meaningless. Therefore, an SDS must be able to disambiguate user input. When ambiguity

occurs in a dialogue, the SDS can generate the most plausible interpretation and use it as the basis to continue the dialogue.

Machine learning algorithms are effective tools for disambiguation. Of the algorithms, *reinforcement learning (RL)* has been chosen by more and more researchers and practitioners for developing disambiguation techniques in recent years. A reinforcement learning algorithm works in an interactive way. A learning agent may obtain the required knowledge or information through its positive and negative experience when interacting with the user. When used for disambiguation, the agent learns how to determine the meaning of ambiguous user speech after it repeatedly converses with the user.

Currently, in the fields of speech and language processing, including building SDSs, reinforcement learning is mainly used in offline manners. In some of the applications of reinforcement learning, training data in human dialogue corpora are used to teach learning agents the knowledge for voice and language processing. In other applications, learning agents interact with simulated users. Few systems employ online algorithms that allow agents to interact with real users.

Offline applications of reinforcement learning are effective in obtaining voice and language processing knowledge, including disambiguation knowledge, about *general users*. However, a system using an offline learning algorithms alone has difficulties to obtain the knowledge about the user or users that the system actually converses with. Different people or different groups of people may have something special in speaking and using a language, for example, spacial accents, dialects, grammar usage, and so on. To individual SDSs, the knowledge about their specific users is as important as the knowledge about general users. Lack of the knowledge about specific users may significantly weaken SDSs' abilities to handle ambiguity, and thus, reduce their dialogue abilities. It is very difficult, if not impossible, for SDS developers to follow every product to acquire knowledge about dialogue behavior of its specific user(s). An automated online learning technique is needed.

In this paper, we present a novel online algorithm for applying reinforcement learning for disambiguation. In the algorithm, the learning agent interacts with the real users. It detects users' dialogue behavior of speaking and using a language, while optimizing the SDS's dialogue policy. A major improvement to the reinforcement learning algorithm is the introduction of a *user dialogue policy* that can be used to model the user's dialogue behavior. When ambiguity occurs in a user's voice input, knowledge about this user's dialogue behavior may be very helpful to determine the meaning that the user really wants to express.

2 Related Work

In this section, we review the major research activities for applying reinforcement learning algorithms in building spoken dialogue systems. The algorithms can be categorized into learning mainly with simulated users, and learning mainly with a human dialogue corpus. Since most of the work is in the first category, we review algorithms for learning with simulated users only. A representative algorithm for learning with a human dialogue corpus can be found in [5].

In the work by Scheffler and Young [8], reinforcement learning was used to learn optimal dialogue strategies for a frame-based cinema telephone system. A detailed probabilistic user simulator that modeled behavior and ASR errors was trained on a human dialogue corpus. An MDP model was constructed and the Watkins $Q(\lambda)$ algorithm with eligibility traces, which incorporated Monte Carlo learning, was used to develop optimal policies. The learned dialogue strategy was shown to be significantly more successful than the baseline.

Levin and co-workers developed a stochastic approach to user simulation in building an airline database query system (ATIS) [4]. Greeting probabilities were learned from a human dialogue corpus by estimating the probability of a user response given system actions. A Monte Carlo Q-Learning algorithm with exploration starts was applied with a simple task completion reward function. The stochastic simulated user was used to produce up to 2500 dialogue sessions to allow RL to find an optimal policy.

Melichell and Cenek developed an n-gram technique to model users thereby extending the human dialogue corpus [1]. The model was trained using existing annotated COMMUNICATOR data describing dialogues for flight booking. The problem was framed as an MDP and up to 50,000 dialogues were used by the RL algorithm. A reward function helped maximize the chances of choosing the subject that matches the users intentions. The RL algorithm was unique in that it not only considered state information, but also the user and systems last actions. Two Dialogue Strategies were simulated - a UDM strategy that considered the last user move and a USDM strategy that considered both the users and systems last moves. The UDM and USDM strategies were able to cope with invalid user input as they took into account the users last dialogue action.

In contrast to the small sized dialogue corpus used in [5], the work reviewed above [8][4][1] all used user simulation to extend the dialogue corpus by a substantial amount. As a result, the RL algorithms used were able to find optimal policies that showed strong improvement. However, the accuracy of the simulated user techniques was not demonstrated. Extending a dialogue corpus by 2500 dialogues sessions as in [4] could modify the overall behavior of the dialogue corpus. This modified behavior could lead to a policy that is not suited for the behavior of the users that the dialogue corpus was created from.

3 The Online Algorithm

In this section, we present the online reinforcement learning algorithm for detecting user dialogue behavior. We introduce a new policy, *the user dialogue policy*, into reinforcement learning to model user dialogue behavior. A major difference from the existing algorithms is that our algorithm allows an SDS to learn through interactions with real users and to use the knowledge for disambiguation and for dialogue flow control. In the existing algorithms, user simulation is used to extend dialogue corpora. In our algorithm, interactions with real users are used to further extend user simulation, to achieve higher accuracy.

3.1 A Reinforcement Learning Algorithm

Before presenting the extended online algorithm, we briefly introduce the general reinforcement learning (RL) technique. Details about reinforcement learning can be found in many excellent books on this subject, for example [9].

Assume an agent is used in a problem-solving task, and a reinforcement learning algorithm helps the agent learn knowledge for solving the problems.

The major components in a reinforcement algorithm include a set of states that the agent may perceive (S), a set of possible actions that the agent may take (A), a set of rewards that the agent may receive (\mathcal{R}) after it takes actions, and a policy π . The policy is used to guide the agent to select the actions to take based on the agent's states. The objective of RL is to develop (and modify) the optimal policy through the interactions between the agent and its environment.

In state s , when a sequence of actions have been taken, there is a *return* R defined as

$$R = \sum_{i=0}^n \gamma^i r_i \quad (1)$$

where $r_i \in \mathcal{R}$ is the reward when the i th action has been performed, and γ is some future reward discounting factor ($0 \leq \gamma \leq 1$). The discounting factor is used to put heavier weights on the rewards in the near future.

When a policy has been developed, each state $s \in S$ is associated with value functions $V(s)$ and $Q(s, a)$. $V(s)$ is an expected return:

$$V(s) = E[R|s, \pi] \quad (2)$$

and $Q(s, a)$ can be defined in terms of V :

$$Q(s, a) = E[R|s, \pi, a] = \sum_{s'} V(s')P(s'|s, a) \quad (3)$$

where s' is the state the agent perceives after it takes action a in s , and $P(s'|s, a)$ is the probability that the agent perceives s' after taking action a in s .

Reinforcement learning is conducted when the agent repeatedly interacts with the environment, that is, taking actions, perceiving new states, and receives rewards. The learning process optimizes π by optimizing the $V(s)$ or $Q(s, a)$ that are related with π . Once the $V(s)$ and $Q(s, a)$ have been developed, they can be used to guide the agent to select actions to take. In a state, the agent can select the action that maximizes V or Q .

In our research, we added another policy π_h into reinforcement learning. π_h is the human user's dialogue "policy" that the learning agent detects through its interactions with the user. It can be used to model the user's dialogue behavior, that is, what information the user may most likely express in different situations. The information may be the meaning of user questions, statements, descriptions, and so on. The user dialogue policy is related with two value functions: $V^{\pi_h}(s)$ and $Q^{\pi_h}(s, a)$.

3.2 Reinforcement Learning in Spoken Dialogues

We apply reinforcement learning in spoken dialogues. The development of an SDS dialogue strategy and detection of user dialogue behavior can be formulated as a reinforcement learning problem, in which the learning agent repeatedly interacts with the environment (including the user and the related aspects of the world) to achieve the goals. The agent learns from its experience in interacting with the user. In this paper, we discuss the detection of user dialogue behavior only. The development of an optimal SDS dialogue strategy is described in [10].

We model an SDS as a tuple

$$(S, U, D, H, \pi_m, \pi_h, \mathcal{R}) \quad (4)$$

where S is a directed graph of the states that the learning agent may perceive and transitions from states to states, U is a database of both system and user utterances, D is a database that contains dialogue history, H is a database recording other historical information that may be required for the dialogue manager to control dialogue flows, π_m is the SDS (machine) dialogue policy, π_h is the human user dialogue policy, and \mathcal{R} is a set of rewards.

State graph S is defined as

$$S = (N, E) \quad (5)$$

where N is a set of nodes and E is a set of edges. Each node represents a state and each edge a transition from a state to another. An $e \in E$ from s to s' is related with a $u \in U$ and an $r \in \mathcal{R}$,

$$s \xrightarrow{u, r} s' \quad (6)$$

denoting utterance u causes a state transition from s to s' and the reward is r .

Each $u \in U$ is an utterance represented as a semantic graph. When the SDS receives a user utterance, it conducts voice recognition (VR), syntactic analysis, and semantic analysis. The output of the semantic analysis is a semantic graph, which is the internal representation of the utterance. The system utterances that the SDS may say are also stored in U as semantic graphs. When the SDS has decided to say an utterance, it converts the corresponding semantic graph into a sentence and then invoke a text-to-speech (TTS) system to say it to the user. (In this paper, we do not address the issues of VR, syntactic and semantic analysis, and TTS. We focus on the learning aspects.)

The D database contains the dialogue sessions that have occurred

$$D = \{d_1, d_2, d_3, \dots\} \quad (7)$$

where d_i is a dialogue session that is a sequence of user and system utterances:

$$d_i = (u_{i,1}^h, u_{i,1}^m, u_{i,2}^h, u_{i,2}^m, \dots) \quad (8)$$

where $u_{i,j}^h$ is the j th (human) user utterance in the i th dialogue session, and $u_{i,j}^m$ is the j th system (machine) utterance in the i th dialogue session. A u in a

dialogue session references a $u \in U$. For not losing generality, we assume that the system and user speak in turn.

Reinforcement learning is based on the assumption of Markov Decision Process (MDP). This implies that all the information required for decision making must be available in the current state, i.e.

$$P(s_{t+1}|s_t, s_{t-1}, \dots, s_0, u_t, u_{t-1}, \dots, u_0) = P(s_{t+1}|s_t, u_t) \tag{9}$$

To make (9) true, we create a mapping

$$\{U, D, H\} \mapsto S. \tag{10}$$

In a later subsection, we describe how a state is associated with the information stored in U, D , and H .

The detection of user dialogue behavior by reinforcement learning consists of two steps: learning in training and learning in interaction. In the first step, the agent learns from training data to create an initial user dialogue policy, which models the dialogue behavior of general users. In the second step, the agent learns from the user(s) it interacts with to refine the initial policy. The refined policy models the dialogue behavior of the user or users who use the SDS.

3.3 Detecting User Dialogue Policy in Training

Training is conducted by applying reinforcement learning to a simulated user, with S, U, D, H being empty.

The simulated user is represented as a set of dialogue sessions. A training dialogue session is a sequence of system and user utterances, with each utterance assigned a reward $r \in \mathcal{R}$. At the end of a training session, there is a value v^h measuring the performance of the user, and a value v^m measuring the performance of the SDS. Formally, a training dialogue session is represented as

$$t_i = (u_{i,1}^h, r_1^h, u_{i,1}^m, r_1^m, u_{i,2}^h, r_2^h, u_{i,2}^m, r_2^m, \dots, v_i^h, v_i^m) \tag{11}$$

where r_j^h is the reward given to the j th user utterance, and r_j^m is the reward given to the j th system utterance.

In the training process, reinforcement learning is used to create the user dialog policy π_h that is optimal with respect to the simulated user (training data). As mentioned before, a user dialogue policy is characterized by functions V^{π_h} and Q^{π_h} . Thus a major task in training is to create V^{π_h} and Q^{π_h} from training data. Here we discuss the creation of V^{π_h} only. Q^{π_h} can be calculated from V^{π_h} .

The value function V^{π_h} of strategy π_h in state s can be expressed as

$$\begin{aligned} &V^{\pi_h}(s) \tag{12} \\ &= E_{\pi_h} \{r_{\tau+1} + \gamma V^{\pi_h}(s_{\tau+1}) | s_{\tau} = s\} \\ &= \sum_u \pi_h(s, u) \sum_{s'} \mathcal{P}_{ss'}^u [R_{ss'}^u + \gamma V^{\pi_h}(s')] \end{aligned}$$

where τ indicates the current point of time, $\pi_h(s, u)$ is the probability for the user to take utterance u in state s when π_h is applied, E_{π_h} is expectation conditional on policy π_h , and

$$\mathcal{P}_{ss'}^u = P\{s_{\tau+1} = s' | s_\tau = s, u_\tau = u\} \tag{13}$$

is the probability that taking utterance u in state s leads to state s' ,

$$\mathcal{R}_{ss'}^u = E_{\pi_h} \{r_{\tau+1} | s_\tau = s, u_\tau = u, s_{\tau+1} = s'\} \tag{14}$$

is the expected immediate rewards when π_h is used to taking u in s to go to s' .

The training algorithm is informally described as follows. When training dialogue session t_i is presented, graph \mathcal{G}_i

$$s_{i0} \xrightarrow{u_{i,1}^h, r_1^h} s_{i1}^m \xrightarrow{u_{i,1}^m, r_1^m} s_{i1}^h \xrightarrow{u_{i,2}^h, r_2^h} s_{i2}^m \xrightarrow{u_{i,2}^m, r_2^m} s_{i2}^h \xrightarrow{u_{i,3}^h, r_3^h} \dots \tag{15}$$

is created, where s_{i0} is the starting state for the i th training session, s_{ij}^m is the j th state in which the SDS says an utterance, and s_{ij}^h is the j th state in which the user says an utterance.

State s is associated with $g(s)$ that is the *state semantic graph* of s . A state semantic graph is derived from the *utterance semantic graphs* of the utterances on the path from s_0 to s . For example, in (15),

$$g(s_{i2}^m) \prec (g(u_{i,1}^h), g(u_{i,1}^m), g(u_{i,2}^h)) \tag{16}$$

where \prec denotes a derivation operation, and $g(u)$ is the utterance semantic graph of u . A semantic graph represents knowledge about a state in a dialogue, for example, what the SDS has told the user, and what the user is asking. Since the semantic graph of a state is derived from the semantic graphs of all the utterances that lead to the state, the state semantic graph contains the all information for decision making. In this way, the states can be used in MDP.

After the semantic graph of a training session is created, each $s \in \mathcal{G}$ is compared with the s 's in S . The comparison is performed by approximate matching of their semantic graphs. If an $s \in \mathcal{G}$ does not match any $s \in S$, it is added to S , and its in and out transitions are added to S too. If an $s \in \mathcal{G}$ matches an $s \in S$, its in and out transitions are added to the $s \in S$.

In S , each state has zero and more in-edges and zero or more out-edges. As mentioned before, each edge denotes a state transition and is associated with a u and an r . For each $s \in S$ we can calculate $\pi_h(s, u)$ using information about the out-edges and their u 's. Using equations (13) and (14) we can calculate $\mathcal{P}_{ss'}^u$ and $\mathcal{R}_{ss'}^u$. Then using (12) we can calculate V^{π_h} for all $s \in S$. The calculation of $\mathcal{R}_{ss'}^u$ and V^{π_h} follows the backup rule.

3.4 Refining User Dialogue Policy in Interaction

Learning in interactions is performed online, when the SDS has been actually used by its user(s). The goal of this step is to customize the user dialogue policy so that it fits best the particular user or users of the system.

Most human-computer dialogues are episodic. We can organize dialogues as sessions like

$$(u_{i,1}^h, u_{i,1}^m, u_{i,2}^h, u_{i,2}^m, \dots). \tag{17}$$

For each dialogue session, a heuristic method is used to evaluate user and SDS performance. The utterances and dialogues are stored in databases U and D , and are used to improve π_m and π_h , which are the dialogue policies of the SDS and user. (We discuss π_h only in this paper.)

Periodically, the actual dialogue sessions are used to improve the dialogue policy π_h . The policy developed for a general user may not fit best the particular user who uses the system. The special dialogue behaviors of the user may require that the policy should be modified (improved) for best fitting the user. The modification occurs when the utterance that maximizes V^{π_h} or Q^{π_h} differs from the actual utterance.

Let π_h be the current user dialogue policy, and $\pi_h(s)$ be the utterance that maximizes $V^{\pi_h}(s)$. If there is an actual user utterance u in state s ($u \neq \pi_h(s)$) that contributes better user performance than $\pi_h(s)$, we may expect that there exists policy π'_h such that $\pi'_h(s) = u$. If $\forall s \in S$,

$$Q^{\pi_h}(s, \pi'_h(s)) \geq V^{\pi_h}(s), \tag{18}$$

then based on the policy improvement theorem, π'_h must be as good as, or better than, π_h . That is, $\forall s \in S$

$$V^{\pi'_h}(s) \geq V^{\pi_h}(s). \tag{19}$$

Policy π'_h can be obtained by improving π_h through the policy iteration process, which consists of repeated policy evaluation and policy improvement until the new policy becomes stable. The following is the iteration process. All the values are calculated using the actual u when $u \neq \pi_h(s)$.

1. Initialization

Initialize V based on π_h

2. Policy Evaluation

Repeat

$\Delta \leftarrow 0$

For each $s \in S$

$v \leftarrow V(s)$

$V(s) \leftarrow \sum_{s'} \mathcal{P}_{ss'}^{\pi_h(s)} [\mathcal{R}_{ss'}^{\pi_h(s)} + \gamma V(s')]$

$\Delta \leftarrow \max(\Delta, |v - V(s)|)$

until $\Delta < \theta$

3. Policy Improvement

policy_stable \leftarrow true

For each $s \in S$

$b \leftarrow \pi_h(s)$

$\pi_h(s) \leftarrow \arg \max_a \sum_{s'} \mathcal{P}_{ss'}^{\pi_h(s)} [\mathcal{R}_{ss'}^{\pi_h(s)} + \gamma V(s')]$

If $b \neq \pi_h(s)$ then `policy_stable` \leftarrow false
 If `policy_stable`, then stop; else go to 2

The V calculated in the above algorithm is the V function for the refined user dialogue policy. The refined user dialogue policy models the dialogue behavior of the user who interacts with the SDS. In state s , V^{π_h} can be used to determine what information the user of the SDS most likely expresses. When ambiguity occurs in s , the knowledge about user dialogue behavior can be used along with other information to best understand user input.

4 Implementation and Experiments

At the time of this writing, we are performing the first stage implementation and experiments. The reinforcement learning algorithm described above has been implemented in a dialogue system that is for tutoring highschool geometry. In this stage, the voice recognition (VR) and text to speech (TTS) components are not included. Only text dialogues are tested. We perform two tests: when the online learning is disabled and activated.

The training data set consists of about 30 dialogue sessions in English, which are created to simulate the user and the system. Each session is regarding a geometric diagram, and contains several pairs of questions and answers. A session starts with a question from the system. If the user correctly answers the question, the system asks another question, otherwise, the system may ask the question again, give a hint, or tell the correct answer.

After the training, the first test is conducted with the online learning ability disabled. A user, who is not the author of the training sessions, tests the tutoring system. In this test, the dialogue manager can properly control a dialogue flow when the user answers a question in exactly the same way as the simulated user, or the answer can be converted into the semantic graph that is the same as the graph of a training answer. However, the system may get confused when the user answers a question in a different way, even though the answer is correct in terms of geometric knowledge. The problem is worse when an answer is ambiguous, for example “draw an auxiliary line perpendicular to the diagonal, which passes thought point D”.

In the second test, the online learning ability is activated. The system asks additional questions when it is uncertain about the answer. For example, it may ask “Did you say ...?”, or “Is this what you said?”. Such questions help the system learn the dialogue behavior of the user who actually uses the system. As explained before, the actual dialogues are used to improve the user dialogue policy that is learned in the offline training.

The initial results are encouraging. We have observed remarkable improvement of the system’s disambiguation ability. As the system becomes more and more experienced with the user, it understands better and better the user’s answers. The system can correctly interpret many ambiguous expressions after it deals with them several times. More experiments have been planned with a much

larger training data set with better designed dialogues. More test cases will be designed to examine and improve the algorithm's learning ability.

5 Concluding Remarks

In this research, we extend the traditional reinforcement learning algorithm to include a user dialogue policy. When using the extended reinforcement learning algorithm in an SDS to generate the optimal system dialogue strategy (policy), we can meanwhile create the user dialogue policy, which models the dialogue behavior of the user who actually converses with the system. The knowledge about dialogue behavior of the user can be very useful in handling ambiguity in user input. System dialogue performance can be dramatically improved when an SDS has better abilities to disambiguate and understand user input.

References

1. Frampton, M., Lemon, O.: Learning more effective dialogue strategies using limited dialogue move features. In: Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics, pp. 185–192. Association for Computational Linguistics, Sydney (2006)
2. Griol, D., Hurtado, L.F., Segarra, E., Sanchis, E.: A statistical approach to spoken dialog systems design and evaluation. *Speech Communication* 50, 666–682 (2008)
3. Jokinen, K., Kerminen, A., Kaipainen, M., Jauhainen, T., Wilcock, G., Turunen, M., Hakulinen, J., Kuusisto, J., Lagus, K.: Adaptive dialogue systems - interaction with interact. In: Proceedings of the 3rd SIGdial workshop on Discourse and dialogue, vol. 2, pp. 64–73. Association for Computational Linguistics, Philadelphia (2002)
4. Levin, E., Pieraccini, R., Eckert, W.: A stochastic model of human-machine interaction for learning dialog strategies. *IEEE Transactions on Speech and Audio Processing* 8, 11–23 (2000)
5. Litman, D.J., Kearns, M.S., Singh, S., Walker, M.A.: Automatic optimization of dialogue management. In: Proceedings of the 18th conference on Computational linguistics, vol. 1, pp. 502–508. Association for Computational Linguistics, Saarbrücken (2000)
6. Melichar, M., Cenek, P.: From vocal to multimodal dialogue management. In: Proceedings of the 8th international conference on Multimodal interfaces, pp. 59–67. ACM, Banff (2006)
7. Mitchell, T.M.: *Machine learning*. WCB McGraw-Hill, New York (1997)
8. Scheffler, K., Young, S.: Automatic learning of dialogue strategy using dialogue simulation and reinforcement learning. In: Proceedings of the second international conference on Human Language Technology Research, pp. 12–19. Morgan Kaufmann Publishers Inc., San Diego (2002)
9. Sutton, R.S., Barto, A.G.: *Reinforcement Learning: An Introduction*. MIT Press, Cambridge (2005)
10. Swegles, K., Wang, F.: A hybrid reinforcement learning algorithm for optimizing dialogue strategies in spoken dialogue systems. In: The 26th International Conference on Machine Learning (ICML 2009), Montreal, Canada (submitted, 2009)

A Fixed-Parameter Enumeration Algorithm for the Weighted FVS Problem*

Jianxin Wang and Guohong Jiang

School of Information Science and Engineering
Central South University, Changsha, 410083, China
jxwang@mail.csu.edu.cn

Abstract. In this paper, we present a fixed-parameter enumeration algorithm for the feedback vertex set problem by using the *branch-and-search* method. The algorithm transforms the feedback vertex set problem to the feedback edge set problem with specific conditions. Then it enumerates the z minimum-weight feedback edge sets by enumerating the z maximum-weight forests. As a result, we show the problem of enumerating the z minimum-weight feedback vertex sets of size k is solvable in time $\mathcal{O}(5^k kn^2 + (5^k + 3^k) n^2 \log n)$.

1 Introduction

A feedback vertex set (FVS) F in a graph G is a set of vertices in G whose removal makes the graph acyclic. The problem to find a minimum FVS in a graph is a classic NP-complete problem. There are numerous applications of the FVS problem in areas such as circuit testing, deadlock resolution, analyzing manufacturing processes, computational biology and so on.

For the parameterized FVS problem in undirected graphs, Downey, Fellows [6] and Bodlaender [2] gave the first group of algorithms to find a FVS of size at most k in $f(k)n^{\mathcal{O}(1)}$ time. Since then many dramatic improvements were obtained [7,10,9,11,8,5,3]. The best previous parameterized algorithm for unweighted FVS problem, due to Chen et al [3], has a running time of $\mathcal{O}(5^k kn^2)$ using the *iterative compression* and the *branch-and-search* methods. Chen et al [3] also gave an algorithm with runtime $\mathcal{O}(5^k kn^2)$ to generate a minimum-weight FVS of size at most k in a weighted graph. Becker [1] developed randomized parameterized algorithms of running time $\mathcal{O}(4^k kn^2)$ for the FVS problem in unweighted graphs and $\mathcal{O}(6^k kn^2)$ for the FVS problem in weighted graphs respectively.

The fixed-parameter enumeration has lately attracted some interest [4]. A NP optimization problem is *fixed-parameter enumerable* if there is an algorithm that, for a given problem instance (x, k) and an integer z , generates the z best (in terms of the solution weight) solutions of size k to x in time $f(k)n^{\mathcal{O}(1)}z^{\mathcal{O}(1)}$.

* This work is supported by the National Grand Fundamental Research 973 Program of China (No.2008CB317107), the National Natural Science Foundation of China (No.60773111).

The fixed-parameter enumeration method has been effectively used to solve some problems, for example k -vertex cover, k -path and k -planar dominating set [4].

In this paper, the fixed-parameter enumeration algorithm for FVS problem is discussed, which is defined as follows: given an undirected weighted graph $G = (V, E)$ on n vertices with vertex weight, and integers k and z , generate the z minimum-weight FVS of size k in G .

We show the problem of enumerating the z minimum-weight FVS of size k is solvable in time $\mathcal{O}(5^k kn^2 + (5^k + 3^k z)n^2 \log n)$. Firstly we get a FVS F of size at most k with the algorithm of [3]. Then we enumerate all the subsets of F , and use each subset F_1 of F to construct a partition structure of the graph. A group of small structures is obtained with the *branch-and-search* method on a partition, so that the FVS problem for the small structures can be transformed to the feedback edge set (shortly as FES, which is a set of edges in G , whose removal results the graph acyclic). At last we enumerate the z minimum-weight FES by enumerating the z maximum-weight forests.

2 Definitions and Preliminaries

Let $G = (V, E)$ be an undirected weighted graph on n vertices, where each vertex $u \in V$ is associated with a positive real number (the vertex weight). For $W \subseteq V$, $G[W]$ denotes the graph induced by W . Let $c[W]$ be the number of connected components in $G[W]$. For a vertex $w \in V$, $G \setminus w$ denotes the graph induced by $V \setminus w$. The size of the set W is the number of elements in W and the weight of the set W is the sum of all the element weights in W . Let k -FVS (k -FES) denote the FVS (FES) of size k , and k -forest denote the forest containing k edges.

Definition 1. [3] A triple (V_0, V_1, V_2) is an independent-forest partition (shortly, IF-partition) of a graph G if (V_0, V_1, V_2) is a partition of V such that: (1) $V_0 \cup V_1 \cup V_2 = V$, and V_0, V_1 and V_2 are pairwise disjoint; (2) $G[V_1]$ and $G[V_2]$ are forests; (3) V_0 is an independent set; (4) Every vertex v in V_0 is of degree 2 in G , with both neighbors in V_2 .

The FVS contained within $V_0 \cup V_1$ is called a FVS F on an IF-partition $I = (V_0, V_1, V_2)$, shortly as FVS(I).

Definition 2. A Compress-graph H for an IF-partition (V_0, V_1, V_2) of G is a new graph $H = (V_H, E_H)$, where each vertex u in V_H corresponds to a connected component in the graph $G[V_2]$, and each edge (u, v) in E_H corresponds to a vertex w in the set V_0 such that the two neighbors of w are in the connected components in $G[V_2]$ that correspond to the two vertices u and v , respectively, in H .

Assume a vertex v has degree 2 in G and vertices u_1 and u_2 are the neighbors of v . Bypassing v means deleting v from G and adding an edge between u_1 and u_2 in G . Intuitively, the Compress-graph H for an IF-partition (V_0, V_1, V_2) of G is obtained from the graph $G[V_0 \cup V_2]$ by shrinking each connected component in $G[V_2]$ into a single vertex and bypassing each degree-2 vertex in V_0 . The weight of an edge in H is equal to the weight of the corresponding vertex in V_0 .

Let two deficiencies $\tau(k, V_0, V_1, V_2) = k - (|V_0| - c[V_2] + 1)$ and $\varphi(k, V_0, V_1, V_2) = k - (|V_0| - c[V_2] + c[V_0 \cup V_2])$. Assume T is a spanning tree or spanning forest in *Compress-graph* H for an IF-partition $I = (V_0, V_1, V_2)$. The set of edges in H which are not in T is the minimum-size FES in H and the set of vertices corresponding to the edges in the FES is also the minimum-size FVS(I) in $G[V_0 \cup G_2]$. There are $|V_0|$ edges and $c[V_2]$ vertices in H , and the number of the connected components in H is equal to that in $G[V_0 \cup V_2]$. So there are $c[V_2] - c[V_0 \cup V_2]$ edges in T , and $|V_0| - c[V_2] + c[V_0 \cup V_2]$ edges in the minimum-size FES in H . Therefore at least $|V_0| - c[V_2] + c[V_0 \cup V_2]$ vertices of a k -FVS(I) in G come from V_0 . Obviously, both $\tau(k, V_0, V_1, V_2)$ and $\varphi(k, V_0, V_1, V_2)$ of the IF-partition I are upper bounds on the number of vertices in a k -FVS(I) that are in the set V_1 , but $\varphi(k, V_0, V_1, V_2)$ is more exact than $\tau(k, V_0, V_1, V_2)$.

Lemma 1. [3] *The FVS problem on n vertex graph is solvable in time $\mathcal{O}(5^k kn^2)$.*

3 The Fixed-Parameter Enumeration Algorithm for FVS Problem

The fixed-parameter enumeration algorithm for FVS problem is designed as follows. Firstly a k -FVS of a graph is found to generate a set of IF-partitions. Secondly the *Tuple-construct* algorithm and *Local-enumeration* algorithm are used to find the z minimum-weight k -FVS(I) for each IF-partition I . At last the z minimum ones among those having been found are saved.

3.1 The Tuple-Construct Algorithm

Fig.1 is the *Tuple-construct* algorithm which returns a collection R of 7-tuples for an IF-partition $I = (V_0, V_1, V_2)$ of a graph G . A tuple $r = (G, V_0, V_1, V_2, F, A, S)$ in R must satisfies $V_2 = \emptyset$, or $V_1 = \emptyset$ and $\varphi(k, V_0, V_1, V_2) > 0$, or $\varphi(k, V_0, V_1, V_2) = 0$ and $G[V_1 \cup V_2]$ is a forest. The triple (V_0, V_1, V_2) in r is an IF-partition of the graph G . F is a subset of a k -FVS(I) in the primary graph. A is a set of degree-0 and degree-1 vertices in V_1 produced in the process of algorithm. S is a set of vertex sets such as S_v (the vertices in S_v are bypassed by the *Tuple-construct* algorithm and they can form a path in the primary graph). In order to talk about the “lowest” leaf in a tree in $G[V_1]$, a root is fixed for each tree in $G[V_1]$.

If the condition in Step 2 is satisfied, the recursive call is stopped and a set R containing the tuple is returned. If the condition in Step 3 is satisfied, any k -FVS(I) can't be found and a empty set R is returned.

In Step 4, if some vertices in V_1 have degree less than 2 in G , they can be deleted safely from graph G and saved in A . In Step 5, if a vertex w in V_1 has degree of 2 in G with both neighbors in V_2 , w is removed from V_1 to V_0 and the triple $(V_0 \cup w, V_1 \setminus \{w\}, V_2)$ is also a valid IF-partition of the graph G .

In Step 6, if a vertex w has at least 2 neighbors in V_2 and has degree at least 3 in G , there are two cases. If there are at least 2 neighbors of w in the same tree in $G[V_2]$, at least a circle exists in the graph $G[V_2 \cup \{w\}]$. The only way to break the circles in $G[V_2 \cup \{w\}]$ is to include the vertex w in the objective

Tuple-construct($G, V_0, V_1, V_2, k, F, A, S$)

Input: a graph $G = (V, E)$, an IF-partition (V_0, V_1, V_2) , and an integer k

Output: a set R , which contains 7-tuples $(G, V_0, V_1, V_2, F, A, S)$

1. $R = \emptyset$;
2. **if** $(V_2 = \emptyset)$ or $(V_1 = \emptyset$ and $\varphi(k - |F|, V_0, V_1, V_2) > 0)$ or $(\varphi(k - |F|, V_0, V_1, V_2) = 0$ and $G[V_1 \cup V_2]$ is a forest), **then** add the tuple $(G, V_0, V_1, V_2, F, A, S)$ to R and return R ;
3. **if** $(k < 0)$ or $(k = 0$ and $\ell(G) = 1)$ or $(\varphi(k, V_0, V_1, V_2) = 0$ and $G[V_1 \cup V_2]$ is not a forest) or $(\varphi(k, V_0, V_1, V_2) < 0)$, return R ;
4. **else if** a vertex w in V_1 has degree less than 2 in G **then**
return **Tuple-construct** $(G \setminus w, V_0, V_1 \setminus \{w\}, V_2, k, F, A \cup \{w\}, S)$;
5. **else if** a vertex w in V_1 has 2 neighbors in V_2 and has degree 2 in G **then**
return **Tuple-construct** $(G, V_0 \cup \{w\}, V_1 \setminus \{w\}, V_2, k, F, A, S)$;
6. **else if** a vertex w in V_1 has at least 2 neighbors in V_2 **then**
 - 6.1 **if** two neighbors of w belong to the same tree in $G[V_2]$ **then**
return **Tuple-construct** $(G \setminus w, V_0, V_1 \setminus \{w\}, V_2, k - 1, F \cup \{w\}, A, S)$;
 - 6.2 **else**
 $R_1 = \mathbf{Tuple-construct}(G \setminus w, V_0, V_1 \setminus \{w\}, V_2, k - 1, F \cup \{w\}, A, S)$;
 $R_2 = \mathbf{Tuple-construct}(G, V_0, V_1 \setminus \{w\}, V_2 \cup \{w\}, k, F, A, S)$;
 return $R_1 \cup R_2$;
7. **else** pick a lowest leaf w_1 in any tree T in $G[V_1]$, let w be the parent of w_1 in T , and let w_1, \dots, w_t be the children of w in T ;
 - 7.1 **if** (w has a neighbor in V_2) or (w has more than one child in T) **then**
 $R_1 = \mathbf{Tuple-construct}(G \setminus w, V_0, V_1 \setminus \{w\}, V_2, k - 1, F \cup \{w\}, A, S)$;
 $R_2 = \mathbf{Tuple-construct}(G, V_0 \cup \{w_1, \dots, w_t\}, V_1 \setminus \{w, w_1, \dots, w_t\}, V_2 \cup \{w\}, k, F, A, S)$;
 return $R_1 \cup R_2$;
 - 7.2 **else** Vertex w_1 in the set V_1 is colored with red and $S_{w_1} = S_{w_1} \cup \{w\}$.
 Let G_b be the result graph after bypassing vertex w , return
 $\mathbf{Tuple-construct}(G_b, V_0, V_1 \setminus \{w\}, V_2, k, F, A, (S \cup S_{w_1}))$;

Fig. 1. The Tuple-construct Algorithm

FVS(I). If the neighbors of w are all in different trees in $G[V_2]$, there are two branches. To include w in the k -FVS(I), the vertex w is removed from V_1 to F and $(k - 1)$ -FVS(I) are recursively found in $G \setminus w$. To exclude w from the k -FVS(I), w is removed from V_1 to V_2 .

In step 7, each vertex in V_1 has degree at least 2 in G and at most one neighbor in V_2 . A “lowest” leaf w_1 is fixed in a tree in $G[V_1]$. Then the vertex w_1 has degree exactly 2 in G . Let w be the parent of w_1 , and w_1, \dots, w_t be the children of w in T . Then all the children of w are leaves in T , and each of them has a unique neighbor in V_2 . In Step 7.1, if w has a neighbor in V_2 or w has more than one child, the *Tuple-construct* algorithm branches on w . Specially, to exclude the vertex w from the objective FVS(I), w is removed from V_1 to V_2 . Since all these degree-2 vertices w_1, \dots, w_t have their both neighbors in the set $V_2 \cup \{w\}$, they can be moved to V_0 safely. In Step 7.2, if w doesn’t have any

neighbor in V_2 and w has only one child, w has degree exactly 2 in G . Therefore, w_1 and w are two adjacent degree-2 vertices in G . In this case, a circle in G contains the vertex w_1 only if it contains w . The vertex w is bypassed and w is added into the vertex set S_{w_1} containing w_1 is searched in S . If there is no such a set S_{w_1} in S , a set $S_{w_1} = \{w_1, w\}$ is added into S . The vertex w_1 in the set V_1 is colored with red to distinguish w_1 from other vertices and its color doesn't change when it is moved into other sets. Moreover, no red vertex will be moved into V_2 and F according to the Step 4 and Step 5.

Lemma 2. *Let R be the collection returned by the *Tuple-construct* algorithm for a given IF-partition I of G . Then every k -FVS(I) in G is consistent with exactly one tuple $r = (G, V_0, V_1, V_2, F, A, S)$ in R (A FVS F' in G is consistent with a tuple r if F' contains all the vertices in F but no vertices in V_2).*

The number of tuples and the runtime of the *Tuple-construct* algorithm can be computed by counting the number of leaves in the search tree corresponding to the execution of the algorithm.

Lemma 3. *The algorithm $\text{Tuple-construct}(G, V_0, V_1, V_2, k, F, A, S)$ returns a tuple set R with at most $\mathcal{O}(2^{\tau(k, V_0, V_1, V_2)})$ tuples, and runs in time $\mathcal{O}(2^{\tau(k, V_0, V_1, V_2)}n^2)$.*

Proof. Let $T(k, V_0, V_1, V_2)$ be the number of leaves in the search tree for the *Tuple-construct* algorithm. Now we prove by induction that $T(k, V_0, V_1, V_2) \leq \max(1, 2^{\tau(k, V_0, V_1, V_2)})$.

Because $\tau(k, V_0, V_1, V_2) \geq \varphi(k, V_0, V_1, V_2)$, $\varphi(k, V_0, V_1, V_2) < 0$ if $\tau(k, V_0, V_1, V_2) < 0$. Then $T(k, V_0, V_1, V_2) = 1 \leq \max(1, 2^{\tau(k, V_0, V_1, V_2)})$ by step 3.

If step 6.2 is executed, we have $T(k, V_0, V_1, V_2) \leq T(k - 1, V_0, V_1 \setminus w, V_2) + T(k, V_0, V_1 \setminus w, V_2 \cup \{w\})$. Because w has at least two neighbors and they are in different trees in $G[V_2]$, adding w to V_2 merges at least two connected components in $G[V_2]$ and reduces the number of connected components by at least 1. So we can get $c[V_2 \cup \{w\}] \leq c[V_2] - 1$. Then $\tau_1 = \tau(k - 1, V_0, V_1 \setminus w, V_2) \leq \tau(k, V_0, V_1, V_2) - 1$, and $\tau_2 = \tau(k, V_0, V_1 \setminus w, V_2 \cup \{w\}) \leq \tau(k, V_0, V_1, V_2) - 1$. With the inductive hypothesis, $T(k - 1, V_0, V_1 \setminus w, V_2) \leq 2^{\tau_1}$, and $T(k, V_0, V_1 \setminus w, V_2 \cup \{w\}) \leq 2^{\tau_2}$. Therefore, we get $T(k, V_0, V_1, V_2) \leq 2^{\tau(k, V_0, V_1, V_2)}$.

As analyzed above, we can also get $T(k, V_0, V_1, V_2) \leq 2^{\tau(k, V_0, V_1, V_2)}$ if step 7.1 is executed. Steps 4, 5, 6.1, 7.2 in the algorithm are non-branching recursive. It can be verified that the instance deficiency τ is never increased for all non-branching recursive in the algorithm.

In conclusion, there are at most $2^{\tau(k, V_0, V_1, V_2)}$ tuples in R . We observe that the total number of executions of all steps of the algorithm is $\mathcal{O}(n)$ along each root-leaf path in the search tree, because each step either stops immediately, or reduces the size of the set V_1 by at least 1. All steps can runs in time $\mathcal{O}(n)$. Therefore, the *Tuple-construct* algorithm runs in time $\mathcal{O}(2^{\tau(k, V_0, V_1, V_2)}n^2)$. \square

3.2 The Local-Enumeration Algorithm

Let G' be the primary graph. A tuple $r = (G, V_0, V_1, V_2, F, A, S)$ is generated by the *Tuple-construct* algorithm, $k_1 = k - |F|$. A k_1 -FVS based on tuple r in the

graph $G'[V - F]$ is called as a k_1 -FVS(r) in $G'[V - F]$. The union set of F and a k_1 -FVS(r) in $G'[V - F]$ is a k -FVS in the graph G' based on r . The main task of the *Local-enumeration* algorithm is to enumerate k_1 -FVS(r) in $G'[V - F]$. Firstly it transforms the k_1 -FVS(r) problem in $G'[V - F]$ to k_1 -FES problem. Secondly it enumerates the z k_1 -FES by enumerating the z maximum-weight forests. Then the z minimum-weight k -FVS in G' based on r can be found.

Transform k_1 -FVS Problem to k_1 -FES Problem. A new graph on a tuple r , named *RCompress-graph* H_r must be constructed for transforming the k_1 -FVS(r) problem to k_1 -FES problem. There are 4 steps to construct H_r .

Step 1: Assume $r = (G, V_0, V_1, V_2, F, A, S)$ is a tuple in R and $k_1 = k - |F|$. If $V_2 = \emptyset$, all the vertices in V_1 are moved to A and V_1 is assigned with \emptyset . If $\varphi(k_1, V_0, V_1, V_2) = 0$, all the vertices in V_1 are moved to V_2 , both V_1 and A are assigned with \emptyset .

Step 2: Construct the *Compress-graph* H for IF-partition (V_0, V_1, V_2) contained in the tuple r .

Step 3: For each red-color vertex w (which is colored with red by the step 7.2 of the *Tuple-construct* algorithm) in V_0 , find the element S_w containing w in S . If the edge (u, v) corresponds with w in H then delete the edge (u, v) from H and add $|S| - 1$ new vertices $u_1, u_2, \dots, u_{|S|-1}$ and $|S|$ edges $(u, u_1), (u_1, u_2), \dots, (u_{|S|-1}, v)$ into H , where the weights of the new $|S|$ edges are orderly equal to the weights of vertices $w, w_1, \dots, w_{|S|-1}$ in S .

Step 4: For each red-color vertex w in A , find the element S_w containing w in S . All the vertices of S_w are added into A . Then if $A = \{w_1, w_2, \dots, w_{|A|}\}$, $2|A|$ new vertices $u_1, v_1, \dots, u_{|A|}, v_{|A|}$, and $|A|$ edges $(u_1, v_1), \dots, (u_{|A|}, v_{|A|})$ are added into H , where the weights of the new $|A|$ edges are orderly equal to the weights of vertices $w_1, \dots, w_{|A|}$ in A .

Let V_S be the set of vertices contained in the elements of S and V'_S be the union of the sets in S containing red-color vertices in A . If there are d red-color vertices in A , then there are $|E_r| = |A \cup V_0 \cup V_S| \leq n$ edges and $|V_r| = c|V_2| + (|V_S - V'_S| - (|S| - d)) + 2|A \cup V'_S| \leq 2n$ in H_r , where n is the number of vertices in G' .

Lemma 4. *Given a graph G' , an integer k , and a tuple $r = (G, V_0, V_1, V_2, F, A, S)$. Let $k_1 = k - |F|$. The problem of enumerating the z minimum-weight k_1 -FVS(r) in the graph $G'[V - F]$ is equivalent to that of enumerating the z minimum-weight k_1 -FES in *RCompress-graph* H_r .*

Proof. Let V'_S be the union of the sets in S which contains red-color vertices in A . There are at most $\varphi(k, V_0, V_1, V_2)$ vertices of a k_1 -FVS(r) in V_1 . If $V_2 = \emptyset$, then $V_0 = \emptyset$ and $G = G[V_1]$. In this case, there is no circle in G , so the vertices of V_1 can be moved into A . If $\varphi(k_1, V_0, V_1, V_2) = 0$ in the tuple r , all vertices of k_1 -FVS(r) in $G'[V - F]$ must come from V_0 . In this case, the vertices of V_1 can be moved into V_2 and $V_1 = \emptyset$.

Let G_0 be the graph by deleting the edges associated with the vertices in $A \cup V'_S$ from $G'[V - F]$. G_0 can also be formed by adding the vertices in $A \cup V'_S$ as degree-0 vertices into graph $G[V_0 \cup V_2 \cup (V_S - V'_S)]$. The vertices in A don't join in any circle in G' . It is obvious that a vertex set F' containing no vertex in V_2 is a FVS of graph G_0 only if F' is a FVS(r) in the graph $G'[V - F]$.

By the construction of H_r , every vertex in $V_0 \cup V_1 \cup A \cup V_S$ corresponds with exact one edge in H_r . For every red-color vertex w in V_0 , let S_w be the set containing w . According to the *Tuple-construct* algorithm, every vertex in S_w has a degree of 2 in graph $G[V_0 \cup V_2 \cup (V_S - V'_S)]$. Thus all the vertices in S_w have degree of 2 in G_0 . Moreover, the path P formed by vertices in S_w in graph G_0 corresponds with a path P_H in H_r . Therefore, there is a one-to-one correspondence between the connected components in graph G_0 and H_r . Thus the problem of finding all the k_1 -FVS containing no vertex in V_2 in the graph G_0 is equivalent to that of finding all k_1 -FES in H_r .

In conclusion, the problem of enumerating the z minimum-weight k_1 -FVS(r) in the graph $G'[V - F]$ is equivalent to that of enumerating the z minimum-weight k_1 -FES in H_r . □

Enumerate the z Maximum-Weight Forests. Enumerating the z minimum-weight k_1 -FES of H_r can be solved by enumerating the z maximum-weight ($|E_r| - k_1$)-forests in the graph H_r . For each $(|E_r| - k_1)$ -forest T , the set F_e of the edges which are not in T is the k_1 -FES in H_r .

Suppose $X = \{(i_1, j_1), \dots, (i_r, j_r)\}$ and $Y = \{(m_1, p_1), \dots, (m_h, p_h)\}$. If a forest T with size t contains all of the edges in X but doesn't contain any edge in Y , the forest T is denoted as a t -forest based on pair (X, Y) . The maximum-weight t -forest based on (X, Y) is indicated as $T(X, Y)$. The idea of finding $T(X, Y)$ is the same with that of Kruskal's algorithm for finding the minimum-weight spanning tree in a graph, which repeats the following step until the set T contains t edges: add to T the maximum-weight edge which doesn't form a circle with edges in T .

Lemma 5. *Given a graph $H_r(V_r, E_r)$, an integer t and a pair (X, Y) , where X and Y are disjoint subsets of E_r . It takes $\mathcal{O}(|E_r| \log |V_r|)$ time to generate the maximum-weight t -forest $T(X, Y)$.*

Suppose $T(X, Y) - X = \{e_1, \dots, e_h\}$. A set of new pairs is generated by $T(X, Y)$ as follows. For all integers $i, 1 \leq i \leq h, X_i = X \cup \{e_1, \dots, e_{i-1}\}$ and $Y_i = Y \cup \{e_i\}$, which form a new pair (X_i, Y_i) . Then there will be h new pairs with the following forms: $(X_1, Y_1) = (X, Y \cup \{e_1\})$, $(X_2, Y_2) = (X \cup \{e_1\}, Y \cup \{e_2\})$, \dots , $(X_h, Y_h) = (X \cup \{e_1, \dots, e_{h-1}\}, Y \cup \{e_h\})$. All the sets of the t -forests based on each pair are mutually disjoint. All void pairs on which any forest can't be found are deleted. Then the set of t -forests based on (X, Y) is the union of $T(X, Y)$ and the t -forests based on all these new pairs.

It is clear that the set of t -forests based on (\emptyset, \emptyset) contains all the t -forests in graph H_r . Let $L = \{(\emptyset, \emptyset)\}$. The following steps are repeated for z times to enumerate the z maximum-weight t -forests: (1) find the maximum-weight t -forest based on every pair in L ; (2) suppose $T(X_j, Y_j)$ is the maximum forest

among those have been found in (1), then generate a group of new pairs by $T(X_j, Y_j)$, add the valid ones into L and delete (X_j, Y_j) from L .

Lemma 6. *Given a graph $H_r(V_r, E_r)$, integers t and z . It takes $\mathcal{O}(zt|E_r| \log |V_r|)$ time to generate the z maximum-weight t -forests in H_r .*

Proof. The *Enumerate-Forest* algorithm creates at most t new pairs every loop of (2) and generates the i maximum-weight t -forests after the i -th loop. It executes the loop z times to generate the z maximum-weight t -forests, so there are at most zt pairs at the z -th loop. With Lemma 5, generating the z maximum-weight t -forests takes time $\mathcal{O}(zt|E_r| \log |V_r|)$. □

The Practical Local-Enumeration Algorithm. As showed in Fig. 2, the *RCompress-graph* $H_r = (V_r, E_r)$ on the tuple r is constructed firstly. Secondly the the z maximum-weight $(|E_r| - k_1)$ -forests in H_r are enumerated by the *Enumerate-Forest* algorithm, where $k_1 = k - |F|$. For each forest T , the set F_T of edges in H_r , which doesn't contain any edge in T , is the k_1 -FES in H_r . Then all vertices which correspond to the edges in F_T in $V_0 \cup A \cup V_S$ are added into F' . Thus the union set of each F' and F is a k -FVS of G' based on the tuple r . According to Lemma 6, it's easy to verified the following lemma.

Local-enumeration (G', r, k, z)
Input: a graph G' , an integer k , and a tuple $r = (G, V_0, V_1, V_2, F, A, S)$
Output: z smallest k -FVS of G' based on the tuple r

1. $k_1 = k - |F|, U = \emptyset;$
2. construct the *RCompress-graph* $H_r = (V_r, E_r)$ on the tuple $r;$
3. find the z maximum-weight $(|E_r| - k_1)$ -forest in H_r and put them into $U_T;$
4. **for** each forest T in U_T **do**
5. F_T is the set of edges in H_r but not in T , and F' is the set of vertices corresponding to the edges in F_T in $V_0 \cup A \cup V_S$, then add $F \cup F'$ into $U;$
6. return $U;$

Fig. 2. The Local-enumeration Algorithm

Lemma 7. *Given a graph G' , an integer k and a tuple r in R generated by the Tuple-construct algorithm, the Local-enumeration algorithm enumerates the z minimum-weight k -FVS of G' based on the tuple r in time $\mathcal{O}(zn^2 \log n)$.*

3.3 The Complexity of the Fixed-Parameter Enumeration Algorithm for FVS Problem

The main process of the enumeration algorithm for FVS problem is to find the z minimum-weight k -FVS(I) for each IF-partition I . The following theorem is obtained by analyzing the former two algorithms.

Theorem 1. *Given a IF-partition $I = (V_0, V_1, V_2)$, the z minimum-weight k -FVS(I) can be generated in time $\mathcal{O}((2^{\tau(k, V_0, V_1, V_2)} + 1.414^{\tau(k, V_0, V_1, V_2)} z)n^2 \log n)$.*

Proof. Firstly, the *Tuple-construct* algorithm on I is executed and R is the returned tuple set. Let N be the total number of tuples in R . Secondly, the z_1 minimum-weight k -FVS based on each tuple in R are constructed. U_1 is the z minimum-weight k -FVS among all these Nz_1 k -FVS. This takes time $\mathcal{O}(Nz_1n^2 \log n)$. For a tuple r whose z_1 minimum consistent k -FVS are not all in U_1 , only those k -FVS consistent with r and that are already in U_1 can be possibly among the z minimum k -FVS of the graph G . Thus, the tuples whose z_1 minimum consistent k -FVS(r) are not all in U_1 can be discarded. With Lemma 2, a k -FVS is consistent with exact one tuple in R , so there are at most $N_1 = z/z_1$ tuples in R for which the z_1 minimum consistent k -FVS are all in U_1 . Now in time $\mathcal{O}(N_1zn^2 \log n)$, the *Local-enumeration* algorithm is applied to each of these N_1 tuples and the z minimum-weight k -FVS based on each tuple are generated. The returned k -FVS in this step are put into U_2 . Now the z minimum-weight k -FVS in $U_1 \cup U_2$ are the z minimum-weight k -FVS(I) in the graph G . In summary, the z minimum-weight k -FVS(I) in the graph G can be generated in time $\mathcal{O}((Nz_1 + N_1z)n^2 \log n)$.

If $z > \sqrt{N}$, let $z_1 = z/\sqrt{N}$. Then N_1 is bounded by \sqrt{N} and the above process runs in time $\mathcal{O}(\sqrt{N}zkn^2 \log n)$. If $z \leq \sqrt{N}$, let $z_1 = 1$. Then $N_1 = z$, and the above process runs in time $\mathcal{O}((N + z^2)n^2 \log n) = \mathcal{O}((N + \sqrt{N}z)n^2 \log n)$. There are at most $2^{\tau(k, V_0, V_1, V_2)}$ in R , so $N \leq 2^{\tau(k, V_0, V_1, V_2)}$, and $\sqrt{N} \leq 1.414^{\tau(k, V_0, V_1, V_2)}$. Moreover, the *Tuple-construct* algorithm runs in time $\mathcal{O}(2^{\tau(k-j, V_0, V_1, V_2)}n^2)$. In conclusion, the z minimum-weight k -FVS(I) in the graph G can be generated in time $\mathcal{O}((2^{\tau(k, V_0, V_1, V_2)} + 1.414^{\tau(k, V_0, V_1, V_2)} z)n^2 \log n)$. □

Using theorem 1, we obtain the main result of this paper.

Theorem 2. *Given a weighted graph $G = (V, E)$, the problem for Enumeration of FVS is solvable in time $\mathcal{O}(5^kkn^2 + (5^k + 3^kz)n^2 \log n)$.*

Proof. For a given FVS F' of size k in the graph $G = (V, E)$, each FVS F of size k for G is a union of a subset F_1 of j vertices in F' and a subset F_2 of $k - j$ vertices in $V \setminus F'$ for all integers $j, 0 \leq j \leq k$. All subsets F_1 of j vertices in F' are enumerated, so $G[V \setminus F']$ and $G[F' \setminus F_1]$ are forests and subgraphs of $G_0 = G - F_1$. Then the z minimum-weight $(k - j)$ -FVS of G_0 are constructed. Note that there is a special IF-partition $I = (V_0, V_1, V_2)$ of G_0 , where $V_0 = \emptyset, V_1 = V \setminus F'$, and $V_2 = F' \setminus F_1$. With Theorem 1, the z minimum-weight $(k - j)$ -FVS(I) in G_0 can be constructed in time $\mathcal{O}((2^{\tau(k-j, V_0, V_1, V_2)} + 1.414^{\tau(k-j, V_0, V_1, V_2)} z)n^2 \log n) = \mathcal{O}((4^{k-j} + 2^{k-j}z)n^2 \log n)$, where we have used the fact that and the fact $\tau(k - j, V_0, V_1, V_2)$ is bounded by $2(k - j)$. Now for all integers $j, 0 \leq j \leq k$, the running time of this process is

$$\sum_{j=0}^k \left(\binom{k}{j} \cdot \mathcal{O}((4^{k-j} + 2^{k-j}z)n^2 \log n) \right) = \mathcal{O}((5^k + 3^kz)n^2 \log n).$$

According to Lemma 1, a FVS F' of size k in the graph $G = (V, E)$ can be found in time $\mathcal{O}(5^k kn^2)$. Therefore, for a weighted graph G , the z minimum-weight k -FVS can be generated in time $\mathcal{O}(5^k kn^2 + (5^k + 3^k Z)n^2 \log n)$. \square

4 Discussion

There are two main subroutines for the fixed-parameter enumeration for FVS, the *Tuple-construct* algorithm and the *Local-enumeration* algorithm. The former uses the *branch-and-search* method to generate a set of tuples. The latter enumerates the z minimum-weight k -FVS based on a tuple. Specially, when the *Local-enumeration* algorithm transforms the k_1 -FVS problem to the k_1 -FES problem, it considers the vertices in A at the same time. If k is equal to the minimum size of FVS in the graph G' or if the z minimal FVS with minimum weight of size at most k is needed, the vertices in A can be discarded. Then the size of H_r will be smaller. The process of enumerating the z maximum forests can also be changed to enumerating the z maximum spanning-forests by [12].

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. In: *Proceedings of the 17th International Workshop*, pp. 230–238. Springer, London (1992)
3. Chen, J., Fomin, F., Liu, Y., Lu, S., Villanger, Y.: Improved algorithms for the feedback vertex set problems. In: Dehne, F., Sack, J.-R., Zeh, N. (eds.) *WADS 2007*. LNCS, vol. 4619, pp. 422–433. Springer, Heidelberg (2007)
4. Chen, J., Kanj, I., Meng, J., Xia, G., Zhang, F.: On the effective enumerability of NP problems. In: Bodlaender, H.L., Langston, M.A. (eds.) *IWPEC 2006*. LNCS, vol. 4169, pp. 215–226. Springer, Heidelberg (2006)
5. Dehne, F., Fellows, M., Langston, M., Rosamond, F., Stevens, K.: An $\mathcal{O}(2^{o(k)} n^3)$ FPT algorithm for the undirected feedback vertex set problem. *Theory Comput. Syst.* 41(3), 479–492 (2007)
6. Downey, R., Fellows, M.: Fixed parameter tractability and completeness. In: *Complexity Theory: Current Research*, pp. 191–225. Cambridge University, Cambridge (1992)
7. Downey, R., Fellows, M.: *Parameterized complexity*. Springer, New York (1999)
8. 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)
9. Kanj, I., Pelsmajer, M., 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)
10. Raman, V., Saurabh, S., Subramanian, C.: 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)
11. Raman, V., Saurabh, S., Subramanian, C.: Faster fixed parameter tractable algorithms for finding feedback vertex sets. *ACM Transactions on Algorithms* 2(3), 403–415 (2006)
12. Sörensen, K., Janssens, G.: An algorithm to generate all spanning trees of a graph in order of increasing cost. *Pesquisa Operacional* 25(2), 219–229 (2005)

On the Tractability of Maximal Strip Recovery

Lusheng Wang¹ and Binhai Zhu²

¹ Department of Computer Science, City University of Hong Kong,
Kowloon, Hong Kong

lwang@cs.cityu.edu.hk

² Department of Computer Science, Montana State University,
Bozeman, MT 59717-3880, USA

bhz@cs.montana.edu

Abstract. Given two genomic maps G and H represented by a sequence of n gene markers, a *strip* (syntenic block) is a sequence of distinct markers of length at least two which appear as subsequences in the input maps, either directly or in reversed and negated form. The problem *Maximal Strip Recovery* (MSR) is to find two subsequences G' and H' of G and H , respectively, such that the total length of disjoint strips in G' and H' is maximized (or, conversely, the number of markers hence deleted, is minimized). Previously, besides some heuristic solutions, a factor-4 polynomial-time approximation is known for the MSR problem; moreover, several close variants of MSR, MSR- d (with $d > 2$ input maps), MSR-DU (with marker duplications) and MSR-WT (with markers weighted) are all shown to be NP-complete. Before this work, the complexity of the original MSR problem was left open. In this paper, we solve the open problem by showing that MSR is NP-complete, using a polynomial time reduction from One-in-Three 3SAT. We also solve the MSR problem and its variants exactly with FPT algorithms, i.e., showing that MSR is fixed-parameter tractable. Let k be the minimum number of markers deleted in various versions of MSR, the running time of our algorithms are $O(2^{2.73k}n + n^2)$ for MSR, $O(2^{2.73k}dn + dn^2)$ for MSR- d , and $O(2^{5.46k}n + n^2)$ for MSR-DU.

1 Introduction

A well-known problem in comparative genomics is to decompose two given genomes into syntenic blocks—segments of chromosomes which are deemed to be homologous in the two input genomes. Many methods have been proposed, but they are very vulnerable to ambiguities and errors. Recently, a heuristic method was first proposed to eliminate noise and ambiguities in genomic maps, through handling a problem called Maximal Strip Recovery (MSR) (see below for the formal definition) [5,14]. In [3], a factor-4 polynomial-time approximation algorithm was proposed for the problem, and several close variants of the problem were shown to be intractable. It was left as an open problem whether the problem can be solved in polynomial time or is NP-complete.

In this paper, we show that MSR is in fact NP-complete, via a polynomial time reduction from One-in-Three 3SAT (which was shown to be NP-complete

in [12,8]). On the other hand, we show that MSR, together with its close variants MSR- d and MSR-DU, is fixed-parameter tractable. More specifically, let k be the minimum number of markers deleted in various versions of MSR, the running time of our algorithms are $O(2^{2.73k}n+n^2)$ for MSR, $O(2^{2.73k}dn+dn^2)$ for MSR- d , and $O(2^{5.46k}n+n^2)$ for MSR-DU respectively.

A genomic map is represented by a sequence of gene markers, and a gene marker can appear in several different genomic maps, in either positive or negative form. A *strip* (syntenic block) is a sequence of distinct markers that appears as subsequences in two or more maps, either directly or in reversed and negated form. Given two genomic maps G and H , the problem *Maximal Strip Recovery* (MSR) [5,14] is to find two subsequences G' and H' of G and H , respectively, such that the total length of disjoint strips in G' and H' is maximized. Intuitively, those gene markers not included in G' and H' are noise and ambiguities.

We give a precise formulation of the generalized problem MSR- d : Given d signed permutations (genomic maps) G_i of $\langle 1, \dots, n \rangle$, $1 \leq i \leq d$, find q sequences (strips) S_j of length at least two, and find d signed permutations π_i of $\langle 1, \dots, q \rangle$, such that each sequence $G'_i = S_{\pi_i(1)} \dots S_{\pi_i(q)}$ (here S_{-j} denotes the reversed and negated sequence of S_j) is a subsequence of G_i , and the total length of the strips S_j is maximized. Note that the problem Maximal Strip Recovery (MSR) [5,14] corresponds to the problem MSR-2 in our new formulation. We refer to Fig. 1 for an example. In this example, each integer represents a marker.

$$\begin{aligned}
 G_1 &= \langle 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 \rangle \\
 G_2 &= \langle -9, -4, -7, -6, 8, 1, 3, 2, -12, -11, -10, -5 \rangle \\
 S_1 &= \langle 1, 2 \rangle \\
 S_2 &= \langle 6, 7, 9 \rangle \\
 S_3 &= \langle 10, 11, 12 \rangle \\
 \pi_1 &= \langle 1, 2, 3 \rangle \\
 \pi_2 &= \langle -2, 1, -3 \rangle \\
 G'_1 &= \langle 1, 2, 6, 7, 9, 10, 11, 12 \rangle \\
 G'_2 &= \langle -9, -7, -6, 1, 2, -12, -11, -10 \rangle
 \end{aligned}$$

Fig. 1. An example for the problem MSR

A heuristic based on Maximum Clique (and its complement Maximum Independent Set) was previously given for the problem MSR (MSR-2) [5,14], which does not guarantee finding the optimal solution. It was shown that this heuristic [5,14] can be modified to achieve a factor-4 approximation for MSR-2 and, in general, a factor- $2d$ approximation for MSR- d . This was done by converting the problem to computing the maximal independent set in t -interval graphs, which admit a factor- $2t$ approximation [1].

In biological data, duplicate markers are possible in some genomic maps, as the so-called paralogy set. We denote by *MSR-DU* the problem MSR with the following variation DU:

DU — Duplicate markers are allowed in the genomic maps and in different strips.

It should be noted that while duplicate markers are allowed in the genomic maps and in *different* strips in the variation MSR-DU, they cannot appear in any individual strip since each strip must be composed of a sequence of distinct markers.

Sometimes, when building genomic maps, a priori information about the gene markers can be derived from comparative analysis. For example, certain genes that are responsible for important genetic functions in several closely related species can often be identified. It is reasonable to give the corresponding gene markers larger weights. Denote by *MSR-WT* the problem MSR with the following additional weight constraint WT:

WT — The total weight of markers in the strips is between two positive integers w_1 and w_2 .

This paper is organized as follows. In Section 2, we show NP-completeness for MSR. In Section 3, we present fixed-parameter algorithms for MSR and some of its variants. In Section 4, we conclude the paper with a few open questions.

2 MSR Is NP-Complete

We prove MSR to be NP-complete in this section. It is clear that MSR is in NP. We show that MSR is NP-hard by a reduction from the NP-hard problem One-in-Three 3SAT [12].

Theorem 1. *MSR is NP-complete.*

Proof. We reduce from the NP-complete problem One-in-Three 3SAT to MSR. Let $\phi = f_1 \wedge f_2 \wedge \dots \wedge f_m$ be an One-in-Three 3SAT instance, i.e., a boolean formula of m clauses in conjunctive normal form, with n variables v_1, v_2, \dots, v_n , where each clause f_k is the disjunction of exactly three distinct literals, like $(v_2 \vee v_5 \vee \bar{v}_7)$. The truth assignment satisfies another constraint that exactly one literal in each clause is set to true. In the above clause, $v_2 = \text{false}$, $v_5 = \text{true}$, and $v_7 = \text{true}$ is a valid one-in-three truth assignment. We assume that both $m, n > 2$.

Our construction uses $11m + 4n + 30n^2m + 15nm^2$ distinct markers:

- $9m$ clause markers — $f_{i,j,k}^1, f_{i,j,k}^2, f_{i,j,k}^3$, if v_i appears as the j -th literal in f_k ; $\bar{f}_{i,j,k}^1, \bar{f}_{i,j,k}^2, \bar{f}_{i,j,k}^3$, if \bar{v}_i appears as the j -th literal in f_k , for $1 \leq i \leq n, 1 \leq j \leq 3, 1 \leq k \leq m$,
- $2m$ clause markers a_i and \dot{a}_i for $1 \leq i \leq m$,
- $2n$ variable markers x_i and \dot{x}_i for $1 \leq i \leq n$,
- $2n$ variable markers y_i and \dot{y}_i for $1 \leq i \leq n$,
- m peg strings (of $15nm$ markers each) Z_k for $1 \leq k \leq m$, with $Z_k = z_{k,1}z_{k,2} \dots z_{k,15nm}$.

- n peg strings (of $15nm$ markers each) U_i for $1 \leq i \leq n$, with $U_i = u_{i,1}u_{i,2} \dots u_{i,15nm}$.
- n peg strings (of $15nm$ markers each) W_i for $1 \leq i \leq n$, with $W_i = w_{i,1}w_{i,2} \dots w_{i,15nm}$.

Throughout this proof, all of the peg strings are used to enforce the truth assignment and, as will be shown a bit later, no peg string is ever deleted to obtain the optimal solution for any converted MSR instance.

For the ease of description, we simply say that $A_{i,j,k} = f_{i,j,k}^1 f_{i,j,k}^2 f_{i,j,k}^3$ ($\bar{f}_{i,j,k}^1 \bar{f}_{i,j,k}^2 \bar{f}_{i,j,k}^3$) are the *associates* of v_i (\bar{v}_i) in f_k and they always appear together in one of the input map G and in the final optimal solution (— but not in the other input map H , as will be explained a bit later). For each variable $v_i, 1 \leq i \leq n$, let F_i and \bar{F}_i , respectively, be the two sequences of clause associates in which the two literals v_i and \bar{v}_i appear:

$$F_i = f_{i,j_1,k_1}^1 f_{i,j_1,k_1}^2 f_{i,j_1,k_1}^3 f_{i,j_2,k_2}^1 f_{i,j_2,k_2}^2 f_{i,j_2,k_2}^3 \dots f_{i,j_p,k_p}^1 f_{i,j_p,k_p}^2 f_{i,j_p,k_p}^3$$

$$\bar{F}_i = \bar{f}_{i,j'_1,k'_1}^1 \bar{f}_{i,j'_1,k'_1}^2 \bar{f}_{i,j'_1,k'_1}^3 \bar{f}_{i,j'_2,k'_2}^1 \bar{f}_{i,j'_2,k'_2}^2 \bar{f}_{i,j'_2,k'_2}^3 \dots \bar{f}_{i,j'_q,k'_q}^1 \bar{f}_{i,j'_q,k'_q}^2 \bar{f}_{i,j'_q,k'_q}^3$$

let

$$X_i = -\dot{x}_i F_i - x_i y_i \bar{F}_i \dot{y}_i.$$

Given three sequences of length p , $B_1 = b_{11}b_{12} \dots b_{1p}$, $B_2 = b_{21}b_{22} \dots b_{2p}$, and $B_3 = b_{31}b_{32} \dots b_{3p}$, let $(B_1 \otimes B_2 \otimes B_3)$ be the sequence obtained by listing letters in B_1, B_2, B_3 alternately; i.e., $B_1 \otimes B_2 \otimes B_3 = b_{11}b_{21}b_{31}b_{12}b_{22}b_{32} \dots b_{1p}b_{2p}b_{3p}$. For each clause $f_k, 1 \leq k \leq m$, let

$$Y_k = a_k(A_{k_1,1,k} \otimes A_{k_2,2,k} \otimes A_{k_3,3,k})\dot{a}_k,$$

where $A_{k_j,j,k} = a_{k_j,j,k}^1 a_{k_j,j,k}^2 a_{k_j,j,k}^3$, with $a_{k_j,j,k} = f_{k_j,j,k}$ if v_{k_j} is the j -th literal in f_k or $a_{k_j,j,k} = \bar{f}_{k_j,j,k}$, if \bar{v}_{k_j} is the j -th literal in f_k , for $1 \leq j \leq 3$ and for some $1 \leq k_j \leq n$. More precisely,

$$Y_k = a_k a_{k_1,1,k}^1 a_{k_2,2,k}^1 a_{k_3,3,k}^1 a_{k_1,1,k}^2 a_{k_2,2,k}^2 a_{k_3,3,k}^2 a_{k_1,1,k}^3 a_{k_2,2,k}^3 a_{k_3,3,k}^3 \dot{a}_k.$$

Construct two genomic maps

$$G = W_1 \dots W_n X_1 U_1 \dots X_n U_n Z_1 \dots Z_m a_m \dot{a}_m \dots a_2 \dot{a}_2 a_1 \dot{a}_1,$$

$$H = x_1 y_1 \dot{x}_1 \dot{y}_1 W_1 \dots x_n y_n \dot{x}_n \dot{y}_n W_n Y_1 Z_1 \dots Y_m Z_m U_1 \dots U_n.$$

Note that G and H each contains the $11m + 4n + 30n^2m + 15nm^2$ distinct markers exactly once. We show that the one-in-three 3SAT formula ϕ is satisfiable if and only if G has a subsequence G' and H has a subsequence H' such that the total length of the strips in G' and H' is exactly $3m + 2n + 30n^2m + 15nm^2$.

We first prove the “only if” direction. Let τ be a truth assignment that satisfies ϕ . For each i , let

$$X'_i = \begin{cases} F'_i y_i \dot{y}_i & \text{if } \tau(v_i) = \text{true,} \\ -\dot{x}_i - x_i \bar{F}'_i & \text{if } \tau(v_i) = \text{false} \end{cases}$$

where F'_i and \bar{F}'_i are subsequences of F_i and \bar{F}_i respectively, which are related to the truth assignment. In short, F'_i is obtained from F_i by deleting the clause associates of v_i in f_k if $\tau(v_i) = \text{false}$. Similarly, \bar{F}'_i is obtained from \bar{F}_i by deleting the clause associates of \bar{v}_i in f_k if $\tau(v_i) = \text{true}$. We obtain Y'_k from Y_k by first deleting a_k and \dot{a}_k . Then, keep the associates of the (only) literal which sets f_k to be true. In other words, if f_k is satisfied, then $|Y'_k| = 3$. (If f_k is not satisfied, then $|Y'_k| = 2$; i.e., we will have to keep $Y'_k = a_k\dot{a}_k$ — that causes a much smaller solution for the MSR instance.)

Formally, as a literal can only appear in a clause exactly once

$$Y'_k = \begin{cases} f_{k_j,j,k}^1 f_{k_j,j,k}^2 f_{k_j,j,k}^3, & \text{if } v_{k_j} \text{ is the } j\text{-th literal in } f_k \text{ and } \tau(v_{k_j}) = \text{true,} \\ \bar{f}_{k_j,j,k}^1 \bar{f}_{k_j,j,k}^2 \bar{f}_{k_j,j,k}^3, & \text{if } \bar{v}_{k_j} \text{ is the } j\text{-th literal in } f_k \text{ and } \tau(v_{k_j}) = \text{false} \end{cases}$$

Then we have

$$G'' = W_1 \dots W_n \ X'_1 U_1 X'_2 U_2 \dots X'_n U_n \ Z_1 \dots Z_m,$$

and

$$H'' = x_1 y_1 \dot{x}_1 \dot{y}_1 W_1 \dots x_n y_n \dot{x}_n \dot{y}_n W_n \ Y'_1 Z_1 \dots Y'_m Z_m \ U_1 \dots U_n.$$

G' and H' are obtained from G'' and H'' as follows. G' and H' each contains exactly one of each of the variable strips $x_i \dot{x}_i$ and $y_i \dot{y}_i$ (with $y_i \dot{y}_i$ corresponding to true, and $x_i \dot{x}_i$ to false), and all of the peg strings (strips) U_i, W_i , and Z_k . F'_i and \bar{F}'_i are obtained by deleting the associates of all literals which do not make f_k true and hence have been deleted from Y_k (i.e., not appearing in Y'_k). The satisfying truth assignment also guarantees that each Y'_k contains exactly three associates corresponding to the true literal in clause f_k . Hence, the total length of the strips in G' and H' is exactly $(9m)/3 + (4n)/2 + 30n^2m + 15nm^2 = 3m + 2n + 30n^2m + 15nm^2$.

For example, an one-in-three 3SAT formula of the following four clauses (over four variables)

$$f_1 = (\bar{v}_1 \vee v_2 \vee \bar{v}_3) \quad f_2 = (v_1 \vee v_2 \vee \bar{v}_4) \quad f_3 = (v_2 \vee v_3 \vee v_4) \quad f_4 = (\bar{v}_1 \vee \bar{v}_2 \vee \bar{v}_4)$$

corresponds to the two genomic sequences

$$\begin{aligned} G = & W_1 W_2 W_3 W_4 \\ & -\dot{x}_1 f_{1,1,2}^1 f_{1,1,2}^2 f_{1,1,2}^3 -x_1 \ y_1 \bar{f}_{1,1,1}^1 \bar{f}_{1,1,1}^2 \bar{f}_{1,1,1}^3 \bar{f}_{1,1,4}^1 \bar{f}_{1,1,4}^2 \bar{f}_{1,1,4}^3 \dot{y}_1 U_1 \\ & -\dot{x}_2 f_{2,2,1}^2 f_{2,2,1}^2 f_{2,2,1}^3 f_{2,2,2}^1 f_{2,2,2}^2 f_{2,2,2}^3 f_{2,1,3}^2 f_{2,1,3}^3 -x_2 \\ & y_2 \bar{f}_{2,2,4}^1 \bar{f}_{2,2,4}^2 \bar{f}_{2,2,4}^3 \dot{y}_2 U_2 \\ & -\dot{x}_3 f_{3,2,3}^1 f_{3,2,3}^2 f_{3,2,3}^3 -x_3 \ y_3 \bar{f}_{3,3,1}^1 \bar{f}_{3,3,1}^2 \bar{f}_{3,3,1}^3 \dot{y}_3 U_3 \\ & -\dot{x}_4 f_{4,3,3}^1 f_{4,3,3}^2 f_{4,3,3}^3 -x_4 \ y_4 \bar{f}_{4,3,2}^1 \bar{f}_{4,3,2}^2 \bar{f}_{4,3,2}^3 \bar{f}_{4,3,4}^1 \bar{f}_{4,3,4}^2 \bar{f}_{4,3,4}^3 \dot{y}_4 U_4 \\ & Z_1 Z_2 Z_3 Z_4 a_4 \dot{a}_4 a_3 \dot{a}_3 a_2 \dot{a}_2 a_1 \dot{a}_1 \end{aligned}$$

$$\begin{aligned}
 H = & x_1y_1\dot{x}_1\dot{y}_1W_1 x_2y_2\dot{x}_2\dot{y}_2W_2 x_3y_3\dot{x}_3\dot{y}_3W_3 x_4y_4\dot{x}_4\dot{y}_4W_4 \\
 & a_1 \bar{f}_{1,1,1}^1 f_{2,2,1}^1 \bar{f}_{3,3,1}^1 \bar{f}_{1,1,1}^2 f_{2,2,1}^2 \bar{f}_{3,3,1}^2 \bar{f}_{1,1,1}^3 f_{2,2,1}^3 \bar{f}_{3,3,1}^3 \dot{a}_1 Z_1 \\
 & a_2 f_{1,1,2}^1 f_{2,2,2}^1 \bar{f}_{4,3,2}^1 f_{1,1,2}^2 f_{2,2,2}^2 \bar{f}_{4,3,2}^2 f_{2,2,2}^3 f_{1,1,2}^3 \bar{f}_{4,3,2}^3 \dot{a}_2 Z_2 \\
 & a_3 f_{2,1,3}^1 f_{3,2,3}^1 f_{4,3,3}^1 f_{2,1,3}^2 f_{3,2,3}^2 f_{4,3,3}^2 f_{2,1,3}^3 f_{3,2,3}^3 f_{4,3,3}^3 \dot{a}_3 Z_3 \\
 & a_4 \bar{f}_{1,1,4}^1 \bar{f}_{2,2,4}^1 \bar{f}_{4,3,4}^1 \bar{f}_{1,1,4}^2 \bar{f}_{2,2,4}^2 \bar{f}_{4,3,4}^2 \bar{f}_{1,1,4}^3 \bar{f}_{2,2,4}^3 \bar{f}_{4,3,4}^3 \dot{a}_4 Z_4 \\
 & U_1 U_2 U_3 U_4.
 \end{aligned}$$

The truth assignment

$$\tau(v_1) = \text{true} \quad \tau(v_2) = \text{false} \quad \tau(v_3) = \text{false} \quad \tau(v_4) = \text{true}$$

corresponds to

$$\begin{aligned}
 G' = & W_1 W_2 W_3 W_4 f_{1,1,2}^1 f_{1,1,2}^2 f_{1,1,2}^3 y_1 \dot{y}_1 U_1 - \dot{x}_2 - x_2 \bar{f}_{2,2,4}^1 \bar{f}_{2,2,4}^2 \bar{f}_{2,2,4}^3 U_2 \\
 & - \dot{x}_3 - x_3 \bar{f}_{3,3,1}^1 \bar{f}_{3,3,1}^2 \bar{f}_{3,3,1}^3 U_3 f_{4,3,3}^1 f_{4,3,3}^2 f_{4,3,3}^3 y_4 \dot{y}_4 U_4 Z_1 Z_2 Z_3 Z_4.
 \end{aligned}$$

and

$$\begin{aligned}
 H' = & y_1 \dot{y}_1 W_1 x_2 \dot{x}_2 W_2 x_3 \dot{x}_3 W_3 y_4 \dot{y}_4 W_4 \bar{f}_{3,3,1}^1 \bar{f}_{3,3,1}^2 \bar{f}_{3,3,1}^3 Z_1 f_{1,1,2}^1 f_{1,1,2}^2 f_{1,1,2}^3 Z_2 \\
 & f_{4,3,3}^1 f_{4,3,3}^2 f_{4,3,3}^3 Z_3 \bar{f}_{2,2,4}^1 \bar{f}_{2,2,4}^2 \bar{f}_{2,2,4}^3 Z_4 U_1 U_2 U_3 U_4,
 \end{aligned}$$

We do not list U_i, W_i and Z_k as they are just long sequences of distinct markers.

We next prove the “if” direction. Let G', H' be a subsequence of G, H respectively such that the total length of the strips in G' and H' is exactly $3m + 2n + 30n^2m + 15nm^2$. It is clear that all the peg strings (strips) U_i, W_i and Z_k must be in the optimal solution for the corresponding MSR instance. The reason is that if we break any strip in U_i, W_i or Z_k , say we want to use strip $y_1 y_2$ by deleting W_1 and U_1 , even if we somehow put all the $11m + 4n$ non-peg markers in the optimal solution, the optimal solution size hence obtained would be less than $30n^2m + 15nm^2 < 3m + 2n + 30n^2m + 15nm^2$. In fact, breaking any one of U_i, V_i or Z_k , which is of length $15nm$, will decrease the optimal solution size to below $30n^2m + 15nm^2$. This is because $11m + 4n < 15m + 15n < 15mn$, when $m, n > 2$.

The alternating pattern of the clause markers in Y_k and F_i, \bar{F}_i ensures that there is at most one common strip of length at most three between any Y_k and F_i, \bar{F}_i . If no strip of length three in Y_k is selected, then $a_k \dot{a}_k$ will be a strip of length two. Hence the length of the clause strips in the optimal solution will be less than $3m$. So, in the optimal solution for this MSR instance, if we have $3m$ of clause strips then we must have exactly one strip of length three from each Y_k and the three markers must belong to some clause associates to match the corresponding ones in some F_i, \bar{F}_i . Similarly, the alternating pattern of the variable markers and the corresponding peg markers in G and H ensures that in the optimal solution there are n variable strips of length two in G' and H' , that is, either $x_i \dot{x}_i$ or $y_i \dot{y}_i$ for $1 \leq i \leq n$.

Therefore, in the optimal solution for this MSR instance, we have a valid truth assignment for ϕ : if clause markers in F_i are in the solution, we set v_i as true; if clause markers in \bar{F}_i are in the solution, we set v_i as false. Obviously, this assignment will satisfy each clause exactly once. Therefore, the one-in-three 3SAT formula ϕ is satisfied by this truth assignment.

The reduction time is clearly $O((m + n)^3)$ time. This completes the proof of Theorem [□](#).

It should be noted that $-\dot{x}_i \cdots -x_i$ in F_i and \bar{F}_i could be changed to $x_i \cdots \dot{x}_i$ and the proof still works. So MSR is in fact NP-complete even when all the markers are of positive signs.

3 FPT Algorithms for MSR and Its Variants

In this section, we consider solving MSR with an FPT algorithm. Basically, an FPT algorithm for an optimization problem Π with optimal solution value k is an algorithm which solves the problem in $O(f(k)n^c)$ time, where f is any function only on k , n is the input size and c is some fixed constant not related to k . More details on FPT algorithms can be found in [\[7\]](#). We first prove the following lemma.

Lemma 1. *Before any marker is deleted, if xy or $-y - x$ appears in both G_1 and G_2 (or, if xy appears in G_1 and $-y - x$ appears in G_2 , and vice versa), then there is an optimal solution for MSR which has xy or $-y - x$ as a strip.*

Proof. Wlog, we only consider the case when xy appears in G_1 and $-y - x$ appears in G_2 . The cases when xy ($-y - x$) appears in both G_1 and G_2 are similar. Let the length-4 substring in G_1 containing xy be $p_1(x)xy s_1(y)$, and let the length-4 substring in G_2 containing xy be $p_2(y) - y - x s_2(x)$. We assume that $p_1(x) \neq -s_2(x)$ and $s_1(y) \neq -p_2(y)$, as otherwise the lemma is obviously true.

If x is deleted to obtain any optimal solution, then $p_1(x)y$ in G_1 is a breakpoint. The reason is that $p_2(y) - y$ and $-y s_2(x)$ in G_2 cannot be equal to $p_1(x)y$ or its signed reversal — the former is due to the positive sign on y in $p_1(x)y$, and the latter is due to $s_1(y) \neq -p_2(y)$. Similarly, $ys_1(y)$ in G_1 is a breakpoint (as $p_2(y) - y$ and $-y s_2(x)$ in G_2 cannot be equal to $p_1(x)y$ or its signed reversal). Therefore, when x is deleted the strip xy is destroyed, which is a contradiction. If y is deleted, the same argument follows.

If both x, y are deleted to obtain any optimal solution, we consider three cases.

1. If a maximal substring S_1 of G_1 ending at $p_1(x)$ and a maximal substring S'_1 of G_1 starting at $s_1(y)$ are strips of length at least two, then we can put x, y back, and delete $p_1(x), s_1(y)$ to obtain a solution of larger size.
2. If one of S_1, S'_1 (say, S_1 , which must be equal to $p_1(x)$) has length one, then we can delete S_1 , put x, y back to obtain a solution of larger size.
3. If both of S_1, S'_1 have length one, then we can delete $p_1(x), s_1(y)$, put x, y back to obtain a solution which is of the same size as the current optimal solution.

Hence, the lemma is proven. □

We note that the above lemma also holds when a strip is of length greater than two.

The above lemma gives us a kernelization procedure.

1. Identify a set of strips from the two sequences, without deleting any gene marker.
2. For each strip identified, change it to a letter in Σ_1 , with $\Sigma_1 \cap \Sigma = \emptyset$. Let the resulting sequences be G'_1, G'_2 .

Let Σ be the alphabet for the input maps G_1 and G_2 . Let Σ_1 be the set of new letters used in the kernelization process, with $\Sigma_1 \cap \Sigma = \emptyset$. We have the following lemmas.

Lemma 2. *There is an optimal MSR solution of size k for G_1 and G_2 if and only if the solution can be obtained by deleting k markers in Σ from G'_1 and G'_2 respectively.*

Proof. In the kernelization process, without deleting any gene marker, we change each (existing) strip into a letter in $\Sigma_1 - \Sigma$. Following Lemma 1, these letters in $\Sigma_1 - \Sigma$ will never be deleted to obtain an optimal solution for MSR. \square

Lemma 3. *In G'_1 (resp. G'_2), there are at most $3k$ letters (markers) in Σ .*

Proof. Following Lemma 2, the optimal solution for MSR is obtained by deleting markers (letters) only in Σ from G'_1 (resp. G'_2). For each letter x deleted, there are at most two other letters in Σ , preceding and succeeding x . Therefore, we have at most $3k$ letters in Σ in G'_1 (resp. G'_2). \square

Theorem 2. *There is an FPT algorithm for MSR which runs in $O(2^{2.73k}n + n^2)$ time.*

Proof. Following Lemma 2 and Lemma 3, we can choose k letters in Σ from G'_1, G'_2 . The number of choices, is hence bounded by

$$\binom{3k}{k} \approx 2^{2.73k},$$

using Stirling’s formula. For each choice, we can check whether it is valid, i.e., whether all remaining markers are in some strip in G'_1 and G'_2 . This can be done in linear time if we spend $O(n^2)$ time in advance, i.e., building a correspondence between all of the identical markers in G_1, G_2 . So the overall running time of the algorithm is $O(2^{2.73k}n + n^2)$ time. \square

It is obvious that the algorithm also works for MSR- d . For MSR-DU, the algorithm is similar. But we need to make $\binom{3k}{k}$ choices of letters in Σ from each of G'_1 and G'_2 . So the running time will be $O(2^{5.46k}n + n^2)$ time.

Corollary 1. *MSR- d can be solved in $O(2^{2.73k}dn + dn^2)$ time and MSR-DU can be solved in $O(2^{5.46k}n + n^2)$ time.*

For MSR-WT, if the weights for markers are arbitrary then obviously Lemma 11 does not hold anymore and the above algorithm will not work. But if the weights are set so that Lemma 11 still holds, e.g., the weights must be one or two, then we will still be able to obtain a similar result.

4 Concluding Remarks

We note that (the minimization version of) the MSR problem can be thought of as the complement of the problem MWIS in 2-interval graphs is also known as the problem *2-Interval Pattern* [13], which has been extensively studied [1, 2, 4, 6, 9, 10, 11, 13] because of its application to RNA secondary structure prediction. This probably explains why there is an FPT algorithm for MSR.

It would be interesting to know whether our FPT algorithms can be further improved. The running times we have obtained for MSR and its variants are not efficient enough to make them truly useful in practice. To make such an FPT algorithm practical for MSR datasets, which usually has k between 50 to 150, it must be more efficient.

Acknowledgment

Lusheng Wang is fully supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China [Project No. CityU 120905]. We also thank referees for several useful suggestions and comments.

References

1. Bar-Yehuda, R., Halldórsson, M.M., Naor, J.(S.), Shachnai, H., Shapira, I.: Scheduling split intervals. *SIAM Journal on Computing* 36, 1–15 (2006)
2. Blin, G., Fertin, G., Vialette, S.: Extracting constrained 2-interval subsets in 2-interval sets. *Theoretical Computer Science* 385, 241–263 (2007)
3. Chen, Z., Fu, B., Jiang, M., Zhu, B.: On recovering syntenic blocks from comparative maps. In: Yang, B., Du, D.-Z., Wang, C.A. (eds.) *COCOA 2008*. LNCS, vol. 5165, pp. 319–327. Springer, Heidelberg (2008)
4. Chen, E., Yang, L., Yuan, H.: Improved algorithms for largest cardinality 2-interval pattern problem. *Journal of Combinatorial Optimization* 13, 263–275 (2007)
5. Choi, V., Zheng, C., Zhu, Q., Sankoff, D.: Algorithms for the extraction of syntenic blocks from comparative maps. In: Giancarlo, R., Hannenhalli, S. (eds.) *WABI 2007*. LNCS (LNBI), vol. 4645, pp. 277–288. Springer, Heidelberg (2007)
6. Crochemore, M., Hermelin, D., Landau, G.M., Rawitz, D., Vialette, S.: Approximating the 2-interval pattern problem. *Theoretical Computer Science (to appear)*; preliminary version appeared in Brodal, G.S., Leonardi, S. (eds.) *ESA 2005*. LNCS, vol. 3669, pp. 426–437. Springer, Heidelberg (2005)
7. Downey, R., Fellows, M.: *Parameterized Complexity*. Springer, Heidelberg (1999)
8. Garey, M.R., Johnson, D.S.: *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York (1979)

9. Jiang, M.: A 2-approximation for the preceding-and-crossing structured 2-interval pattern problem. *Journal of Combinatorial Optimization* 13, 217–221 (2007)
10. Jiang, M.: Improved approximation algorithms for predicting RNA secondary structures with arbitrary pseudoknots. In: Kao, M.-Y., Li, X.-Y. (eds.) *AAIM 2007*. LNCS, vol. 4508, pp. 399–410. Springer, Heidelberg (2007)
11. Jiang, M.: A PTAS for the weighted 2-interval pattern problem over the preceding-and-crossing model. In: Dress, A.W.M., Xu, Y., Zhu, B. (eds.) *COCOA 2007*. LNCS, vol. 4616, pp. 378–387. Springer, Heidelberg (2007)
12. Schaefer, T.: The complexity of satisfiability problem. In: *Proceedings of the 10th ACM Symposium on Theory of Computing (STOC 1978)*, pp. 216–226 (1978)
13. Vialette, S.: On the computational complexity of 2-interval pattern matching problems. *Theoretical Computer Science* 312, 223–249 (2004)
14. Zheng, C., Zhu, Q., Sankoff, D.: Removing noise and ambiguities from comparative maps in rearrangement analysis. *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 4, 515–522 (2007)

Greedy Local Search and Vertex Cover in Sparse Random Graphs

(Extended Abstract)

Carsten Witt*

DTU Informatics,
Technical University of Denmark,
2800 Kgs. Lyngby, Denmark

Abstract. Recently, various randomized search heuristics have been studied for the solution of the minimum vertex cover problem, in particular for sparse random instances according to the $G(n, c/n)$ model, where $c > 0$ is a constant. Methods from statistical physics suggest that the problem is easy if $c < e$. This work starts with a rigorous explanation for this claim based on the refined analysis of the Karp-Sipser algorithm by Aronson et al. Subsequently, theoretical supplements are given to experimental studies of search heuristics on random graphs. For $c < 1$, a greedy and randomized local-search heuristic finds an optimal cover in polynomial time with a probability arbitrarily close to 1. This behavior relies on the absence of a giant component. As an additional insight into the randomized search, it is shown that the heuristic fails badly also on graphs consisting of a single tree component of maximum degree 3.

1 Introduction

Randomized search heuristics (RSHs) such as Evolutionary Algorithms (EAs) [10, 19], Simulated Annealing [18], Ant Colony Optimization [14] etc. are general optimization techniques that prevail in applications where problem-specific algorithms are not available. In the last years, substantial progress has been made in the rigorous runtime analysis of RSHs for problems from combinatorial optimization [7, 8, 10, 14, 15, 18, 19]. In these works, the key question is how long the heuristics take in expectation to find a solution of a prespecified quality.

Recently, the behavior of RSHs for the minimum vertex cover (VC) problem has received increasing attention [7, 8, 15]. These studies were concerned with specific instances to the problem. A mostly empirical work [16] studies an average case where the graph is drawn randomly according to the $G(n, p)$ model [3] with $p = c/n$ for constant c . In particular, that paper highlights the well-known phase transition in this model. If $c < 1$ then all connected components of the graph are of size $O(\log n)$ with high probability $1 - o(1)$ (abbreviated as *w. h. p.* hereinafter),

* The author was supported by the Deutsche Forschungsgemeinschaft (DFG) as a part of the Collaborative Research Center “Computational Intelligence” (SFB 531). The work was mainly done while the author was at TU Dortmund University, Germany.

and the problem is easy to solve by complete enumeration. Actually, by references to methods from statistical physics [11], it is claimed in [16] that VC is easy in random graphs if $c < e$. However, no rigorous argument is given.

One aim of this work is to supply theoretical justifications to the experimental analyses of RSHs for VC in sparse random graphs. We start with a rigorous proof in Section 2 that VC in sparse random graphs can be solved in polynomial time w. h. p. for $c < e$. Despite being a simple consequence of the refined analysis of the Karp-Sipser algorithm presented by Aronson, Frieze and Pittel [1], this result does not seem to have been stated explicitly so far yet (at least in the community of theoretical computer science). Afterwards, we study the behavior of a simple RSH in our random graph model. A hybrid algorithm called Greedy-LS that combines ingredients of evolutionary algorithms and randomized local search with a greedy component is defined in Section 3. It is proved to find optimal VCs in the domain $c < 1$ with probability at least $1 - \epsilon$, $\epsilon > 0$ an arbitrary small constant. This result relies on the absence of the giant component but still comes unexpected since the local search is far from complete enumeration. In order to fathom the limits of the approach, it is shown that Greedy-LS fails even on trees of small degree when the tree forms a giant component. We finish with some conclusions. Most proofs in this extended abstract are only sketched.

2 A Modified Karp-Sipser Algorithm for Vertex Cover

In [16], various heuristics including EAs, SA, and branch and bound are studied for the VC problem in sparse random graphs drawn according to the $G(n, c/n)$ model. While evaluating their experiments, the authors of [16] claim that the problem is “typically” polynomial-time solvable for $c < e$. This is explained by a reference to methods from statistical physics [2, 11]. The underlying idea described by Bauer and Golinelli [2] is to study the application of a procedure called *leaf-removal* to the input graph: as long as the graph has at least one leaf (a vertex of degree 1), choose such a leaf uniformly and delete the leaf and its neighbor (along with incident edges) from the graph. The graph that finally remains is called the “core” in [2] (a notion different from the well-known (k -)core of a graph). The non-rigorous analysis using statistical mechanics reveals that the “core” is of size $O(\log n)$ provided that $c < e$. Hence, it is proposed in [2] to solve the VC problem by applying leaf-removal, putting cover marks on the leaves’ neighbors, and finally solving the VC problem on the “core” using branch and bound as a brute-force approach.

Roughly speaking, Bauer and Golinelli [2] identify a second phase transition in the $G(n, c/n)$ model besides the well-known emergence of a giant component at $c = 1$: namely, the emergence of a giant “core” at $c = e$. They also relate the latter to the so-called e -phenomenon first observed by Karp and Sipser [12] w. r. t. the maximum matching problem and studied in more detail by Aronson et al. [1]: i. e., the leaf-removal approach can also be applied to find a large matching in the graph. As long as $c < e$, the “core” remaining after leaf-removal is of asymptotically negligible size w. h. p., hence the set of edges chosen by leaf removal yields a $(1 - o(1))$ -approximation of a maximum matching.

The reference to the Karp-Sipser algorithm is more or less a side remark in [2]. From the viewpoint of theoretical computer science, we are aiming at making the interplay of the Karp-Sipser algorithm, the “core”, maximum matchings and minimum VC more explicit. In particular, we are interested in a rigorous statement regarding the “typical” $O(\log n)$ size claimed in [2] for the “core” graph. Fortunately, such a statement is already contained in the analysis in [1]. This results in the forthcoming Theorem 2 that VC in sparse random graphs can be solved to optimality w. h. p. if $c < e$. The algorithm used is a modification of the Karp-Sipser algorithm for the VC problem, called KS-VC and described in Algorithm 1. The notion $G \setminus N$ for a set of vertices N denotes the subgraph of G induced by $V(G) \setminus N$. By $V(G)$ and $E(G)$ we denote the set of vertices and edges of G , respectively.

Algorithm 1 (KS-VC)

1. $C := \emptyset$.
2. While $E(G) \neq \emptyset$
 - If G has at least one leaf then
 - choose a leaf $w \in V(G)$ uniformly,
 - let $\{v, w\} \in E(G)$ be the unique edge incident on w ,
 - $C := C \cup \{v\}$, $G := G \setminus \{v, w\}$ (**double-vertex removal**)
 - else
 - choose $v \in V(G)$ uniformly,
 - $C := C \cup \{v\}$, $G := G \setminus \{v\}$ (**single-vertex removal**).
3. Output C as Vertex Cover.

We describe the underlying ideas of the algorithm. If the graph has at least one leaf, the vertex adjacent to the leaf gets a cover mark, and these two vertices, along with their incident edges, are removed. The idea not to choose leaves for the cover is sometimes called *domination* and is present in many different approximation algorithms and search heuristics for VC (or, from a different viewpoint, independent set), see, e. g., [17] and [5]. Otherwise, a vertex is picked uniformly for the cover and only this single vertex and its incident edges are removed. In the first case, the graph is reduced in same manner as with the original Karp-Sipser algorithm for maximum matching. At the first instance where there are no leaves left, our approach starts to behave differently. By definition, KS-VC outputs a valid vertex cover.

Let Phase 1 end at the first point of time when G has no more leaves (note that new leaves can still be created afterwards), in other words G corresponds to the “core” as defined in [2]. We summarize a main result from [1]:

Theorem 1 ([1]). *Let $c < e$. Then at the end of Phase 1 of KS-VC, G is w. h. p. a collection of vertex disjoint cycles.*

We are ready to present the rigorous supplement to the study by Bauer and Golinelli [2]. Despite being a simple consequence from the previous theorem, the following result does not seem to have been stated explicitly so far (at least in the TCS community). A connection between Karp-Sipser and minimum VC

is drawn in [9], however, only the possibility of a $(1 + o(1))$ -approximation is noticed in the interesting domain $c < e$.

Theorem 2. *Let $c < e$. Then KS-VC finds an optimal vertex cover w. h. p.*

Proof. We analyze the first phase and the rest of the run separately. Let G^* be the graph remaining after the end of the first phase. Each optimal VC for the graph $G \setminus G^*$ can be converted into the result produced by KS-VC on this subgraph by iteratively moving the cover marks from leaves to their neighbors. All edges incident on vertices from $G \setminus G^*$ are covered in the first phase. Hence, it remains to prove that KS-VC produces an optimal cover on G^* .

By Theorem 1, G^* is a collection of vertex-disjoint cycles w. h. p. Let us assume this to happen. KS-VC covers a cycle of length k by a single execution of a single-vertex removal, followed by $\lceil (k - 2)/2 \rceil$ executions of a double-vertex removal, altogether using $\lceil k/2 \rceil$ cover marks. Since the cycles are disjoint, an optimal cover for G^* is produced, in total yielding an optimal cover for G . \square

3 Greedy Local Search

Many analyses of RSHs, most notably EAs, on problems from combinatorial optimization reveal that the heuristics mimick components of problem-specific algorithms with a certain probability [10, 13, 19]. Often this results in expected polynomial runtimes to find optimal or at least good approximate solutions to the problem. Compared to problem-specific algorithms, a loss of polynomial factors in the runtime seems to be a fair price to pay for the wide applicability of the heuristic. Of course, if the problem at hand is well understood and tailored algorithms are available, one would probably prefer the tailored algorithm. Still, it is interesting how RSHs compete with problem-specific algorithms on well-studied problems from combinatorial optimization since such analyses improve our understanding of the working principles of heuristics on realistic problems. Therefore, a main contribution of this paper are the methods for the analysis of the heuristics, not the heuristics themselves or the graph instances studied.

Most of the above-mentioned runtime analyses consider the worst case from a class of problems rather than average-case models. As seen before, the KS-VC algorithm is itself a heuristic, which performs extraordinarily well in the average-case model of sparse random graphs if $c < e$. We now turn our view to more classical search heuristics such as EAs, which were already analyzed on certain VC instances [7, 8, 15]. The well-known (1+1) EA (e. g., [10]) maintains search points from $\{0, 1\}^{|V|}$, i. e., each bit decides whether a vertex is included in the cover or not. The “fitness” of a search point is just the size of the current cover, or a penalty value greater than $|V|$ if no valid VC is encoded. The (1+1) EA creates a new tentative solution by flipping each bit of the current cover independently with probability $1/|V|$. If the new solution has at most the same fitness value as the current one, the new solution is accepted as the current solution, otherwise it is rejected. This procedure is repeated until some stopping criterion is satisfied.

We see that the (1+1) EA has the ability to change many bits in a step but is more likely to perform local steps changing only few bits. If at most two bits are allowed to flip, we arrive at a randomized local search (RLS) algorithm as investigated in [10]. Often RLS is much easier to analyze but is still competitive with the (1+1) EA since the search is driven by the local changes. Let us consider the local steps of the (1+1) EA flipping only a single or two bits, i. e., vertices. Since the (1+1) EA only goes from valid to valid covers and never increases cover size, the steps of size 1 are only accepted if they remove a cover mark from a vertex. Steps of size 2 may remove two cover marks or swap a cover mark on a vertex with a previously unmarked vertex. If the two vertices involved in such a swap are not connected by an edge, we already obtain a cover of smaller size by only removing the cover mark from the first vertex. Only if the swap goes along an edge, the removal of a single cover mark might be precluded. Hence, the search by means of 1-bit and 2-bit flips seems to be driven by steps removing single cover marks or swapping cover marks along edges. Most of the other steps will be wasted since they would lead to invalid covers.

Inspired by the preceding considerations, we define a search heuristic which can be seen as a hybridization of RLS and a greedy algorithm. Often RSHs are enhanced by problem-specific components, and the runtime analysis of such hybrid approaches is another branch in the theory of RSHs which has gained increasing interest [8]. The hybrid algorithm studied here is called *Greedy Local Search (Greedy-LS)* and defined as Algorithm 2. It starts from the full cover. If there is a vertex with all neighbors covered, this vertex is removed from the cover immediately, which is the greedy aspect. Otherwise, an edge is chosen uniformly. If swapping its endpoints in and out the cover leads to a still valid cover, the swap is accepted. The swaps can be considered as the local changes of the cover.

Algorithm 2 (Greedy-LS)

1. $C := V(G)$.
2. Repeat forever
 - If there is a vertex with all neighbors in C then
 - choose such a vertex, say v , uniformly, and set $C := C \setminus \{v\}$
(**vertex removal operation**)
 - else
 - choose $\{v, w\} \in E(G)$ uniformly; assume w. l. o. g. that $v \in C$,
if $w \notin C$ and $(C \setminus \{v\}) \cup \{w\}$ is a cover then set $C := (C \setminus \{v\}) \cup \{w\}$
(**edge swap operation**).

In our description, Greedy-LS does not terminate. We are interested in the random first point of time that the algorithm finds an optimum. Like for many local search algorithms, this time is not necessarily finite. Consider a bipartite graph with unequally sized subsets in the bipartition. If Greedy-LS happens to remove all vertices of the smaller subset from the cover, there will be no possible swap operations. Still, VC on bipartite graphs is polynomial-time solvable [4].

For the following analyses, it is crucial that Greedy-LS never includes a vertex and all its neighbors in the cover when an edge swap is executed. If the conditions for the cover marks to be swapped are satisfied, we call an edge *selectable*.

3.1 Sparse Random Graphs and the Case $c < 1$

In this subsection, we investigate the performance of the Greedy-LS heuristic in the $G(n, c/n)$ model. As mentioned above, the idea is to relate the random choices of the heuristic to a problem-specific algorithm, in this case KS-VC. It will turn out that Greedy-LS is able to reproduce an important subset of the decisions made by KS-VC with polynomial probability, implying that Greedy-LS finds minimum VCs in the domain $c < 1$ with good probability in polynomial time. This result is not obvious since our heuristic only allows local steps and is unable to explore connected components by complete enumeration.

We are ready to state the announced positive result regarding Greedy-LS.

Theorem 3. *Let $c < 1$. For every constant $\epsilon > 0$, Greedy-LS finds an optimal vertex cover in polynomial time with probability at least $1 - \epsilon - o(1)$.*

Sketch of proof: We use standard results on random graphs [3, p. 105] and a small technical lemma. As a result, we assume G to have unicyclic connected components (CCs) of maximal size $O(\log n)$ with $O(\log n)$ edges in each CC and length $O(1)$ for the longest cycles. This assumption is valid with probability at least $1 - \epsilon/2 - o(1)$. Actually, all central proof ideas go back to the case of cycle-free graphs (i. e., trees) and can be extended to the case of small cycles of length $O(1)$. In the following, we only consider trees.

We consider the CCs of G separately and study the probability that Greedy-LS to a sufficient extent “simulates” the behavior of KS-VC on this CC. Let a component C^* be fixed. Since it is assumed as a tree, KS-VC only executes double-vertex removals, i. e., it chooses edges incident on leaves. We have the freedom to determine the random order according to which KS-VC selects these edges. Therefore, let a root vertex $r^* \in V(C^*)$ for the connected component be fixed arbitrarily and let us assume that KS-VC always chooses a leaf of maximal distance to r^* . Denote by $e_1 = \{v_1, w_1\}, \dots, e_k = \{v_k, w_k\}$ the edges from C^* chosen in this order by KS-VC, i. e., v_1, \dots, v_k are covered. Again due to the tree structure, v_1, \dots, v_k is in fact a valid and optimal vertex cover w. r. t. C^* . Let $N(v_i)$ denote the set of neighbors of v_i and note that $w_i \in N(v_i)$. Since v_1, \dots, v_k is a cover, we have $V(C^*) = \bigcup_{i=1}^k (\{v_i\} \cup N(v_i))$.

While analyzing Greedy-LS, we concentrate on the edges e_1, \dots, e_k and their neighbors. Let $e_i, 1 \leq i \leq k$, be called *consistent* w. r. t. a current solution of Greedy-LS if only its v -vertex is chosen, *inconsistent* if only its w -vertex is chosen and *undecided* otherwise. The case of both vertices unchosen cannot happen as Greedy-LS always maintains valid covers. Moreover, it is crucial that Greedy-LS never increases the number of cover marks in a CC. The idea is to show that with a polynomial probability of $n^{-O(1)}$, Greedy-LS is able to make the k edges consistent and, in a manner sketched below, also their neighbors, or to arrive at an equally good cover iteratively by a sequence of swap and removal operations. Using standard waiting time arguments and Markov’s inequality, the polynomial probability is finally translated into a polynomial runtime that holds, altogether, with probability at least $1 - \epsilon - o(1)$.

Keeping in mind the tree structure, the core of the proof distinguishes two cases. Each edge $e_i = \{v_i, w_i\}$, $1 \leq i \leq k$, is either adjacent to a leaf in the original graph G , or adjacent to a leaf only after at least one double-vertex removal of KS-VC. In the first case, the edge can always be made consistent by either removing the w -vertex from the cover (if the edge was undecided) or a swap operation (if the edge was inconsistent). Let p_i be the parent of v_i and note that all neighbors $y \in N(v_i)$ except p_i must be leaves since w_i is a leaf of maximal depth. After making e_i consistent, all covered vertices in $N(v_i) \setminus \{p_i\}$ (and possibly further ones) will undergo removal operations by Greedy-LS. Restricted to the induced subgraph on $N^*(i) := \{v_i\} \cup N(v_i) \setminus \{p_i\}$, we have already obtained the same cover as KS-VC, which is, in fact, an optimal cover for this subgraph; we say that $N^*(i)$ has been made consistent. Moreover, unless v_i is involved in a swap operation, the number of cover marks in the subtree rooted at v_i never increases. This is a simple consequence of the tree structure.

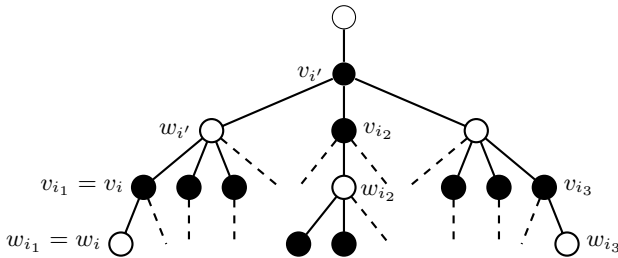


Fig. 1. Example: edge $e_{i'} = \{v_{i'}, w_{i'}\}$, the subtree of $v_{i'}$ and the edges $e_{i_j} = \{v_{i_j}, w_{i_j}\}$ with all these edges in correct state. The v_{i_j} have distance 1 or 2 from $v_{i'}$.

Now we sketch the case that e_i becomes incident on a leaf only after some steps of KS-VC. Consider the deepest ancestor of v_i lying on one of the edges e_1, \dots, e_k . Let i' denote the index of this edge and observe that $i' > i$. If $v_{i'}$ has s children, we investigate for each child the subtree rooted at the child and, in this subtree, all edges from e_1, \dots, e_k having a v -vertex of smallest depth. In particular, e_i is such an edge. Let $e_{i_1} = e_i, e_{i_2}, \dots, e_{i_s}$ be all these edges (see Figure 1 for an example). Due to the choice of these edges, each v_{i_j} , $1 \leq j \leq s$, has distance either 1 or 2 from $v_{i'}$. The first case is a child of $v_{i'}$ being also in the cover produced by KS-VC, the second case corresponds to an uncovered child. KS-VC processes the e_{i_j} -edges before $e_{i'}$. Hence, let us suppose that $N^*(i_1), \dots, N^*(i_s)$ are consistent in the current cover of Greedy-LS; different neighborhoods $N^*(\ell)$, $\ell < i'$ and $\ell \notin \{i_1, \dots, i_s\}$, need not be consistent at this moment (any more). In order to make $N^*(i')$ consistent, it can be necessary for Greedy-LS to (1) apply a swap operation to $e_{i'}$ and (2) apply removal operations to children of $v_{i'}$. Both operations remove cover marks from some children of $v_{i'}$ and are only guaranteed to lead to a valid cover if all edges incident on these children are covered. However, this holds by construction and assumption since we consider all edges from e_1, \dots, e_k with the v -vertex at distance 1 or 2 from $v_{i'}$ and assume

consistency for the $N(i_j)$. Hence, if Greedy-LS makes $N^*(i')$ consistent before any of the edges incident on $v_{i_1} \cup \dots \cup v_{i_s}$ are touched by Greedy-LS again, we obtain a cover of optimal size in terms of the subgraph rooted at $v_{i'}$. A not too involved combinatorial argument relying on the limited neighborhood and the assumed graph structure identifies a sufficient event of probability $n^{-O(1)}$. \square

3.2 Trees with Large Connected Components

The previous results relied on the fact that connected components are of logarithmic size w. h. p. In this section, we study the behavior of Greedy-LS on a sparse graph of maximum degree 3 having a single connected component and show a superpolynomial lower bound on its runtime. Previous results showing that randomized search heuristics fail to efficiently find minimum VCs were available only for dense graphs or graphs with large maximum degree [7, 8, 15].

Our example called FOOLINGTREE is defined on $n = 2^{k+2} - 3$ vertices. The graph is the subdivision obtained from a complete, rooted binary tree by replacing each edge with a path of length two (a figure had to be omitted due to space limitations). Hence, there are $2^{\lceil i/2 \rceil}$ vertices at depth $0 \leq i \leq 2k$. Any VC must include vertices from at least every second level. Moreover, the unique optimal VC for FOOLINGTREE chooses all vertices of odd depth.

There are many VCs being by one vertex away from optimality, e. g., the VC choosing all vertices of even depth. In our analyses, we concentrate mainly on the so-called *deep-end* edges between levels $2i - 1$ and $2i$, $1 \leq i \leq k$, i. e., the deeper edges from the paths of length two; all other edges are called *high end*. Each deep-end edge has a unique upper neighbor in a high-end edge, which neighbor is itself either the root or endpoint of another deep-end edge. Since all vertices except the root are endpoint of exactly one deep-end edge, the configuration of the deep-end edges together with the root is sufficient to specify any vertex cover. We call a configuration of a deep-end edge *correct* if only its odd-level vertex is chosen, *wrong* if only its even-level vertex is chosen and *complete* otherwise (the empty case cannot occur in valid vertex covers). The optimal vertex cover sets all deep-end edges correctly. In the following, we conceptually restrict the tree to the deep-end edges. From this perspective, we denote the set of deep-end edges below a deep-end edge e along with e itself as the *subtree rooted at e* and denote the upmost deep-end edges in the subtree as the two children of e .

We justify why FOOLINGTREE is fooling Greedy-LS. Let us consider a configuration of Greedy-LS where at least one deep-end edge is wrong. To correct the edge by an edge-swap operation of Greedy-LS, the configuration of the two children of the deep-end edge is crucial. Only if the two children are not wrong, the edge is selectable and the swap results in a valid VC; otherwise an edge in between the deep-end edge and its children would be uncovered. Let us assume for a moment that the two children are correct. Then there are two choices among the deep-end edge and its children that decrease the number of correct edges but only a single choice increasing this number. Hence, given that a wrong deep-end edge can be corrected, there seems to be a tendency towards more wrong edges.

We make these ideas a bit more precise. We give each deep-end edge a depth, defined by the number of deep-end edges from the root to the edge itself. Fix a valid VC and a deep-end edge e . In the subtree $T(e)$ of deep-end edges rooted at e , we define the following potential function denoted by $P(e)$: consider the set of edges in $T(e)$ that are (1) wrong, (2) selectable, and (3) have only wrong ancestors in $T(e)$. If this set is empty then $P(e) := 0$. Otherwise, $P(e)$ is the maximum depth (w. r. t. e) of these edges, increased by 1. The first aim is to show that $P(e)$ has a strong tendency to increase in the run of Greedy-LS on the FOOLINGTREE instance. Later, this result will be “amplified” in order to show that Greedy-LS needs expected superpolynomial time to optimize FOOLINGTREE. In the following, we call a step of Greedy-LS *relevant* for a subtree if it chooses a selectable edge from this tree for an edge-swap operation.

Lemma 1. *Let e be a deep-end edge and consider its current $P(e)$ -value. If $P(e) > 0$ then, with probability at least $1/2$, the $P(e)$ -value increases to its maximum $d(e)$ before it reaches 0; the conditional expected number of relevant steps for this is at most $12d(e)$. From a current value $P(e) = d(e)$, the probability to reach $P(e) = 0$ before falling back to $P(e) = d(e)$ is at most $2^{-\Omega(d(e))}$.*

In the forthcoming theorem, we let the potential increase simultaneously for the complete tree and several of its subtrees. Since Lemma 1 assumes a positive initial potential, we set up some sufficient conditions to increase a zero potential.

Lemma 2. *Let e be a deep-end edge with $P(e) = 0$. If its upper neighbor is covered, then there is a single edge-swap operation leading to $P(e) > 0$.*

A consequence of the preceding two lemmas, we obtain a third lemma not displayed here. Informally, as long as a deep-end edge can be turned wrong, the probability of observing it in correct state decreases exponentially with the depth of the subtree below the edge. The depth is $\Omega(\log n)$, which implies polynomial time for the correction with probability $\Omega(1)$. With high probability, there are linearly many such situations concurrently, resulting in the following theorem.

Theorem 4. *The expected optimization time of Greedy-LS on FOOLINGTREE is superpolynomial.*

4 Conclusions

We have studied the behavior of randomized search heuristics for minimum vertex cover in sparse random graphs according to the $G(n, c/n)$ model and supplemented previous experimental analyses. At first, we have rigorously proven that the problem can be solved to optimality for $c < e$ using a modification of the Karp-Sipser algorithm called KS-VC. Afterwards, a hybrid heuristic called Greedy-LS was investigated. For $c < 1$ it reproduces the decisions of KS-VC with a good probability in polynomial time. However, it fails badly already on graphs consisting of a single tree component of maximum degree 3. Our analyses provide insight into the behavior of greedy randomized search and present methods for its analysis. At the same time, they illustrate principles of hybridizations of problem-specific greedy algorithms and randomized search heuristics.

References

1. Aronson, J., Frieze, A., Pittel, B.G.: Maximum matchings in sparse random graphs: Karp-Sipser revisited. *Random Structures and Algorithms* 12(2), 111–177 (1998)
2. Bauer, M., Golinelli, O.: Core percolation in random graphs: a critical phenomena analysis. *The European Physical Journal B* 24(3), 339–352 (2001)
3. Bollobás, B.: *Random Graphs*, 2nd edn. Cambridge University Press, Cambridge (2001)
4. Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: *Introduction to Algorithms*, 2nd edn. MIT Press, Cambridge (2001)
5. Evans, I.K.: Evolutionary algorithms for vertex cover. In: Porto, V.W., Waagen, D. (eds.) EP 1998. LNCS, vol. 1447, pp. 377–386. Springer, Heidelberg (1998)
6. Feller, W.: *An Introduction to Probability Theory and Its Applications*, 3rd edn., vol. 1. Wiley, Chichester (1968)
7. Friedrich, T., He, J., Hebbinghaus, N., Neumann, F., Witt, C.: Approximating covering problems by randomized search heuristics using multi-objective models. In: Proc. of GECCO 2007, pp. 797–804. AMC Press (2007)
8. Friedrich, T., He, J., Hebbinghaus, N., Neumann, F., Witt, C.: Analyses of simple hybrid evolutionary algorithms for the vertex cover problem. *Evolutionary Computation* 17(1), 3–20 (2009)
9. Gamarnik, D., Nowicki, T., Swirszcz, G.: Maximum weight independent sets and matchings in sparse random graphs. Exact results using the local weak convergence method. *Random Structures and Algorithms* 28(1), 76–106 (2005)
10. Giel, O., Wegener, I.: Evolutionary algorithms and the maximum matching problem. In: Alt, H., Habib, M. (eds.) STACS 2003. LNCS, vol. 2607, pp. 415–426. Springer, Heidelberg (2003)
11. Hartmann, A., Weigt, M.: Statistical mechanics perspective on the phase transition in vertex covering of finite-connectivity random graphs. *Theoretical Computer Science* (265), 199–225 (2001)
12. Karp, R.M., Sipser, M.: Maximum matchings in sparse random graphs. In: Proc. of FOCS 1981, pp. 364–375. IEEE Press, Los Alamitos (1981)
13. Neumann, F., Wegener, I.: Randomized local search, evolutionary algorithms, and the minimum spanning tree problem. *Theoretical Computer Science* 378(1), 32–40 (2007)
14. Neumann, F., Witt, C.: Runtime analysis of a simple ant colony optimization algorithm. In: Asano, T. (ed.) ISAAC 2006. LNCS, vol. 4288, pp. 618–627. Springer, Heidelberg (2006); Extended version to appear in *Algorithmica*
15. Oliveto, P.S., He, J., Yao, X.: Evolutionary algorithms and the vertex cover problem. In: Proc. of CEC 2007, pp. 1870–1877. IEEE Press, Los Alamitos (2007)
16. Pelikan, M., Kalapala, R., Hartmann, A.K.: Hybrid evolutionary algorithms on minimum vertex cover for random graphs. In: Proc. of GECCO 2007, pp. 547–554. ACM Press, New York (2007)
17. Tarjan, R.E., Trojanowski, A.E.: Finding a maximum independent set. *SIAM Journal on Computing* 6(3), 537–546 (1977)
18. Wegener, I.: Simulated annealing beats metropolis in combinatorial optimization. In: Caires, L., Italiano, G.F., Monteiro, L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 589–601. Springer, Heidelberg (2005)
19. Witt, C.: Worst-case and average-case approximations by simple randomized search heuristics. In: Diekert, V., Durand, B. (eds.) STACS 2005. LNCS, vol. 3404, pp. 44–56. Springer, Heidelberg (2005)

Embedding the Diamond Lattice in the c.e. tt -Degrees with Superhigh Atoms

Douglas Cenzer¹, Johanna N.Y. Franklin², Jiang Liu³, and Guohua Wu³

¹ Department of Mathematics
University of Florida
310 Little Hall, Gainesville, FL 32611-8105, USA

² Department of Mathematics
National University of Singapore
2, Science Drive 2, Singapore 117543, Singapore

³ Division of Mathematical Sciences
School of Physical and Mathematical Sciences
Nanyang Technological University
Singapore 637371, Singapore

Abstract. The notion of superhigh computably enumerable (c.e.) degrees was first introduced by Mohrherr in [7], where she proved the existence of incomplete superhigh c.e. degrees, and high, but not superhigh, c.e. degrees. Recent research shows that the notion of superhighness is closely related to algorithmic randomness and effective measure theory. Jockusch and Mohrherr proved in [4] that the diamond lattice can be embedded into the c.e. tt -degrees preserving 0 and 1 and that the two atoms can be low. In this paper, we prove that the two atoms in such embeddings can also be superhigh.

1 Introduction

Lachlan proved in 1966 in [5] the classical Non-Diamond Theorem: no diamond can be embedded in the c.e. Turing degrees preserving both 0 and 1. However, Cooper showed that such a diamond can be embedded into the Δ_2^0 degrees if we do not require that the atoms be c.e. [1]. Later, Epstein showed that both atoms can be made low and that both atoms can be made high [3], and Downey proved in [2] that both atoms can be d.c.e. degrees, giving an extremely sharp result in terms of the Ershov hierarchy.

Alternately, we can consider the possibility of constructing a diamond preserving 0 and 1 if we consider a stronger reducibility. Since the proof of Lachlan's Non-Diamond Theorem holds in the c.e. wtt -degrees as well, no such diamond exists in the c.e. wtt -degrees. However, Jockusch and Mohrherr showed in [4] that the diamond lattice can be embedded into the c.e. tt -degrees preserving 0 and 1 and, furthermore, that the two atoms can be low. In this paper, we present a proof that such a diamond can be embedded into the c.e. tt -degrees in such a way that both atoms are superhigh.

The notion of superhigh c.e. degrees was first introduced by Mohrherr in [7], where a computably enumerable set A is defined to be *superhigh* if $A' \equiv_{tt} \emptyset''$.

In the same paper, Mohrherr proved the existence of incomplete superhigh c.e. degrees and also the existence of high, but not superhigh, c.e. degrees. More recently, Ng has shown in [8] that there is a minimal pair of superhigh c.e. degrees. Recent research in computability theory shows that the notion of superhighness is closely related to algorithmic randomness and effective measure theory. For instance, Simpson showed that uniformly almost everywhere dominating degrees are all superhigh [9] (the uniformly almost everywhere dominating degrees are all high follows from the characterization of highness via domination due to Martin), and Kjos-Hanssen, Miller and Solomon showed that the uniformly almost everywhere dominating degrees are exactly the degrees containing a set A such that \emptyset' is K -trivial relative to A .

Our theorem is stated as follows.

Theorem 1. *There are superhigh computably enumerable sets A and B such that $\mathbf{0}$, $deg_{tt}(A)$, $deg_{tt}(B)$, and $\mathbf{0}'_{tt}$ form a diamond in the computably enumerable tt -degrees.*

Our construction differs from Jockusch and Mohrherr’s in several important ways. Jockusch and Mohrherr’s construction involves only a finite injury argument, while ours involves an infinite injury argument, which is necessary to make A and B superhigh. Due to this, our sets A and B will not have some of the nice properties that Jockusch and Mohrherr’s do. For instance, they were able to build their atoms A and B with $A \cup B = K$, guaranteeing that $K \equiv_{tt} A \cup B$ in a very obvious way. In our construction, the superhighness strategies will force us to enumerate elements into A and B from time to time to maintain our computations that witness $A' \geq_{tt} TOT$ and $B' \geq_{tt} TOT$, where $TOT = \{e : \varphi_e \text{ is total}\}$ is a Π_2 -complete set. To ensure that $K \leq_{tt} A \oplus B$, we dedicate the numbers of the form $\langle x, 0 \rangle$ to meeting this requirement. This allows us to replace Jockusch and Mohrherr’s conclusion that $x \in K$ if and only if $x \in A \cup B$ by the slightly more complicated conclusion that $x \in K$ if and only if $\langle x, 0 \rangle \in A \cup B$. Again, for the consistency between the superhighness strategies and the minimal pair strategies, we need to be extremely careful when we switch from one outcome to another one.

Our notations and terminologies are standard and generally follow Soare [10]. Let φ_e and Φ_e^A be the e -th partial computable function and the e -th A -partial computable function, respectively. In particular, if $\varphi_e(x) \downarrow$, then $[e](x)$ denotes the truth table with index $\varphi_e(x)$ in some effective enumeration of all truth tables, denoted as $\tau_{\varphi_e(x)}$, and $||[e](x)||$ denotes the length of this truth table. For any set A , $[e]^A(x)$ is 0 or 1 depending on whether or not A satisfies the truth table condition with index $\varphi_e(x)$ (denoted by $A \models [e](x)$ if $[e](x) = 1$, otherwise, $A \not\models [e](x)$). Given two sets A and B , we say that $A \leq_{tt} B$ iff there is an e with φ_e total such that for all x , $[e]^B(x) = A(x)$. When we choose a *fresh* number as a γ -use or a δ -use at stage s , this number is the least number bigger than the corresponding restraint that is not of the form $\langle x, 0 \rangle$.

2 Requirements and Basic Strategies

To prove Theorem [□](#), we will construct two c.e. sets A and B such that both of them are superhigh, K is truth-table reducible to $A \oplus B$, and the tt -degrees of A and B form a minimal pair in the tt -degrees. A and B will satisfy the following requirements:

- \mathcal{P} : $K \leq_{tt} A \oplus B$;
- \mathcal{S}^A : $TOT \leq_{tt} A'$;
- \mathcal{S}^B : $TOT \leq_{tt} B'$;
- $\mathcal{N}_{i,j}$: $[i]^A = [j]^B = f$ total $\Rightarrow f$ is computable;

Recall that $TOT = \{e : \varphi_e \text{ is total}\}$ is a Π_2^0 -complete set. Therefore, if \mathcal{S}^A and \mathcal{S}^B are satisfied, then A and B will both be superhigh.

2.1 The \mathcal{P} -Strategy

To satisfy the requirement \mathcal{P} , we simply code K into $A \oplus B$. We will fix a computable enumeration of K such that at each odd stage s , exactly one number, k_s , enters K . At each odd stage s , we will enumerate $\langle k_s, 0 \rangle$ into A , B , or both. We will decide which of these sets to enumerate $\langle k_s, 0 \rangle$ into based on the actions of the minimal pair strategies $\mathcal{N}_{i,j}$. If $k \notin K$, then numbers of the form $\langle k, 0 \rangle$ will never be enumerated into A and B . It is obvious that we will have the equality $K = \{k : \langle k, 0 \rangle \in A \cup B\}$, and hence $K \leq_{tt} A \oplus B$.

The \mathcal{P} -requirement is global, so we do not need to place it on the construction tree.

2.2 An \mathcal{S}_e^A -Strategy

To make A superhigh, instead of giving a truth-table reduction from TOT to A' explicitly, we will construct a binary functional $\Gamma^A(e, x)$ such that for all $e \in \omega$,

$$TOT(e) = \lim_{x \rightarrow \infty} \Gamma^A(e, x)$$

with $|\{x : \Gamma^A(e, x) \neq \Gamma^A(e, x + 1)\}|$ bounded by a computable function h , which will ensure that $TOT \leq_{tt} A'$. (In the case of B , we will construct a binary functional $\Delta^B(e, y)$ with use $\delta(e, y)$ satisfying a similar requirement.) The crucial point is to find this computable bounding function h .

As usual, \mathcal{S}^A is divided into infinitely many substrategies \mathcal{S}_e^A , $e \in \omega$, each of which is responsible for the definition of $\Gamma^A(e, x)$ for $x \in \omega$, and has two outcomes, ∞ (a Π_2^0 -outcome) and f (a Σ_2^0 -outcome), where ∞ denotes the guess that φ_e is total and f denotes the guess that φ_e is not total. The main idea is that all the \mathcal{S}_e^A strategies (they will be arranged on a single level on the construction tree) work for the definition of $\Gamma^A(e, x)$, $x \in \omega$, jointly, and the one on the true path defines $\Gamma^A(e, x)$ for almost all x such that $\lim_{x \rightarrow \infty} \Gamma^A(e, x)$ exists and equals to $TOT(e)$.

Let β be an \mathcal{S}_e^A -strategy on the priority tree. As usual, we have the following standard definition of length agreement function:

$$l(\beta, s) = \max\{ x < s : s \text{ is a } \beta\text{-stage and } \varphi_e(y)[s] \downarrow \text{ for all } y < x\};$$

$$m(\beta, s) = \max\{ l(\beta, t) : t < s \text{ is a } \beta\text{-stage}\}.$$

Say that s is a β -expansionary stage if $s = 0$ or $l(\beta, s) > m(\beta, s)$.

Let s be a β -stage. If s is a β -expansionary stage, then we believe that φ_e is total, and for those $\Gamma^A(e, x)$ either defined by lower priority strategies, or defined by β itself, but with value 0, we undefine them by enumerating the corresponding $\gamma(e, x)$ into A and then define $\Gamma^A(e, y)$ to be 1 for the least y such that $\Gamma^A(e, y)$ is undefined. If s is not a β -expansionary stage, then we believe that φ_e is not total, and again, we undefine those $\Gamma^A(e, x)$ defined by lower priority strategies by enumerating the corresponding $\gamma(e, x)$ into A and then define $\Gamma^A(e, y)$ to be 0 for the least y with $\Gamma^A(e, y)$ not defined. Thus, if there are infinitely many β -expansionary stages (so φ_e is total, $e \in \text{TOT}$, and ∞ is the true outcome of β), then $\Gamma^A(e, x)$ is defined as 1 for almost all $x \in \omega$. On the other hand, if there are only finitely many β -expansionary stages (so φ_e is not total, $e \notin \text{TOT}$, and f is the true outcome of β), then $\Gamma^A(e, x)$ is defined as 0 for almost all $x \in \omega$.

Thus, for a fixed \mathcal{S}_e^A -strategy β on the construction tree, β will attempt to redefine $\Gamma^A(e, x)$ for almost all x . The only γ -uses which it will not be allowed to enumerate into A are (a) some γ -uses are prevented from being enumerated into A by higher priority strategies (when a disagreement is produced), or (b) $\Gamma^A(e, x)$ is defined by another \mathcal{S}_e^A -strategy with higher priority. In particular, if β is the \mathcal{S}_e^A -strategy on the true path, then there are only finitely many strategies with higher priority that can be visited during the whole construction, and hence β can succeed in defining $\Gamma^A(e, x)$ for almost all x .

Suppose that \mathcal{S}_e^A is assigned to nodes on level n . We will see that $|\{x : \Gamma^A(e, x) \neq \Gamma^A(e, x + 1)\}| \leq 2^{3^n + 1}$. To see this, note that (a) above can happen at most 2^{3^n} times, as there are at most 3^n many strategies with length less than n , and each time when one of them produces (not preserves) a disagreement, a restraint is set, preventing α from rectifying $\Gamma^A(e, x)$ for some x . Note that after an \mathcal{N} -strategy α (see below) produces a disagreement, say at stage s , whenever α requires us to preserve this disagreement, all the strategies with lower priority will be initialized, and at the same time, all of the γ -uses and δ -uses defined after stage s will be enumerated into A and B respectively (one by one, as pointed out above, for the sake of the \mathcal{N} -strategies with priority higher than α). It is crucial for us to ensure that TOT is truth-table reducible to A' and B' , as we will discuss below.

Here, when β is initialized by a strategy with higher priority with length $\geq n$, an \mathcal{S}_e^A -strategy β' on the left of β is visited, and β' takes the responsibility of rectifying $\Gamma^A(e, x)$ for some x , which can lead to an equality between $\Gamma^A(e, x)$ and $\Gamma^A(e, x + 1)$. Thus, (b) can happen at most 3^n many times. In total, the number of those x such that β cannot rectify $\Gamma^A(e, x)$ is at most $2^{3^n + 1}$, which ensures that $\text{Tot} \leq_{tt} A'$, where the corresponding bounding function h is given by $h(e) = 2^{3^{3^e} + 1}$.

We remark here that as a bounding function, h is not tight, but it is enough to show that TOT is truth-table reducible to A' , as we want.

2.3 An $\mathcal{N}_{i,j}$ -Strategy

Recall that if $[i]$ is a *tt*-reduction, then for any oracle $X \subseteq \omega$ and any input x , $[i]^X(x)$ converges. The computation $[i]^X(x)$ can be injured at most finitely many times due to the enumeration of numbers less than or equal to $|\tau_{\varphi_i(x)}|$ into X in our construction.

For the requirement $\mathcal{N}_{i,j}$, we apply the diagonalization argument introduced by Jockusch and Mohrherr in [4]. That is, once we see a disagreement between $[i]^A$ and $[j]^B$, we will preserve it forever to make $[i]^A \neq [j]^B$. On the other hand, if $[i]^A$ and $[j]^B$ are equal and total, then we will ensure that they are computable.

Given values for A_s and B_s at stage s , we will define A_{s+1} and B_{s+1} at stage $s + 1$ by possibly enumerating into them. Furthermore, if we know that $[i]^A$ and $[j]^B$ differ at k at stage s , we will have to preserve this disagreement at stage $s + 1$. This is achieved by the following. Let n be a number we want to put into $A_{s+1} \cup B_{s+1}$. There are two cases.

(1) Our number n is of the form $\langle x, 0 \rangle$ for some x . Then n is enumerated into A , B , or both for the sake of the requirement \mathcal{P} . There are three subcases.

Subcase 1: If $[i]^{A_s}(k) = [i]^{A_s \cup \{n\}}(k)$, then n will be enumerated into A but not into B . The disagreement is preserved as well.

Subcase 2: If Subcase 1 does not apply but $[j]^{B_s}(k) = [j]^{B_s \cup \{n\}}(k)$, then n is enumerated into B but not into A . As in Case 1, the disagreement is preserved.

Subcase 3: If $[i]^{A_s}(k) \neq [i]^{A_s \cup \{n\}}(k)$ and $[j]^{B_s}(k) \neq [j]^{B_s \cup \{n\}}(k)$, then n is enumerated into both A and B . In this case, the disagreement is again preserved, as both values are changed.

Note that once one subcase above applies, then we initialize all the strategies with lower priority to avoid conflict among the \mathcal{N} -strategies — obviously, such initializations can happen at most finitely often. We need to be careful here when more \mathcal{N} -strategies are considered. It can happen that if we decide to enumerate into A , B , or both, we also need to take care of those \mathcal{N} -strategies with higher priority, say $\mathcal{N}_{i',j'}$, as we need to avoid the following situation: according to the $\mathcal{N}_{i,j}$ -strategy, at stage s_1 , a number n_1 is enumerated into A , and at stage s_2 , a number n_2 is enumerated into B (corresponding to Subcases 1 and 2, respectively), and such enumerations change $[i']^A(m)$ and $[j']^B(m)$, though separately, and at the next $\mathcal{N}_{i',j'}$ -expansionary stage, we may have $[i']^A(m) = [j']^B(m)$, which is different from its original value — $\mathcal{N}_{i',j'}$ is injured.

With this in mind, when we see that an $\mathcal{N}_{i,j}$ -strategy wants to enumerate a number into A (or B , or both), instead of enumerating it immediately, we first check whether such an enumeration into A can lead to a disagreement between $[i']^A$ and $[j']^B$. If not, then we just work as described above (in Subcase 3, we now enumerate n into B and check whether this enumeration into B can lead

to a disagreement for $\mathcal{N}_{i',j'}$ — here n is enumerated into A and B separately). Otherwise, we start to preserve this disagreement to satisfy $\mathcal{N}_{i',j'}$ — the $\mathcal{N}_{i,j}$ considered above is initialized, and again, even if Subcase 3 applies, we do not enumerate n into B .

The \mathcal{N} -strategies are arranged linearly according to priority, and each time \mathcal{P} decides to act it checks for the highest priority \mathcal{N} -strategy for which the enumeration of $\langle k, 0 \rangle$ into A or B will change an \mathcal{N} -computation. We then act for \mathcal{N} as in subcases 1-3 above. This clearly injures an \mathcal{N}' -strategy of lower priority and it will need to be initialized, but it is easy to see that each \mathcal{N}' is injured in this way by the global \mathcal{P} only finitely often.

(2) Our number n is a number chosen by an \mathcal{S}_e^A -strategy or an \mathcal{S}_e^B -strategy. Without loss of generality, suppose that n is selected by an \mathcal{S}_e^A -strategy and we want to put it into A . As in the standard construction of high sets, we only consider believable computations; for instance, $[i]^A(m)$. Therefore, when we see $[i]^A$ and $[j]^B$, if this \mathcal{S}_e^A -strategy has higher priority than $\mathcal{N}_{i,j}$, then the enumeration of n into A does not affect the computation $[i]^A(m)$. We will have more discussion on this soon.

An $\mathcal{N}_{i,j}$ -strategy has three outcomes: ∞ , f and d , where ∞ denotes that there are infinitely many expansionary stages, f denotes that there are only finitely many expansionary stages, but no disagreement is produced, and d denotes that a disagreement between $[i]^A$ and $[j]^B$ is produced and preserved successfully.

2.4 More on Interactions among Strategies

We have seen some interactions between the \mathcal{P} -strategy and the \mathcal{N} -strategies. Now we describe the interactions between the \mathcal{N} -strategies, the \mathcal{S} -strategies, and the \mathcal{P} -strategy.

Assume that α is an $\mathcal{N}_{i,j}$ -strategy, β is an \mathcal{S}_e^A -strategy, and ζ is an \mathcal{S}_e^B -strategy with $\beta \frown \infty \subseteq \zeta \frown \infty \subseteq \alpha$. The following may happen: at a stage s , a disagreement between $[i]^A$ and $[j]^B$ appears at α , so α wants to preserve this disagreement by initializing all strategies with lower priority. However, this disagreement can be destroyed by β and ζ , as they may enumerate small γ -uses and δ -uses into A and B separately. To avoid this, we only use α -believable computations, a standard technique in the construction of high degrees.

Definition 1. *Let α be an $\mathcal{N}_{i,j}$ -strategy, and β be an \mathcal{S}_e^A -strategy with $\beta \frown \infty \subseteq \alpha$.*

- (1) *A computation $[i]^{A_s}(m)$ is α -believable at β at stage s if for each x with $\gamma(e, x)[s]$ defined by β and less than the length of the truth-table of $[i](m)$, $\Gamma^{A_s}(e, x)[s]$ is equal to 1.*
- (2) *A computation $[i]^{A_s}(m)$ is α -believable at stage s if it is α -believable at β at stage s for any \mathcal{S}_e^A -strategy β , $e \in \omega$, with $\beta \frown \infty \subseteq \alpha$.*

Similarly, we can define an α -believable computation $[j]^{B_s}(m)$.

We are ready to define an α -expansionary stage for an $\mathcal{N}_{i,j}$ -strategy α .

Definition 2. Let α be an $\mathcal{N}_{i,j}$ -strategy. The length of agreement between $[i]^A$ and $[j]^B$ is defined as follows:

$$l(\alpha, s) = \max\{x < s : \text{for all } y < x, [i]^A(y)[s] = [j]^B(y)[s] \\ \text{via } \alpha\text{-believable computations}\}.$$

$$m(\alpha, s) = \max\{l(\alpha, t) : t < s \text{ is an } \alpha\text{-stage}\}.$$

Say that a stage s is α -expansionary if $s = 0$ or $l(\alpha, s) > m(\alpha, s)$.

At an α -expansionary stage, before α is allowed to access outcome ∞ , it has to clear every γ, δ -use in $F_\alpha^A \cup F_\alpha^B$, where F_α^A, F_α^B are the collections of γ, δ -uses defined by \mathcal{S} -strategies with priority lower than α after the last α -expansionary stage. We enumerate these uses one at a time into A or B respectively, until a disagreement is produced at some \mathcal{N}' -strategy $\beta \subset \alpha$. We then stop and do not access the nodes extending $\alpha \frown \infty$ at this current stage. This is alright because a strong priority β has made permanent (subject to β 's ability to protect this disagreement) progress on its basic strategy. We will refer to this enumeration process as an “*outcome-shifting enumeration process*” for simplicity. *So a tt-minimal pair strategy does not enumerate numbers into sets, which is completely different from the minimal pair argument used in the c.e. Turing degrees.*

Now we consider the situation when β , an \mathcal{S}_e^A -strategy, changes its outcome from f to ∞ at a β -expansionary stage. Again, when β sees such a change of outcome, it also perform the outcome-shifting enumeration process by enumerating numbers into A and B as needed. That is, let s' be the last β -expansionary stage. Unlike the construction of high degrees, to make A and B superhigh, we need to enumerate all the γ -uses and δ -uses defined by strategies below outcome f , including those uses defined by β under the outcome f , between stages s' and s into A and B respectively. Again, these numbers cannot be enumerated into A and B simultaneously, as discussed above in the section on the \mathcal{N} -strategies, for the sake of \mathcal{N} -strategies with priority higher than β . Let F_β^A and F_β^B be the collections of these γ -uses and δ -uses respectively. We put the numbers in $F_\beta^A \cup F_\beta^B$ into A or B correspondingly, one by one, from the smallest to the largest, and whenever one number is enumerated, we reconsider the \mathcal{N} -strategies with higher priority to see whether a disagreement appears. Once such a disagreement appears at an \mathcal{N} -strategy, say α , we stop the enumeration as we need to satisfy α via this disagreement. In this case, β is injured. Note that β can be injured in this way only by those \mathcal{N} -strategies α such that $\alpha \subset \beta$.

2.5 Construction

First, we define the priority tree T and assign requirements to the nodes on T as follows. Suppose $\sigma \in T$. If $|\sigma| = 3e$, then σ is assigned to the $\mathcal{N}_{i,j}$ -strategy such that $e = \langle i, j \rangle$. It has three possible outcomes: ∞, f , and d , with $\infty <_L f <_L d$. If $|\sigma| = 3e + 1$, then σ is assigned to the \mathcal{S}_e^A -strategy. If $|\sigma| = 3e + 2$, then σ is assigned to the \mathcal{S}_e^B -strategy. In the latter two cases, σ has two possible outcomes: ∞ and f , with $\infty <_L f$.

\mathcal{P} is a global requirement, and we do not put it on the tree.

We assume that K is enumerated at odd stages. That is, we fix an enumeration $\{k_{2s+1}\}_{s \in \omega}$ of K such that at each odd stage $2s + 1$, exactly one number, k_{2s+1} , is enumerated into K .

In the construction, we say that an $\mathcal{N}_{i,j}$ -strategy α sees a disagreement at k at a stage s if $k \leq s$, $[i]^{A_s}$ and $[j]^{B_s}$ agree on all arguments $\leq k$, and one of the following cases applies:

- (i) s is odd (k_s enters K and we need to put $\langle k_s, 0 \rangle$ into $A \cup B$). In this case, either
 - (1) $[i]^{A_s}(k) \neq [i]^{A_s \cup \{k_s, 0\}}(k)$,
 - (2) $[j]^{B_s}(k) \neq [j]^{B_s \cup \{k_s, 0\}}(k)$, or
 - (3) there is an \mathcal{N} -strategy $\alpha' \supset \alpha$ that attempts to preserve a disagreement, and the enumeration of $\langle k_s, 0 \rangle$ into A or B or both (depending on α') and an one-by-one enumeration of elements of $F_{\alpha'}^A \cup F_{\alpha'}^B$ into A and B (in increasing order, as described in the \mathcal{S} -strategies) leads to either $[i]^A(k) \neq [i]^{A_s}(k)$ or $[j]^B(k) \neq [j]^{B_s}(k)$. Here, $F_{\alpha'}^A$ and $F_{\alpha'}^B$ are the finite collections of γ -uses and δ -uses defined below outcome $\alpha' \frown d$ after the last stage α' that produces or preserves its disagreement.

If (1) is true, then we enumerate $\langle k_s, 0 \rangle$ into A . If (1) is not true but (2) is, then we enumerate $\langle k_s, 0 \rangle$ into B . Otherwise, (3) is true, and we enumerate $\langle k_s, 0 \rangle$ into A or B or both, according to α' . We also enumerate the corresponding numbers in $F_{\alpha'}^A \cup F_{\alpha'}^B$ into A and B respectively.

As a consequence, a disagreement between $[i]^A(k)$ and $[j]^B(k)$ is produced, and α will preserve this disagreement forever unless it is initialized later.

- (ii) s is even (s is a β -expansionary stage for some \mathcal{S} -strategy β). Let β be such a strategy, and let s' be the last β -expansionary stage. At stage s , to change its outcome from f to ∞ , we need to enumerate all of the elements in F_{β}^A and F_{β}^B into A and B respectively one by one. Here, F_{β}^A and F_{β}^B are the finite collections of γ -uses and δ -uses defined below outcome $\beta \frown f$, including those defined by β under the outcome f , after stage s' . Again, we enumerate these numbers into A and B in increasing order until we find that either $[i]^A(k) \neq [i]^{A_s}(k)$ or $[j]^B(k) \neq [j]^{B_s}(k)$ is true; that is, until a disagreement between $[i]^A(k)$ and $[j]^B(k)$ is produced. From now on, α will preserve this disagreement forever unless it is initialized later.

We recall that an $\mathcal{N}_{i,j}$ -strategy α preserves a disagreement at k at an odd stage s if this disagreement was produced before and has been preserved so far (so $[i]^{A_s}(k) \neq [j]^{B_s}(k)$) and $\langle k_s, 0 \rangle$ is less than one of the lengths of the truth-tables $[i](k)$ and $[j](k)$. Enumerating $\langle k_s, 0 \rangle$ into $A \cup B$ causes one of the following to happen:

1. If $[i]^{A_s}(k) = [i]^{A_s \cup \{k_s, 0\}}(k)$, then $\langle k_s, 0 \rangle$ is enumerated into A but not into B . Both values are preserved, and the disagreement is preserved as well.
2. If $[j]^{B_s}(k) = [j]^{B_s \cup \{k_s, 0\}}(k)$, then $\langle k_s, 0 \rangle$ is enumerated into B but not into A . As in Case 1, the disagreement is preserved.

3. If $[i]^{A_s}(k) \neq [i]^{A_s \cup \{(k_s, 0)\}}(k)$ and $[i]^{B_s}(k) \neq [i]^{B_s \cup \{(k_s, 0)\}}(k)$, then $\langle k_s, 0 \rangle$ is enumerated into both A and B . In this case, the disagreement is again preserved, as both values are changed.

Note that whenever α produces or preserves a disagreement in this manner, all the strategies below the outcome $\alpha \frown d$ are initialized. Such initializations can happen at most finitely often.

Construction

Stage 0: Initialize all the nodes on T and set $A_0 = B_0 = \emptyset$. Let $\Gamma^A(e, x)[0]$ and $\Delta^B(e, x)[0]$ be undefined for each e and x .

Stage $s > 0$:

Case 1: s is odd. We will put $\langle k_s, 0 \rangle$ into $A \cup B$ at this stage.

First check whether there is an \mathcal{N} -strategy that can produce a disagreement or needs to preserve a disagreement. Let α be the least such \mathcal{N} -strategy. Enumerate $\langle k_s, 0 \rangle$ into A or B or both accordingly. Initialize all the strategies with lower priority.

Case 2: s is even. We define the approximation to the true path σ_s of length $\leq s$. Suppose that $\sigma_s \upharpoonright u$ has been defined for $u \leq t$ and let ξ be $\sigma_s \upharpoonright t$. We will define $\sigma_s(t)$. We have the following two subcases.

Subcase 1: ξ is an $\mathcal{N}_{i,j}$ -strategy for some i and j . If ξ has produced a disagreement before and ξ has not been initialized since then, we let $\sigma_s(t) = d$. Otherwise, we check whether s is a ξ -expansionary stage. If not, then let $\sigma_s(t) = f$. If it is, then we start the outcome-shifting enumeration process to enumerate those γ -uses from F_ξ^A and δ -uses from F_ξ^B defined below the outcome $\xi \frown f$ from the last ξ -expansionary stage into A and B respectively, one by one and in increasing order. At the same time, each time we enumerate such a number, we check whether there is an \mathcal{N} -strategy $\alpha \subset \xi$ that can produce a disagreement. If there is, then we stop the enumeration of F_ξ^A and F_ξ^B into A and B and let $\delta_s = \alpha$. Declare that α produces a disagreement at stage s , let $\sigma_s = \alpha$, and go to the ‘defining’ phase. If not, then after all numbers in $F_A \cup F_B$ have been enumerated, we let $\sigma_s(t) = \infty$ and go to the next substage.

Subcase 2: ξ is an \mathcal{S}_e^A -strategy or an \mathcal{S}_e^B -strategy for some e . If s is not a ξ -expansionary stage, let $\sigma_s(t) = f$ and go to the next substage. Otherwise, we start the outcome-shifting enumeration process as described in Subcase 1. Here F_ξ^A and F_ξ^B should also contain those γ -uses or δ -uses defined by ξ under the outcome f .

Defining Phase of stage s : For those \mathcal{S}_e^A -strategies β with $\beta \frown \infty \subseteq \sigma_s$, find the least y such that $\Gamma^A(e, y)$ is currently not defined, define it as 1 and let the use $\gamma(e, y)$ be a fresh number, and for those \mathcal{S}_e^A -strategies β with $\beta \frown f \subseteq \sigma_s$, find

the least y such that $\Gamma^A(e, y)$ is currently not defined, define it as 0, and let the use $\gamma(e, y)$ be a fresh number. For those \mathcal{S}_e^B -strategies β , we define $\Delta^B(e, y)$ in the same way. Initialize all the strategies with lower priority than σ_s and go to the next stage.

Note that the enumeration of those γ -uses and δ -uses at substages into A and B ensures that those $\Gamma^A(e, x)$ and $\Delta^B(e, y)$ defined by those strategies with priority lower than σ_s are undefined.

This completes the construction.

Let $TP = \liminf_s \sigma_{2s}$ be the true path of the construction. We can first prove that TP is infinite and then verify that the construction given above satisfies all the requirements. Also it is obvious from the construction that

$$x \in K \iff \langle x, 0 \rangle \in A \cup B,$$

and hence $K \leq_{tt} A \oplus B$.

This completes the proof of Theorem \square .

Acknowledgement

Cenzer is partially supported by NSF grants DMS-0554841 and DMS-652372, U.S.A. Wu is partially supported by a research grant RG58/06 from Nanyang Technological University, Singapore.

References

1. Cooper, S.B.: Degrees of unsolvability complementary between recursively enumerable degrees. I. *Ann. Math. Logic* 4, 31–73 (1972)
2. Downey, R.: D.r.e. degrees and the nondiamond theorem. *Bull. London Math. Soc.* 21, 43–50 (1989)
3. Epstein, R.L.: Minimal degrees of unsolvability and the full approximation construction. *Mem. Amer. Math. Soc.* 162(3) (1975)
4. Jockusch Jr., C.G., Mohrherr, J.: Embedding the diamond lattice in the recursively enumerable truth-table degrees. *Proc. Amer. Math. Soc.* 94, 123–128 (1985)
5. Lachlan, A.H.: Lower bounds for pairs of recursively enumerable degrees. *Proc. London Math. Soc.* 16, 537–569 (1966)
6. Martin, D.: Classes of recursively enumerable sets and degrees of unsolvability. *Z. Math. Logik Grundlag. Math.* 12, 295–310 (1966)
7. Mohrherr, J.: A refinement of low_n and high_n for the r.e. degrees. *Z. Math. Logik Grundlag. Math.* 32, 5–12 (1986)
8. Ng, K.M.: On Very High Degrees. *Jour. Symb. Logic* 73, 309–342 (2008)
9. Simpson, S.G.: Almost everywhere domination and superhighness. *Math. Log. Q.* 53, 462–482 (2007)
10. Soare, R.I.: *Recursively enumerable sets and degrees.* Springer, Heidelberg (1987)

Feasibility of Motion Planning on Directed Graphs

Zhilin Wu¹ and Stéphane Grumbach²

¹ CASIA-LIAMA

zlwu@liama.ia.ac.cn

² INRIA-LIAMA

stephane.grumbach@inria.fr

Abstract. Because of irreversibility of movements, motion planning on directed graphs is much more intricate than that on graphs. Recently we showed that the feasibility of motion planning on acyclic and strongly connected directed graphs can be decided in time $O(nm)$ (n, m are respectively the number of vertices and arcs of the directed graph), but left the feasibility of motion planning on (general) directed graphs open. In this paper, we complete the solution by showing that the feasibility of motion planning on directed graphs can be decided in time $O(n^2m)$.

1 Introduction

Motion planning is a fundamental problem of robotics. It has been extensively studied [LaV06], and has numerous practical applications beyond robotics, such as in manufacturing, animation, games [MPG] as well as in computational biology [SA01, FK99]. The study of motion planning on graphs was proposed by Papadimitriou et al. [PRST94] to strip away the geometric considerations of the general motion planning problem and concentrate on the combinatorial aspects.

Motion planning on graphs is defined as follows. Suppose a graph $G = (V, E)$ is given. There is one robot at a source vertex s and some of the other vertices contain a movable obstacle. The objective is to move the robot from s to a destination vertex t with the smallest number of moves, where a move consists in moving an object (robot or obstacle) from one vertex to an adjacent vertex that contains a hole (if a vertex does not contain an object, then it is said to contain a hole; if an object is moved from v to w , we can also say that a hole is moved from w to v).

If there are too many obstacles, it may be impossible to move the robot from the source to the destination. So before considering the optimization problem, the decision problem whether a given instance of the problem of motion planning on graphs is feasible or not, should be considered first.

Motion planning on graphs is an abstraction of the practical problems, such as track transportation systems [Per88] and packet transfer in communication buffers of networks.

In practice, for the above two examples, tracks or links might be asymmetric. For instance, there may be unidirectional links in networks, especially in wireless

networks, due to the heterogeneity of receiver and transmitter hardware [MD02, JJ06]. This motivates the study of motion planning on directed versus undirected graphs.

Directed graphs (abbreviated as digraphs from now on) differ from undirected graphs mainly in that movements in digraphs are irreversible. In [WG08], we proposed two algorithms to decide the feasibility of motion planning on acyclic and strongly-connected digraphs in time $O(nm)$ (n, m are respectively the number of vertices and arcs).

For digraphs which are neither acyclic nor strongly connected, the motion planning problem may become much more tricky. For instance, the motion planning problem given in Fig. 1(a) is feasible. Let C_s denote the strongly connected component of s , then if initially the hole in v_7 is moved into C_s through v_2 (see Fig. 1(b)), the problem will become infeasible, which the reader can easily verify.

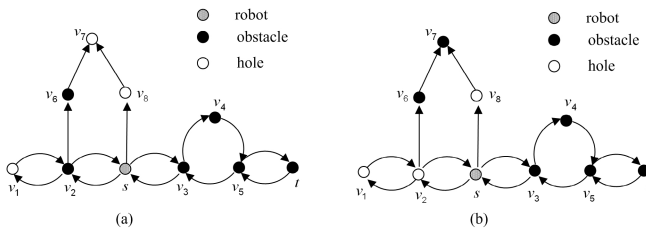


Fig. 1. Motion planning on digraphs

In this paper, we give a complete solution to the feasibility of motion planning on digraphs and show that it can be decided in time $O(n^2m)$. We distinguish between the cases whether C_s , the strongly connected component containing s , is trivial or not, and whether s and t belong to the same strongly connected component or not. If C_s is trivial, then the feasibility can be solved by combining the two algorithms for feasibility of motion planning on acyclic and strongly connected digraphs in [WG08]. Otherwise, a greedy strategy to move the outside holes into C_s can be designed to solve the feasibility problem.

Without loss of generality, we assume in this paper that in the motion planning problem, the source vertex s and the target vertex t of the robot are different, and there is at least one path from s to t .

The paper is organized as follows: In Section 2, some preliminaries are given. The structure of digraphs is discussed in Section 3. Then in Section 4, feasibility of motion planning on digraphs is solved case by case.

2 Preliminaries

The notations of this paper follow those in [Wes00, BJG00].

The *underlying graph* of a digraph $D = (V, E)$, denoted $\mathcal{G}(D)$, is the graph obtained from D by ignoring the directions of the arcs.

Let $G = (V, E)$ be a graph. The *biconnected-component graph* of G , denoted by $\mathcal{G}_{bc}(G)$, is a bipartite graph (V_{bc}, W_{bc}, E_{bc}) defined by

- V_{bc} contains the biconnected components of G ;
- W_{bc} contains all $v \in V$ such that v is shared by at least two distinct biconnected components of G ;
- E_{bc} is defined as follows: let $B \in V_{bc}$ and $w \in W_{bc}$, then $\{B, w\} \in E_{bc}$ iff $w \in V(B)$.

In [WG08], strongly biconnected digraphs were introduced to decide the feasibility of motion planning on strongly connected digraphs.

Definition 1. Let D be a digraph. D is said to be strongly biconnected if D is strongly connected and $\mathcal{G}(D)$ is biconnected. The strongly biconnected components of D are the maximal strongly biconnected sub-digraphs of D .

With regard to the feasibility of the motion planning problem, strongly biconnected digraphs have the following nice property.

Theorem 1 ([WG08]). The motion planning problem on a strongly biconnected digraph D is feasible iff there is at least one hole in D .

Strongly connected digraphs admit the following decomposition.

Theorem 2 ([WG08]). Let $D = (V, E)$ be a nontrivial strongly connected digraph. Then the strongly biconnected components of D are those $D[V(B)]$, the sub-digraph of D induced by $V(B)$, where B is a biconnected component of $\mathcal{G}(D)$.

Let $D = (V, E)$ be a strongly connected digraph, define the *strongly-biconnected-component graph* of D , denoted $\mathcal{G}_{sbc}(D) = (V_{sbc}, W_{sbc}, E_{sbc})$, as the biconnected-component graph of $\mathcal{G}(D)$. Let $v \in V$, v is called a *branching vertex* if $v \in W_{sbc}$ and the degree of v is greater than 2 in $\mathcal{G}_{sbc}(D)$. A *chain* of $\mathcal{G}_{sbc}(D)$ is a path $B_0 v_1 B_1 \cdots B_{k-1} v_k B_k$ ($k \geq 1$) in $\mathcal{G}_{sbc}(D)$ such that $|V(B_i)| = 2$ for all $1 \leq i \leq k - 1$, and v_i is not a branching vertex for all $1 < i < k$.

The *length* of a chain is the number of vertices in W_{sbc} on the chain.

Since the biconnected-component graph of a graph is a tree [Wes00], it follows that $\mathcal{G}_{sbc}(D)$ is a tree as well.

Theorem 3 ([WG08]). Feasibility of motion planning on acyclic and strongly connected digraphs can be decided in time $O(nm)$, where n, m are respectively the number of vertices and arcs of the digraph.

3 Structure of Digraphs

Let $D = (V, E)$ be a digraph. Then the vertex sets of strongly connected components of D form a partition of V .

Definition 2. Let $D = (V, E)$ be a digraph. The *strongly-connected-component digraph* of D , $\mathcal{D}_{scc}(D) = (V_{scc}, E_{scc})$, is defined as follows:

- $V_{scc} = V_{tr} \cup V_{nt} \cup V_{pt}$, where
 - V_{tr} contains all $v \in V$ such that v is a trivial strongly connected component of D ;
 - V_{nt} contains all nontrivial strongly connected components of D ;

- V_{pt} contains all $v \in V$ such that v belongs to some nontrivial strongly connected component of D (say C) and there is some $w \notin V(C)$ such that $(v, w) \in E$ or $(w, v) \in E$. Those v 's are called the ports of C .
- E_{scc} is defined by the following two rules:
 - If $C \in V_{nt}$, $v \in V_{pt}$, and $v \in V(C)$, then $(C, v) \in E_{scc}$ and $(v, C) \in E_{scc}$;
 - If $v, w \in V_{tr} \cup V_{pt}$, $(v, w) \in E$, and v, w do not belong to the same strongly connected component, then $(v, w) \in E_{scc}$.

4 Motion Planning on Digraphs

Throughout this section, let $D = (V, E)$ be a digraph, and $\mathcal{D}_{scc}(D) = (V_{scc}, E_{scc})$ ($V_{scc} = V_{tr} \cup V_{nt} \cup V_{pt}$) be the strongly-connected-component digraph of D .

Theorem 4. *Feasibility of motion planning on D can be decided in time $O(n^2m)$ where n, m are resp. the number of vertices and arcs of D .*

In the following, we design an algorithm to prove the theorem. We illustrate the main idea of the algorithm, but leave the correctness proof and the detailed complexity analysis to the full paper. We first introduce some notations.

Let C_s and C_t be the strongly connected components which s and t belong to respectively.

For each $v \in V$, let $h(v)$ denote the number of holes that are reachable from v , namely, to which there is a path from v in D .

Let V_{cr} denote the set of vertices $v \in V_{tr} \cup V_{pt}$ such that t is reachable from v , and v is reachable from s in D . The vertices in V_{cr} are called the *critical* vertices of the motion planning problem on D .

We consider motion planning on digraphs case by case:

Case I: C_s is trivial;

Case II: C_s is nontrivial and $C_s = C_t$;

Case III: C_s is nontrivial and $C_s \neq C_t$.

Note that since we assume that $s \neq t$, if C_s is trivial, then $C_s \neq C_t$.

4.1 Case I: C_s Is Trivial

We introduce some additional notations.

Let C be a nontrivial strongly connected component of D , $In(C)$ (resp. $Out(C)$) are used to denote the set of ports of C , i.e. vertices $v \in V_{pt} \cap V(C)$, such that there is some $w \in (V_{tr} \cup V_{pt}) \setminus V(C)$ satisfying that $(w, v) \in E$ (resp. $(v, w) \in E$). Vertices in $In(C)$ (resp. $Out(C)$) are called *inward* ports (resp. *outward* ports) of C . Note that $In(C) \cap Out(C)$ may be nonempty.

Let V_{cr}^{in} denote the set of vertices $v \in V_{cr}$ such that either $v \in V_{tr}$ and $v \neq s$, or $v \in In(C)$ for some nontrivial strongly connected component C . And let V_{cr}^{out} denote the set of vertices $v \in V_{cr}$ such that either $v \in V_{tr}$ and $v \neq t$, or $v \in Out(C)$ for some nontrivial strongly connected component C such that $C \neq C_t$.

For $v \in V_{cr}^{in}$, define $h_{in}(v)$ as follows:

Let $w \in V_{cr}$ such that $(w, v) \in E_{scc}$, imagine that the robot is in w . If the robot can be moved from w to t under the restriction that the first move of the robot is from w to v , then $h_{in}(v)$ is the minimal number of (distinct) holes used during the movement of the robot from w to t ; otherwise, $h_{in}(v) := \infty$.

For $v \in V_{cr}^{out}$, define $h_{out}(v)$ as follows:

Imagine that the robot is in v . If the robot can be moved from v to t under the restriction that the first move of the robot is from v to some $w \in V_{cr}$ such that $(v, w) \in E_{scc}$, then $h_{out}(v)$ is the minimal number of holes used during the movement of the robot from v to t ; otherwise, $h_{out}(v) := \infty$.

The algorithm for deciding the feasibility of motion planning on digraphs in **Case I** goes as follows: Starting from the vertices in $V_{cr} \cap V(C_t)$, compute $h_{in}(v)$ and $h_{out}(v)$ for all $v \in V_{cr}$ inductively in a backward way. When these computations are finished, the algorithm reports “yes” (the motion planning problem is feasible) iff $h_{out}(s) < \infty$.

Initial step: For vertices in $V_{cr} \cap V(C_t)$,

- If C_t is trivial, then $t \in V_{cr}^{in}$ and $t \notin V_{cr}^{out}$: if $h(t) \geq 1$, then $h_{in}(t) := 1$, otherwise $h_{in}(t) := \infty$;
- If C_t is nontrivial, then $In(C_t) \subseteq V_{cr}^{in}$ and $Out(C_t) \cap V_{cr}^{out} = \emptyset$: for $v \in In(C_t)$, if $h(v) \geq MinNum(C_t, v, t) + 1$, then $h_{in}(v) := MinNum(C_t, v, t) + 1$, otherwise, $h_{in}(v) := \infty$.

Remark 1. $MinNum(D, v, w)$ is used in [WG08] to compute the minimal number of holes used to move the robot from v to w in a strongly connected digraph D over all instances of the motion planning on D such that v, w are respectively the source and the destination. $MinNum(D, v, w)$ works as follows: If $v = w$, then return 0; Otherwise if v, w belong to the same strongly biconnected component of D , then return 1 (according to Theorem 1); Otherwise, let $P = B_0v_1B_1\dots B_{r-1}v_rB_r$ ($r \geq 1$) be the path in $\mathcal{G}_{sbc}(D)$ such that $v \in B_0$, $w \in B_r$, $v \neq v_1$ and $w \neq v_r$, and l be the maximal length of the chains of $\mathcal{G}_{sbc}(D)$ such that they are contained in P . Return $l + 1$. □

Induction step: For $v \in V_{cr} \cap V_{tr}$, if for each $w \in V_{cr}$ such that $(v, w) \in E_{scc}$, the computation of $h_{in}(w)$ has been finished, then

- $h_{out}(v) := \min\{h_{in}(w) | w \in V_{cr}, (v, w) \in E_{scc}\}$;
- if $v \neq s$: if $h(v) \geq h_{out}(v) + 1$, then $h_{in}(v) := h_{out}(v) + 1$, otherwise $h_{in}(v) := \infty$.

For each nontrivial strongly connected component C such that $C \neq C_t$ and $In(C) \cup Out(C) \subseteq V_{cr}$, if for each $v \in V_{cr} \cap Out(C)$ and each $w \in V_{cr}$ such that $(v, w) \in E_{scc}$, the computation of $h_{in}(w)$ has been finished, then

- for each $v \in V_{cr} \cap Out(C)$, $h_{out}(v) := \min\{h_{in}(w) | w \in V_{cr}, (v, w) \in E_{scc}\}$;
- for each $v \in V_{cr} \cap In(C)$, if $h(v) \geq \min\{MinNum(C, v, v') + h_{out}(v') + 1 | v' \in Out(C)\}$, then $h_{in}(v) := \min\{MinNum(C, v, v') + h_{out}(v') + 1 | v' \in Out(C)\}$, otherwise $h_{in}(v) := \infty$.

Example 1 (Case I: C_s is trivial). The digraph D is given in Fig 2(a), C_s is trivial, and the strongly-connected-component digraph of D is given in Fig 2(b). The critical vertices, V_{cr} , are those within the dashed cycle in Fig 2(b), $V_{cr}^{in} = \{v_1, v_4, t, v_8, v_9\}$ and $V_{cr}^{out} = \{s, v_2, v_3, v_4, v_7, v_{11}\}$. The $h(v)$'s are given in Fig 2(a) and pairs $(h_{in}(v), h_{out}(v))$ for $v \in V_{cr}$ are given in Fig 2(b). Because $h_{out}(s) = 4$, the motion planning problem is feasible. Four holes can be moved to v_8, v_9, v_{11} and t before moving the robot, then the robot can be moved from s to v_8 , and moved to v_9 by rotating around the cycle $v_8v_9v_{10}$, then to v_{11} , and finally to t . □

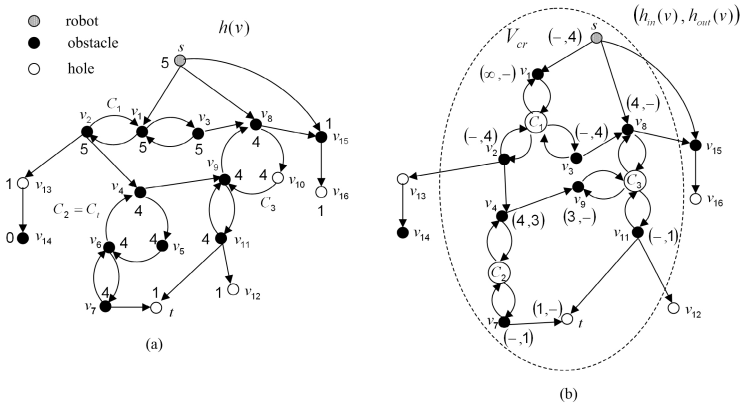


Fig. 2. Case I: C_s is trivial

4.2 Case II: C_s Is Nontrivial and $C_s = C_t$

Let inside (outside) holes denote the holes in some $v \in V(C_s)$ ($v \notin V(C_s)$).

We first use the algorithm for feasibility of motion planning on strongly connected digraphs in [WG08] to decide whether the inside holes are sufficient to move the robot from s to t . If it is, then report “yes”; otherwise, the motion planning problem may still be feasible since the outside holes can be moved into C_s and used to move the robot from s to t .

For each outside hole, there may be multiple ports of C_s through which the hole can be moved into C_s , we should choose carefully those ports, otherwise, the feasibility may be destroyed, which has been illustrated in Fig 1. We introduce a greedy strategy to move the outside holes into C_s to avoid this.

Before presenting the greedy strategy, we recall a definition about the relative positions of the vertices in [WG08].

Definition 3. Let $D = (V, E)$ be a strongly connected digraph, $\mathcal{G}_{sbc}(D) = (V_{sbc}, W_{sbc}, E_{sbc})$ be the strongly-biconnected-component graph of D , $u, v, w \in V$ and $v \neq w$. Then u is said to be on the w -side of v , if $u \neq v$ and one of the following two conditions holds:

1. $v \in W_{sbc}$ (v is shared by at least two strongly biconnected components of D), and u, w are in the same connected component of $\mathcal{G}(D - v)$.
2. $v \notin W_{sbc}$, and either u, w are in the same connected component of $\mathcal{G}(D - V(B))$, or $u \in V(B)$, where B is the unique strongly biconnected component of D to which v belongs.

Otherwise, u is said to be not on the w -side of v . And u is said to be on the non- w -side of v if u is not on the w -side of v and $u \neq v$.

In C_s , when we say that an inside hole is on the w -side of the robot, or on the non- w -side of the robot, and so on, we are talking about the positions of the inside hole and the robot.

An outside hole of C_s is said to be on the w -side (resp. non- w -side) of v if it can be moved into C_s through some port of C_s which is on the w -side (resp. non- w -side) of v . Note that an outside hole can be both on the w -side of v and on the non- w -side of v , since it can be moved into C_s both through some port on the w -side of v and through some other port on the non- w -side of v . An outside hole is said to be not on the w -side of v if it cannot be moved into C_s through some port on the w -side of v .

Let $h_{w-side}(v), h_{not-w-side}(v), h_{non-w-side}(v)$ denote respectively the number of (inside or outside) holes on the w -side of v , not on the w -side of v , and on the non- w -side of v .

Now we introduce the greedy strategy. The intuition of the strategy is that when it is necessary to move the robot away from t , first use the inside holes not on the t -side of the robot, then use the outside holes not on the t -side of the robot and farthest from t (the distance between an outside hole and t is the minimal distance between a port, through which the hole can be moved into C_s , and t).

If s, t do not belong to the same strongly biconnected component of C_s , then let $P := B_0v_1B_1 \cdots v_pB_p (p \geq 1)$ be a path in $\mathcal{G}_{sbc}(C_s) = (V_{sbc}, W_{sbc}, E_{sbc})$ such that $s \in V(B_0), s \neq v_1, t \in V(B_p), t \neq v_p$; otherwise, let $P := B_0$ and $p := 0$, where B_0 is the strongly biconnected component such that $s, t \in V(B_0)$.

We distinguish the following five cases,

1. $s \notin W_{sbc}$;
2. $s \in W_{sbc}$ and $h_{t-side}(s) \geq MinNum(C_s, s, t)$;
3. $s \in W_{sbc}, h_{t-side}(s) < MinNum(C_s, s, t)$ and $|V(B_0)| \geq 3$;
4. $s \in W_{sbc}, h_{t-side}(s) < MinNum(C_s, s, t), |V(B_0)| = 2$ and s is a branching vertex;
5. $s \in W_{sbc}, h_{t-side}(s) < MinNum(C_s, s, t), |V(B_0)| = 2$ and s is not a branching vertex.

In the following, we illustrate the greedy strategy by considering the **5th case**. The discussions of the other cases are similar and they are omitted due to space limitation.

Since $s \in W_{sbc}$ and s is not a branching vertex, there is a unique strongly biconnected component B such that $B \neq B_0$ and $s \in V(B)$.

If $t \in V(B_0)$, let $i_0 := 0$, otherwise, let

$$i_0 := \min(\{p\} \cup \{i : |V(B_i)| \geq 3, \text{ or } v_i \text{ is a branching vertex}\}).$$

We further distinguish the following four subcases,

Subcase 5.1. $h_{t-side}(s) \geq i_0 + 1$;

Subcase 5.2. $h_{t-side}(s) \leq i_0$ and $|V(B)| \geq 3$;

Subcase 5.3. $h_{t-side}(s) \leq i_0$, $|V(B)| = 2$ and B is a leaf of $\mathcal{G}_{sbc}(C_s)$;

Subcase 5.4. $h_{t-side}(s) \leq i_0$, $|V(B)| = 2$ and B is not a leaf of $\mathcal{G}_{sbc}(C_s)$.

Due to space limitation, we consider only Subcase 5.4. in the following.

Subcase 5.4. Because $h_{t-side}(s) \leq i_0$, it is necessary to move the robot away from t to move more holes to the t -side of the robot.

Let $v' \in V(B)$ such that $(s, v') \in E$, and $Q := B'_0 v'_1 B'_1 \dots v'_q B'_q$ ($q \geq 1$) be a path in $\mathcal{G}_{sbc}(C_s)$ such that

1. $B'_0 = B$;
2. either $|V(B'_q)| \geq 3$, or v'_q is a branching vertex, or B'_q is a leaf of $\mathcal{G}_{sbc}(C_s)$;
3. $\forall i : 1 \leq i < q, |V(B'_i)| = 2$, and v'_i is not a branching vertex.

Now we move the outside holes into C_s as follows:

Let $v'_0 = s$, then from $i = 1$ to $i = q$, do the following,

- If there are inside holes not on the t -side of v'_i , then move one such hole to v'_i , move the robot from v'_{i-1} to v'_i , and move the outside holes into C_s through v'_{i-1} as much as we can;
- If there are no inside holes not on the t -side of v'_i , but there are outside holes not on the t -side of v'_i , let k be the largest index such that there is at least one outside hole not on the t -side of v'_k , move one outside hole not on the t -side of v'_k into C_s , then to v'_i , move the robot from v'_{i-1} to v'_i , and move the outside holes into C_s through v'_{i-1} as much as we can.

Suppose the robot is in v'_r ($0 \leq r \leq q$) after the above loop.

If during the above loop, when the robot is in v'_i ($1 \leq i \leq r$), and the number of holes on the t -side of v'_i is $\geq i + i_0 + 1$, then: if $h(v'_i) \geq \text{MinNum}(C_s, v'_i, t)$, then report “yes”, otherwise report “no”.

Otherwise, there are two situations: $r < q$ or $r = q$.

In case of $r < q$: We must have $h_{not-t-side}(v'_{r+1}) = 0$. If $h_{t-side}(v'_r) > 0$ and $h_{not-t-side}(v'_r) > 0$, then move one hole on the t -side of v'_r to v'_{r-1} , move the robot from v'_r to v'_{r-1} , move one outside hole not on the t -side of v'_r into C_s through v'_r , then to v'_{r+1} , move the robot from v'_{r-1} to v'_{r+1} , now all the holes are on the t -side of v'_{r+1} , report “yes” iff $h(v'_{r+1}) \geq \text{MinNum}(C_s, v'_{r+1}, t)$. If $h_{t-side}(v'_r) = 0$, $h_{not-t-side}(v'_r) > 0$ and $h_{non-t-side}(v'_r) > 0$, then one outside hole not on the t -side of v'_r can be moved into C_s and to v'_{r+1} without moving

the robot, move the robot to v'_{r+1} , now all the holes are on the t -side of v'_{r+1} , report “yes” iff $h(v'_{r+1}) \geq \text{MinNum}(C_s, v'_{r+1}, t)$.

In case of $r = q$: then there are the following three possibilities:

- $|V(B'_q)| \geq 3$;
- $|V(B'_q)| = 2$ and v'_q is a branching vertex;
- $|V(B'_q)| = 2$, v'_q is not a branching vertex, and B'_q is a leaf.

The first possibility above is reduced to Subcase 5.2., the second possibility above is reduced to Case 4., and the third possibility above is reduced to Subcase 5.3.

In all the other situations of Subcase 5.4., report “no”.

Example 2. The motion planning problem is given in Fig 1(a). To preserve the feasibility, we need first move the holes not on the t -side of v_2 , i.e. the hole in v_1 , to v_2 , then move the robot to v_2 and move the outside holes into C_s through s as much as we can. So we move the two holes at v_7 and v_8 into C_s through s . Then all the holes are on the t -side of the robot, the problem is feasible iff the total number of holes is $\geq \text{MinNum}(C_s, v_2, t) = 3$. Thus the motion planning problem is feasible.

4.3 Case III: C_s Is Nontrivial and $C_s \neq C_t$

We can decide the feasibility in this case by combining the algorithms for Case I and Case II, the details are omitted and the algorithm will appear in the full paper.

5 Conclusions

In this paper, based on the work in [WG08] for motion planning on acyclic and strongly connected digraphs, we gave a complete solution to the feasibility of motion planning on digraphs. The most intricate part of this solution is to design a strategy to move the outside holes into C_s , the strongly connected component containing s , while not destroying the feasibility of the motion planning problem.

It would be interesting in the future to consider the optimization of motion planning on digraphs as well as the other variations of motion planning problem on digraphs, e.g. the reconfiguration problem that was considered on graphs in [KMS84].

References

- [BJG00] Bang-Jensen, J., Gutin, G.: Digraphs: Theory, Algorithms and Applications. Springer Monographs in Mathematics. Springer, Heidelberg (2000)
- [FK99] Finn, P.W., Kavmkit, L.E.: Computational approaches to drug design. *Algorithmica* 25, 347–371 (1999)
- [JJ06] Jetcheva, J.G., Johnson, D.B.: Routing characteristics of ad hoc networks with unidirectional links. *Ad Hoc Networks* 4(3), 303–325 (2006)

- [KMS84] Kornhauser, D., Miller, G., Spirakis, P.: Coordinating pebble motion on graphs, the diameter of permutation groups, and applications. In: FOCS 1984, pp. 241–250 (1984)
- [LaV06] LaValle, S.M.: Planning Algorithms. Cambridge University Press, Cambridge (2006)
- [MD02] Marina, M.K., Das, S.R.: Routing performance in the presence of unidirectional links in multihop wireless networks. In: Proc. of ACM MobiHoc, pp. 12–23 (2002)
- [MPG] Motion planning game,
http://www.download-game.com/Motion_Planning_Game.htm
- [Per88] Perrott, Y.: Track transportation systems. European patent (1988),
<http://www.freepatentsonline.com/EP0284316.html>
- [PRST94] Papadimitriou, C.H., Raghavan, P., Sudan, M., Tamaki, H.: Motion planning on a graph. In: FOCS 1994, pp. 511–520 (1994)
- [SA01] Song, G., Amato, N.M.: Using motion planning to study protein folding pathways. In: RECOMB 2001: Proceedings of the Fifth Annual International Conference on Computational Biology, pp. 287–296. ACM, New York (2001)
- [Wes00] West, D.B.: Introduction to Graph Theory, 2nd edn. Prentice-Hall, Englewood Cliffs (2000)
- [WG08] Wu, Z., Grumbach, S.: Feasibility of motion planning on acyclic and strongly connected directed graphs (manuscript, 2008)

Polynomial-Time Algorithm for Sorting by Generalized Translocations^{*}

Xiao Yin and Daming Zhu

School of Computer Science and Technology,
Shandong University, Jinan 250101, P.R. China
xyin@mail.sdu.edu.cn, dmzhu@sdu.edu.cn

Abstract. Translocation is a prevalent rearrangement event in the evolution of multi-chromosomal species which exchanges ends between two chromosomes. A translocation is reciprocal if none of the exchanged ends is empty; otherwise, non-reciprocal. Given two signed multi-chromosomal genomes A and B , the problem of sorting by translocations is to find a shortest sequence of translocations transforming A into B . Several polynomial algorithms have been presented, all of them only allowing reciprocal translocations. Thus they can only be applied to a pair of genomes having the same set of chromosome ends. Such a restriction can be removed if non-reciprocal translocations are also allowed. In this paper, for the first time, we study the problem of sorting by generalized translocations, which allows both reciprocal translocations and non-reciprocal translocations. We present an exact formula for computing the generalized translocation distance, which leads to a polynomial algorithm for this problem.

Keywords: Algorithm, genome rearrangement, translocation.

1 Introduction

Genome rearrangement is a common mode of molecular evolution in biological species [1, 2, 3, 4]. Although the rearrangement process is very complicated, there are three basic operations: *reversal*, *translocation* and *transposition*. In this paper, we study the translocation operations. Translocation is a prevalent rearrangement event in the evolution of multi-chromosomal species which exchanges ends between two chromosomes. A translocation is *reciprocal* if none of the exchanged ends is empty; otherwise, *non-reciprocal*.

The problem of *sorting by translocations* is defined as follows: given two signed multi-chromosomal genomes A and B , find a shortest sequence of translocations transforming A into B . The length of this sequence is called the *translocation distance* between A and B . Hannenhalli designed the first $O(n^3)$ algorithm [5] for *sorting by reciprocal translocations* (abbreviated as SRT) which only allows reciprocal translocations. Bergeron et al. [6] pointed out an error in Hannenhalli's sorting strategy and gave a new $O(n^3)$ algorithm for SRT. The time complexity was improved to $O(n^{3/2} \sqrt{\log(n)})$

^{*} Supported by (1) National Nature Science Foundation of China, 60573024. (2) Chinese National 973 Plan, previous special, 2005cca04500.

[7, 8, 9]. All of these algorithms assume that the two genomes have the same set of chromosome ends, which rarely happens in biology.

In this paper, we consider a more general case - when A and B have different chromosome ends. Clearly, in such a case, non-reciprocal translocations are needed. We call this problem *sorting by generalized translocations* (abbreviated as SGT). We present an exact formula for computing the generalized translocation distance, which leads to a polynomial algorithm for SGT. The basic idea is to transform SGT into SRT by “capping” A and B , by adding additional markers to the ends of chromosomes in the two genomes. The main idea is to find an optimal “capping” of B such that the solution to the resulting SRT instance is minimal.

2 Preliminaries

A genome is a set of chromosomes and a chromosome is a sequence of genes. Each gene is identified by an integer with a sign of '+' or '-' which denotes its direction. For example, $\{(3, -5), (2, 4, -6), (-1, 7)\}$ is a genome with three chromosomes and seven genes.

Given a sequence of genes $I = x_1, x_2, \dots, x_k$, the *reverse* of I is $-I = -x_k, -x_{k-1}, \dots, -x_1$. A chromosome is orientation-less. Therefore, a chromosome X is said to be *identical* to a chromosome Y if either $X=Y$ or $X=-Y$. Genomes A and B are said to be *identical* if they have the same set of chromosomes.

Let $X=(X_1, X_2)$ and $Y=(Y_1, Y_2)$ be two chromosomes, where X_1, X_2, Y_1, Y_2 are sequences of genes. A *translocation* cuts X into X_1 and X_2 and Y into Y_1 and Y_2 and exchanges segments between the chromosomes. There are two types of translocations. A *prefix-prefix* translocation switches X_1 with Y_1 resulting in $(Y_1, X_2), (X_1, Y_2)$. A *prefix-suffix* translocation switches X_1 with Y_2 resulting in $(-Y_2, X_2), (Y_1, -X_1)$. A translocation is *reciprocal* if none of X_1, X_2, Y_1 and Y_2 are empty. Otherwise, it is *non-reciprocal*. There are three kinds of non-reciprocal translocations: *fusion, fission* and *fission-fusion*. They can be regarded as special cases of reciprocal translocations where one or two segments are empty. A fusion of X and Y connects X and Y into one chromosome (X, Y) . It can be viewed as the translocation between (X, \emptyset) and (\emptyset, Y) , resulting in (X, Y) and a null chromosome (\emptyset, \emptyset) . A fission of X cuts X into two chromosomes (X_1) and (X_2) . It can be viewed as the translocation between (X_1, X_2) and (\emptyset, \emptyset) . A *fission-fusion* of X and Y cuts X into X_1 and X_2 , and then pastes one segment to Y , resulting in (X_1) and (Y, X_2) , or (X_1) and $(-X_2, Y)$, or (X_1, Y) and (X_2) , or $(Y, -X_1)$ and (X_2) . It can be viewed as the translocation between (X_1, X_2) and (Y, \emptyset) or the translocation between (X_1, X_2) and (\emptyset, Y) .

For a chromosome $X=(x_1, x_2, \dots, x_m)$, define $Tails(X)=\{x_1, -x_m\}$. $x_1, -x_m$ are *tails* of X . For a genome A , define $Tails(A)=\bigcup_{X \in A} Tails(X)$. Genomes A and B are *co-tailed* if $Tails(A)=Tails(B)$. Note that SRT is solvable only for co-tailed genomes.

2.1 Cycle Graph

Let A and B be a pair of co-tailed genomes. Let n and N be the number of genes and chromosomes in A (equivalently B). We will always assume that both A and B consist

of genes $\{1, 2, \dots, n\}$. For a chromosome $X=(x_1, x_2, \dots, x_m)$, replace each gene x_i by a pair of ordered vertices $(l(x_i), r(x_i))$. If the sign of x_i is '+', then $l(x_i)=x^l, r(x_i)=x^h$. If the sign of x_i is '-', then $l(x_i)=x^h, r(x_i)=x^l$. As a result, each chromosome in A and B corresponds to an ordered list of vertices as $l(x_1)r(x_1)l(x_2)r(x_2) \dots l(x_m)r(x_m)$. Vertices u and v are neighbors in A (B) if they are adjacent in the ordered list of a chromosome in A (B) constructed by afore mentioned method. For a gene x, x' and x^h are always neighbors, for simplicity, we exclude them from the definition of "neighbors". The bicolored cycle graph of A and B , denoted $G(A, B)$, is defined as follows. The set of vertices is $\bigcup_{i=1}^n \{i^l, i^h\}$. Vertices u and v are connected by a black edge if they are neighbors in A and are connected by a gray edge if they are neighbors in B .

A gray edge (u, v) in $G(A, B)$ is external if u, v belong to different chromosomes of A ; otherwise is internal. Each vertex in $G(A, B)$ has degree 0 or 2, where vertices of degree 0 (isolated vertices) belong to tails. Therefore, $G(A, B)$ can be uniquely decomposed into a number of disjoint cycles. A cycle is long if it contains at least two black edges, otherwise short. If $A=B$, then all cycles in $G(A, B)$ are short.

2.2 MSP and Even-Isolation

Considering a sequence $I = x_i, x_{i+1}, \dots, x_{j-1}, x_j$ in a chromosome of A . Let $V(I) = \bigcup_{i \leq k \leq j} \{x_k^l, x_k^h\}$, $IN(I) = V(I) \setminus \{l(x_i), r(x_j)\}$. I is a sub-permutation (SP) if there exists an sequence $I' = x_i, \text{permutation}(x_{i+1}, \dots, x_{j-1}), x_j$ in some chromosome in B and $\text{permutation}(x_{i+1}, \dots, x_{j-1}) \neq (x_{i+1}, \dots, x_{j-1})$. A minimal sub-permutation (MSP) is a SP not containing any other SP . A SP I can be viewed as a subgraph of $G(A, B)$ containing the vertex set $IN(I)$ such that: (1) there is no edge (u, v) such that $u \in IN(I), v \notin IN(I)$, (2) the subgraph corresponding to I has at least one long cycle.

There exists an even-isolation in $G(A, B)$ if the following two conditions hold: (1) all the MSP 's are contained in a single SP , (2) there are even number of MSP 's in $G(A, B)$.

2.3 Reciprocal Translocation Distance

Let $c(A, B)$ be the number of cycles and let $s(A, B)$ be the number of MSP 's in $G(A, B)$. Define $f(A, B)$: $f(A, B)=2$ if there exists an even-isolation in $G(A, B)$, $f(A, B)=1$ if $s(A, B)$ is odd and otherwise $f(A, B)=0$. Let $d_r(A, B)$ denote the reciprocal translocation distance between A and B .

Theorem 1. [5] $d_r(A, B) = n - N - c(A, B) + s(A, B) + f(A, B)$.

3 Capping the Genomes

We now turn to the general case when A and B might have different tails and different number of chromosomes. Let n be the number of genes in A (equivalently B). We still assume that both A and B consist of the genes $\{1, \dots, n\}$. Let M be the number of chromosomes in A and let N be the number of chromosomes in B . Suppose $A=\{X_1, X_2, \dots, X_M\}$, $B=\{Y_1, Y_2, \dots, Y_N\}$, where X_i is the i th chromosome of A and Y_i is the i th chromosome of B . We introduce additional $2n$ markers called caps: $C_k = n + k$, for $k = 1, 2, \dots, 2n$. Construct a pair of co-tailed genomes \hat{A} and \hat{B} with n chromosomes by adding $2n$ caps to A and B as follows: The capping of A is

$$\hat{A} = \{\hat{X}_1, \hat{X}_2, \dots, \hat{X}_n\}$$

where $\hat{X}_i=(C_{2i-1}, X_i, C_{2i})$ for $1 \leq i \leq M$ and $\hat{X}_i=(C_{2i-1}, C_{2i})$ for $M < i \leq n$. The *capping* of B is

$$\hat{B} = \{\hat{Y}_1, \hat{Y}_2, \dots, \hat{Y}_n\}$$

where $\hat{Y}_i=(-1)^{j+1}C_j, Y_i, (-1)^kC_k$ for $1 \leq i \leq N$, $\hat{Y}_i=(-1)^{j+1}C_j, (-1)^kC_k$ for $N < i \leq n$ and each cap appears exactly once in \hat{B} . The signs of C_j and C_k are assigned $(-1)^{j+1}$ and $(-1)^k$ is to ensure that \hat{A} and \hat{B} are co-tailed. Let \mathbb{B} be the set of all possible cappings of B . There are $n - N$ “null chromosomes” containing only two caps in \hat{B} . Thus $|\mathbb{B}|=\frac{(2n)!}{2^{(n-N)}(n-N)!}$. For example, let $A=\{(3, 4), (2, 1)\}$, $B=\{(1), (2), (3, 4)\}$, then the capping of A is $\{(5,3,4,6),(7,2,1,8),(9,10),(11,12)\}$, $|\mathbb{B}|=\frac{8!}{2}$ and one capping of B is $\{(5,1,6),(7,2,8),(9,3,4,10),(11,12)\}$.

For the rest of this paper, the word “generalized translocation” refers to either a reciprocal translocation or a non-reciprocal translocation. Note that each generalized translocation in A corresponds to a reciprocal translocation in \hat{A} , and each reciprocal translocation in \hat{A} either corresponds to a generalized translocation in A or only exchanges caps between two chromosomes in A . Let $d(A, B)$ denote the generalized translocation distance between A and B .

Theorem 2. *Let $\hat{B} \in \mathbb{B}$. Then $d(A, B) = \min\{d_r(\hat{A}, \hat{B})|\hat{B} \in \mathbb{B}\}$.*

Proof. Suppose $\rho_1, \rho_2, \dots, \rho_k$ is a shortest sequence of generalized translocations transforming A into B . Obviously, $\rho_1, \rho_2, \dots, \rho_k$ induces a sequence of reciprocal translocations of the same length transforming \hat{A} into some $\hat{B} \in \mathbb{B}$. Thus $k = d(A, B) \geq \min\{d_r(\hat{A}, \hat{B})|\hat{B} \in \mathbb{B}\}$.

Let $d_r(\hat{A}, \hat{B}_o) = \min\{d_r(\hat{A}, \hat{B})|\hat{B} \in \mathbb{B}\}$. Suppose $\hat{\rho}_1, \hat{\rho}_2, \dots, \hat{\rho}_t$ is a shortest sequence of reciprocal translocations transforming \hat{A} into \hat{B}_o . $\hat{\rho}_1, \hat{\rho}_2, \dots, \hat{\rho}_t$ induces a sequence of generalized translocations transforming A into B which length is no more than t , thus $\min\{d_r(\hat{A}, \hat{B})|\hat{B} \in \mathbb{B}\} = t \geq d(A, B)$. □

Let \hat{B}_o be a capping of B . \hat{B}_o is an *optimal capping* of B if $d_r(\hat{A}, \hat{B}_o)=\min\{d_r(\hat{A}, \hat{B})|\hat{B} \in \mathbb{B}\}$. Theorem 2 implies an algorithm for SGT which is exponential of n . The rest of this paper is going to find an optimal capping of B in polynomial time.

4 Path-Cycle Graph

Choose $\hat{B} \in \mathbb{B}$ arbitrarily and construct the cycle graph $G(\hat{A}, \hat{B})$. Define the *path-cycle graph*, denoted $G^P(A, B)$, that does not depend on the capping of B by deleting from $G(\hat{A}, \hat{B})$ the gray edges incident on vertices that belong to caps. These vertices adjacent to the deleted gray edges which belong to caps are called *A-caps*. The vertex on the other end of the deleted gray edge is called a *B-tail*, unless the gray edge arises from a null chromosome, in which case both its ends are A-caps. For example, let $A=\{(2, 1, 3, 5), (6, 8, -7, 4, 10, -9)\}$, $B=\{(1, 2, 3, 4), (5, 6, 7), (8, 9, 10)\}$, the cycle graph $G(\hat{A}, \hat{B})$ is shown in Fig. 1(a) and the path-cycle graph $G^P(A, B)$ is shown in Fig. 1(b). $G^P(A, B)$ has $2n$ A-caps and $2N$ B-tails. Each of the $\frac{(2n)!}{2^{(n-N)}(n-N)!}$ possible cappings of B corresponds to adding $n + N$ gray edges to $G^P(A, B)$, $2N$ of which join a A-cap and a B-tail, and the remaining $n - N$ of which join two A-caps.

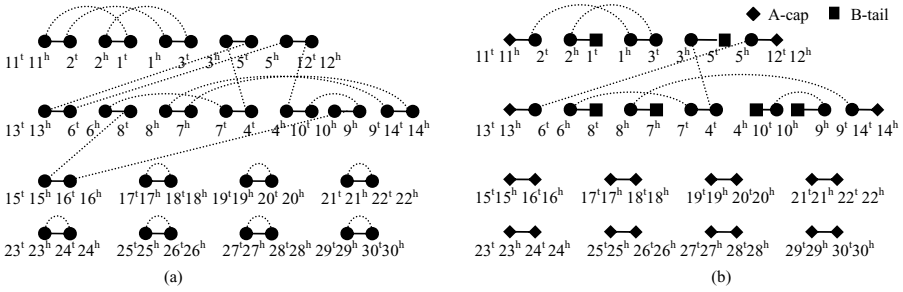


Fig. 1. Cycle graph and path-cycle graph

From theorem 2, the basic idea is to consecutively add $n + N$ gray edges to $G^P(A, B)$ such that the reciprocal translocation distance of the resulting cycle graph is minimal. Our analysis uses the distance formula in theorem 1. We need to study in detail the effect of each gray edge added to $G^P(A, B)$ and guarantee the value of $c - s - f$ of the resulting cycle graph is maximal.

Each vertex in $G^P(A, B)$ has degree 1 or 2 (Those isolated vertices are trivial and are excluded from our discussion), so the graph consists of vertex-disjoint cycles and paths. A cycle/path is *external* if it contains at least one external gray edge, otherwise *internal*. A cycle/path is *long* if it contains at least two black edges, otherwise *short*. If a path starts and ends with A-caps (B-tails), call it a *AA-path* (*BB-path*). If a path starts with a A-cap and ends with a B-tail, call it a *AB-path*. Let $l(A, B)$ be the total number of cycles and paths and let $p(A, B)$ be the number of BB-paths in $G^P(A, B)$.

Lemma 1. *Let $\hat{B} \in \mathbb{B}$, then $c(\hat{A}, \hat{B}) \leq l(A, B) - p(A, B)$.*

Proof. If each path in $G^P(A, B)$ can be “closed” by a gray edge (i.e., the two ends of the path are joined by a gray edge), then $c(\hat{A}, \hat{B}) = l(A, B)$. However, a BB-path can not be closed by a gray edge. It could only be connected with a AA-path or a AB-path by a gray edge. Thus $c(\hat{A}, \hat{B}) \leq l(A, B) - p(A, B)$. \square

5 Lower and Upper Bounds of $d(A, B)$

In this section, we present the lower and upper bounds for the generalized translocation distance. They will provide an intuition for the rather complicated formula for the generalized translocation distance presented in the next section.

Considering a sequence $I = x_i, x_{i+1}, \dots, x_{j-1}, x_j$ in a chromosome of \hat{A} , call I a *real-sub-permutation (RSP)* if the subgraph of $G^P(A, B)$ induced by vertex set $IN(I)$ satisfies the following conditions: (1) there is no edge (u, v) such that $u \in IN(I), v \notin IN(I)$, (2) the subgraph contains at least one long cycle, (3) the subgraph doesn’t contain any path. Call I a *semi-sub-permutation (SSP)* if the subgraph of $G^P(A, B)$ induced by $IN(I)$ satisfies: (1) there is no edge (u, v) such that $u \in IN(I), v \notin IN(I)$, (2) the subgraph contains at least one long path, (3) the subgraph does not contain a AA- or BB-path. A *real-minimal-sub-permutation (RMSP)* is a RSP not containing any other RSP. A *semi-minimal-sub-permutation (SMSP)* is a SSP not containing any other SSP or RSP. For

example, in Fig 1(b), the subgraph of $G^P(A, B)$ induced by $\{1^h, 2', 2^h, 1', 1^h, 3'\}$ is a *SMSP*. The difference between *SMSP* and *RMSP* is that *SMSP* contains AB-path while *RMSP* does not contain any path. Note that a *SMSP* can be turned into a *RMSP* by closing all the AB-paths in it. Let g be a gray edge added to $G^P(A, B)$. g destroys a *SMSP*, say S , if S is not a *SMSP* in the resulting graph. g merges a pair of *SMSP*'s, say S_1 and S_2 , if it joins a A-cap of a AB-path in S_1 and a B-tail of a AB-path in S_2 . Let $sm(A, B)$ be the number of *SMSP*'s and let $r(A, B)$ be the number of *RMSP*'s in $G^P(A, B)$.

Lemma 2. *Let $\hat{B} \in \mathbb{B}$, then $c(\hat{A}, \hat{B}) - s(\hat{A}, \hat{B}) \leq l(A, B) - p(A, B) - r(A, B) - \lceil \frac{sm(A, B)}{2} \rceil$.*

Proof. Suppose $n+N$ gray edges are consecutively added transforming $G^P(A, B)$ into $G(\hat{A}, \hat{B})$: $G^P(A, B) = G_0 \xrightarrow{s^1} G_1 \xrightarrow{s^2} \dots \xrightarrow{s_{n+N}} G_{n+N} = G(\hat{A}, \hat{B})$. For a graph G_i , the parameters l_i, p_i, r_i, sm_i are defined in the same way as for the graph $G^P(A, B)$. For a parameter ϕ , define $\Delta\phi_i = \phi_i - \phi_{i-1}$. Denote $\Delta_i = (l_i - p_i - r_i - \lceil \frac{sm_i}{2} \rceil) - (l_{i-1} - p_{i-1} - r_{i-1} - \lceil \frac{sm_{i-1}}{2} \rceil)$.

Note that $c(\hat{A}, \hat{B}) = l_{n+N}, s(\hat{A}, \hat{B}) = r_{n+N}, p_{n+N} = sm_{n+N} = 0$. If $\Delta_i \leq 0$ for $1 \leq i \leq n+N$, then $c(\hat{A}, \hat{B}) - s(\hat{A}, \hat{B}) = l_{n+N} - p_{n+N} - r_{n+N} - \lceil \frac{sm_{n+N}}{2} \rceil \leq l_0 - p_0 - r_0 - \lceil \frac{sm_0}{2} \rceil = l(A, B) - p(A, B) - r(A, B) - \lceil \frac{sm(A, B)}{2} \rceil$. We will prove $\Delta_i \leq 0$ for $1 \leq i \leq n+N$. For a fixed i , ignore the index i , i.e., denote $\Delta = \Delta_i, \phi = \phi_i$. Depending on the edge g_i the following seven cases are possible:

Case 1: Edge g_i closes a AB-path. If a *SMSP* in G_{i-1} turns into a *RMSP* in G_i then $\Delta l = \Delta p = 0, \Delta r = 1, \Delta sm = -1$, so $\Delta \leq 0$. If the number of *SMSP*'s is not changed then no other parameter is affected and $\Delta = 0$.

Case 2: Edge g_i connects a A-cap with a B-tail in different AB-paths. This edge destroys at most two *SMSP*'s and $\Delta l = -1, \Delta p = \Delta r = 0, \Delta sm \geq -2$, so $\Delta \leq 0$.

Case 3: Edge g_i connects a AA-path with a BB-path. This edge can not destroy any *SMSP*'s and $\Delta l = \Delta p = -1, \Delta r = 0, \Delta sm \geq 0, \Delta \leq 0$.

Case 4: Edge g_i connects a B-tail in a AB-path with a A-cap in a AA-path (or a A-cap in a AB-path with a B-tail in a BB-path, or a A-cap in a AB-path with a A-cap in a AA-path). This edge destroys at most one *SMSP* and $\Delta l = -1, \Delta p = \Delta r = 0, \Delta sm \geq -1, \Delta \leq 0$.

Case 5: Edge g_i closes a AA-path. This edge can't produce new *RMSP*, so no parameter is affected and $\Delta = 0$.

Case 6: Edge g_i connects two A-caps in different AB-paths. This edge destroys at most two *SMSP*'s and $\Delta l = -1, \Delta p = 1, \Delta r = 0, \Delta sm \geq -2, \Delta \leq -1$.

Case 7: Edge g_i connects two A-caps in different AA-paths. Then $\Delta l = -1, \Delta p = \Delta r = \Delta sm = 0, \Delta = -1$. □

Let S_1, S_2 be two *SMSP*'s in the same chromosome of \hat{A} , S_1 and S_2 are *interdependent* if: (1) S_1, S_2 are contained in a single *SSP*, (2) there exists no *RMSP* between S_1 and S_2 . A pair of interdependent *SMSP*'s are shown in Fig 2(a). Let P_1 be an internal AA-path and P_2 be an internal BB-path in the same chromosome of \hat{A} , say X . P_1 and P_2 are *interdependent* if: (1) X contains only P_1 and P_2 and no other paths, (2) X contains no *RMSP* and no vertices of external gray edges. A pair of interdependent paths are shown in Fig 2(b). Note that adding a gray edge merging a pair of interdependent *SMSP*'s turns

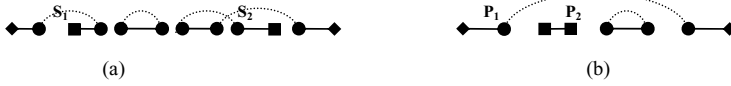


Fig. 2. (a) S_1 and S_2 are a pair of interdependent SMSP's, (b) P_1 and P_2 are a pair of interdependent paths

them into a larger SMSP, and adding a gray edge connecting a pair of interdependent paths produces a new SMSP.

For $G^P(A, B)$, define the parameter $ds(A, B)$: $ds(A, B)=1$ if $sm(A, B)=2$ and two SMSP's are interdependent and otherwise, $ds(A, B)=0$. Taking $ds(A, B)$ into account will lead to a tighter bound for $d(A, B)$.

Lemma 3. *If $ds(A, B)=0$, then there exist $\lfloor \frac{sm(A, B)}{2} \rfloor$ gray edges, each destroying two non-interdependent SMSP's.*

Proof. It suffices to describe a method for finding $\lfloor \frac{sm(A, B)}{2} \rfloor$ such gray edges. Suppose $ds(A, B)=0$ and let v be the number of pairs of interdependent SMSP's.

If $v=1$, then there must exist a SMSP, say S , not interdependent with any other SMSP's (Otherwise, $ds(A, B)=1$). Add a gray edge merging S and a SMSP in the pair of interdependent SMSP's. If $v>1$, number the v pairs of interdependent SMSP's as $1, 2, \dots, v$. For $1 \leq i \leq v-1$, add a gray edge merging a SMSP in the i th pair and a SMSP in the $(i+1)$ th pair. Then any pair of the remaining SMSP's are non-interdependent. Arbitrarily choose two SMSP's and add a gray edge merging them, until the number of SMSP's is less than 2. Each edge added by this method merges two non-interdependent SMSP's, the lemma holds. \square

Lemma 4. *There exist $p(A, B)$ gray edges, each connecting a BB-path with a AA-path without producing a new SMSP.*

Proof. It suffices to describe a method for finding $p(A, B)$ such gray edges. Let w be the number of pairs of interdependent paths in $G^P(A, B)$.

If $w=1$, there must exist a chromosome in A containing more than one genes. Thus $n>M$, implying that there exists a AA-path, say P , not interdependent with any BB-path. Add a gray edge connecting P with the BB-path in the pair of interdependent paths. If $w>1$, number the w pairs of interdependent paths as $1, 2, \dots, w$. For $1 \leq i \leq w-1$, add a gray edge connecting a AA-path in the i th pair and a BB-path in the $(i+1)$ th pair. Then any pair of the remaining paths are non-interdependent. For each BB-path, arbitrarily choose a AA-path and add a gray edge joining them, until there exists no BB-path. Each edge added by this method connects a pair of non-interdependent paths, and clearly no new SMSP is produced, the lemma holds. \square

Lemma 5. $max\{c(\hat{A}, \hat{B})-s(\hat{A}, \hat{B})|\hat{B} \in \mathbb{B}\} = l(A, B) - p(A, B) - r(A, B) - \lfloor \frac{sm(A, B)}{2} \rfloor - ds(A, B)$.

Proof. Similar to the proof of lemma 2, we consider a transformation of $G^P(A, B)$ into $G(\hat{A}, \hat{B})$ defined by $n+N$ gray edges: $G^P(A, B) = G_0 \xrightarrow{g^1} G_1 \xrightarrow{g^2} \dots \xrightarrow{g^{n+N}} G_{n+N} = G(\hat{A}, \hat{B})$. The parameters $l_i, p_i, r_i, sm_i, ds_i$ and Δ_i are defined in the same way as in Lemma 2. Denote $\Delta_i^{(1)} = (l_i - p_i - r_i - \lfloor \frac{sm_i}{2} \rfloor - ds_i) - (l_{i-1} - p_{i-1} - r_{i-1} - \lfloor \frac{sm_{i-1}}{2} \rfloor - ds_{i-1})$. Below we prove that $\Delta_i^{(1)} \leq 0$ for $1 \leq i \leq n+N$. For a fixed i , ignore the index i .

If $ds_{i-1}=0$, clearly $\Delta ds \geq 0$. By lemma 2, $\Delta \leq 0$, so $\Delta^{(1)} = \Delta - \Delta ds \leq 0$. If $ds_{i-1}=1$, then there exist a pair of interdependent *SMSP*'s, say S_1, S_2 , in G_{i-1} . If $\Delta ds=0$, clearly $\Delta^{(1)} = \Delta - \Delta ds \leq 0$. If $\Delta ds=-1$, then the following five cases are possible:

Case 1: Edge g_i closes a *AB*-path in S_1 (or S_2). S_1 (or S_2) turns into a *RMSP* in G_i and $\Delta l = \Delta p = 0, \Delta r = 1, \Delta sm = -1$, so $\Delta^{(1)} = 0$.

Case 2: Edge g_i merges S_1 and S_2 into a larger *SMSP*. Then $\Delta l = -1, \Delta p = \Delta r = 0, \Delta sm = -1$, $\Delta^{(1)} = 0$.

Case 3: Edge g_i connects a *AB*-path in S_1 (or S_2) with a path not in a *SMSP*. S_1 (or S_2) is destroyed and $\Delta l = -1, \Delta p \geq 0, \Delta r = 0, \Delta sm = -1, \Delta^{(1)} = 0$.

Case 4: Edge g_i connects two *A*-caps in *AB*-paths in S_1 and S_2 . S_1 and S_2 are both destroyed and $\Delta l = -1, \Delta p = 1, \Delta r = 0, \Delta sm = -2, \Delta^{(1)} = 0$.

Case 5: Edge g_i connects a pair of interdependent paths. This edge produces a new *SMSP* and $sm_i = 3$. Therefore, $\Delta l = \Delta p = -1, \Delta r = 0, \Delta sm = 1, \Delta^{(1)} = 0$.

We now prove that there exists a capping of B , say \hat{B} , such that $c(\hat{A}, \hat{B}) - s(\hat{A}, \hat{B}) = l(A, B) - p(A, B) - r(A, B) - \lceil \frac{sm(A, B)}{2} \rceil - ds(A, B)$, by constructing a sequence of $n+N$ gray edges g_1, g_2, \dots, g_{n+N} such that $\Delta_i^{(1)} = 0$ for all $1 \leq i \leq n+N$.

Assume that the first $i-1$ such edges are already added to $G^P(A, B)$ and we get G_{i-1} .

If G_{i-1} has at least two *SMSP*'s (i.e., $sm_{i-1} > 1$), consider two sub-cases. If $ds_{i-1} = 1$, let g_i be the edge merging two *SMSP*'s. Since the two *SMSP*'s are interdependent, g_i may produce a larger *SMSP*. Therefore, $\Delta l = -1, \Delta p = \Delta r = 0, \Delta sm = \Delta ds = -1$, thus $\Delta^{(1)} = 0$. If $ds_{i-1} = 0$, by lemma 3, there exist $\lfloor \frac{sm_{i-1}}{2} \rfloor$ gray edges, each edge destroying two non-interdependent *SMSP*'s. Thus $\Delta l = -1, \Delta p = \Delta r = 0, \Delta sm = -2, \Delta ds = 0$ and $\Delta^{(1)} = 0$ for each of them.

If G_{i-1} has only one *SMSP*, let P be a *AB*-path in it. If P is the only *AB*-path in the *SMSP*, then the gray edge g_i closing P satisfies $\Delta l = \Delta p = 0, \Delta r = 1, \Delta sm = -1, \Delta ds = 0$, $\Delta^{(1)} = 0$. Otherwise, $\Delta l = \Delta p = \Delta r = \Delta sm = \Delta ds = 0$ and $\Delta^{(1)} = 0$.

If G_{i-1} has no *SMSP* and has at least one *BB*-path (i.e., $p_{i-1} > 0$), by lemma 4, there exists p_{i-1} gray edges, each connecting a *BB*-path with a *AA*-path without producing a new *SMSP*. Clearly $\Delta l = \Delta p = \Delta r = \Delta sm = \Delta ds = 0$ and $\Delta^{(1)} = 0$ for each of them.

If G_{i-1} has neither a *BB*-path, nor a *SMSP*, let g_i be an edge closing an arbitrary *AB*-path or *AA*-path. Since the closed path does not belong to a *SMSP*, no parameter is affected and $\Delta^{(1)} = 0$. □

Theorem 3. Let $h = 2n - l(A, B) + p(A, B) + r(A, B) + \lceil \frac{sm(A, B)}{2} \rceil + ds(A, B)$. Then $h \leq d(A, B) \leq h + 2$.

Proof. Since $f(\hat{A}, \hat{B}) \in \{0, 1, 2\}$, $0 \leq \max\{c(\hat{A}, \hat{B}) - s(\hat{A}, \hat{B}) | \hat{B} \in \mathbb{B}\} - \max\{c(\hat{A}, \hat{B}) - s(\hat{A}, \hat{B}) - f(\hat{A}, \hat{B}) | \hat{B} \in \mathbb{B}\} \leq 2$. By theorem 1, theorem 2 and lemma 5, $d(A, B) = \min\{d_r(\hat{A}, \hat{B}) | \hat{B} \in \mathbb{B}\} = 3n - n - \max\{c(\hat{A}, \hat{B}) - s(\hat{A}, \hat{B}) - f(\hat{A}, \hat{B}) | \hat{B} \in \mathbb{B}\} = h + \max\{c(\hat{A}, \hat{B}) - s(\hat{A}, \hat{B}) | \hat{B} \in \mathbb{B}\} - \max\{c(\hat{A}, \hat{B}) - s(\hat{A}, \hat{B}) - f(\hat{A}, \hat{B}) | \hat{B} \in \mathbb{B}\}$. Therefore, $h \leq d(A, B) \leq h + 2$. □

Lemma 5 and theorem 3 lead to an approximation algorithm that sorts A into B using at most $d(A, B)+2$ generalized translocations. The algorithm is given as *Generalized_Sorting_I*(A, B).

Algorithm *Generalized_Sorting_I*(A, B)

1. construct the path-cycle graph $G = G^P(A, B)$
2. **while** G contains a path
3. **if** G has more than 2 *SMSP*'s
4. let sm be the number of *SMSP*'s, add $\lfloor \frac{sm}{2} \rfloor$ gray edges, each destroying two non-interdependent *SMSP*'s (lemma 3)
5. **else if** G has 2 *SMSP*'s
6. add a gray edge merging the two *SMSP*'s
7. **else if** G has 1 *SMSP*
8. add a gray edge closing the AB-path in this *SMSP*
9. **else if** there exists a BB-path in G
10. let p be the number of BB-paths, add p gray edges, each connecting a BB-path with a AA-path without producing new *SMSP* (lemma 4)
11. **else**
12. add a gray edge closing arbitrary path
13. let \hat{B} be a capping of B defined by G
14. solve SRT on \hat{A} and \hat{B}
15. sort A into B by using the sorting of \hat{A} into \hat{B}

6 The Generalized Translocation Distance

In this section, we present an exact formula for $d(A, B)$, which leads to a polynomial-time algorithm for SGT.

There exists a *real-even-isolation* in $G^P(A, B)$ if the following conditions hold: (1) $r(A, B)$ is even, (2) all the *RMSP*'s are contained in a single *RSP*.

There exists a *semi-even-isolation* in $G^P(A, B)$ if the following conditions hold: (1) $G^P(A, B)$ contains no real-even-isolation, (2) $r(A, B)$ is even, (3) all the *RMSP*'s are contained in a single *SSP*.

There exists a *strong-even-isolation* in $G^P(A, B)$ if the following conditions hold: (1) $G^P(A, B)$ contains a real-even-isolation, (2) there exist two *SMSP*'s which are contained in a single *SSP* with the real-even-isolation.

There exists a *weak-even-isolation* in $G^P(A, B)$ if the following conditions hold: (1) $r(A, B)$ is odd, (2) all the *RMSP*'s are on a single chromosome of \hat{A} , say X , (3) there exists a *SMSP* in X which is contained in a single *SSP* together with all the *RMSP*'s.

Note that, $G^P(A, B)$ contains at most one real-, semi-, strong- or weak-even-isolation.

Define $o(A, B) \in \{0, 1\}$: $o(A, B) = 1$ if $r(A, B)$ is odd, otherwise $o(A, B) = 0$. Define $\delta(A, B) \in \{0, 1, 2\}$ as follows. $\delta(A, B) = 2$ iff at least one of the following are satisfied:

($\alpha 1$) $G^P(A, B)$ contains a real-even-isolation and $sm(A, B) = 0$.

($\alpha 2$) $G^P(A, B)$ contains a strong-even-isolation and $sm(A, B) = 2$.

If $\delta(A, B) \neq 2$ then $\delta(A, B) = 1$ iff at least one of the following is satisfied:

($\beta 1$) $G^P(A, B)$ contains a real-even-isolation.

($\beta 2$) $G^P(A, B)$ contains a weak-even-isolation and $sm(A, B) = 1$.

(β_3) $G^P(A, B)$ contains a semi-even-isolation and $sm(A, B)$ is even.

(β_4) $G^P(A, B)$ contains a semi-even-isolation, $sm(A, B) = 1$ and the only one *SMSP* is contained in a single *SSP* with the semi-even-isolation.

(β_5) $ds(A, B) = 1$ and $o(A, B) = 0$.

If $\delta(A, B) \neq 1, 2$, then $\delta(A, B) = 0$.

Theorem 4. *The generalized translocation distance between A and B is $2n - l(A, B) + p(A, B) + r(A, B) + \lceil \frac{sm(A, B) + o(A, B)}{2} \rceil + \delta(A, B)$.*

The proof of theorem 4 is similar to the proofs of lemma 5 and theorem 3. It is by a case analysis of the change in each of the parameters l , p , r , sm , o and δ , for each gray edge added to $G^P(A, B)$, and hence is quite involved. It leads to a $O(n)$ algorithm for finding an optimal capping of B . This algorithm can be viewed as an extension of steps 2-12 of algorithm *Generalized_Sorting_I(A, B)* that includes a constant number of additional operations considering o and δ .

Theorem 5. *SGT can be solved in $O(n^{3/2} \sqrt{\log(n)})$ time.*

Proof. Finding an optimal capping of B , say \hat{B}_o , by adding $n + N$ gray edges to $G^P(A, B)$ can be done in $O(n)$ time. Sorting \hat{A} into \hat{B}_o by using the algorithm in [9] can be implemented in $O(n^{3/2} \sqrt{\log(n)})$ time. Therefore, SGT can be solved in $O(n^{3/2} \sqrt{\log(n)})$ time. \square

References

1. Bafna, V., Pevzner, P.: Sorting by reversals: Genome rearrangements in plant organelles and evolutionary history of x chromosome. *Molecular Biology Evolution* 12, 239–246 (1995)
2. Hannenhalli, S., Pevzner, P.: Transforming men into mice: Polynomial algorithm for genomic distance problem. In: Proc. 36th Ann. Symp. Foundations of Computer Science (FOCS 1995), pp. 581–592 (1995)
3. Hannenhalli, S., Pevzner, P.: Transforming cabbage into turnip: Polynomial algorithm for sorting signed permutations by reversals. *Journal of the ACM* 46, 1–27 (1999)
4. Kececioglu, J.D., Ravi, R.: of mice and men: Algorithms for evolutionary distance between genomes with translocation. In: Proc. 6th Annu. ACM-SIAM Symp. Discrete Algorithms (SODA 1995), pp. 604–613 (1995)
5. Hannenhalli, S.: Polynomial algorithm for computing translocation distance between genomes. *Discrete Applied Mathematics* 71, 137–151 (1996)
6. Bergeron, A., Mixtacki, J., Stoye, J.: On sorting by translocations. *Journal of Computational Biology* 13, 567–578 (2006)
7. Zhu, D.M., Ma, S.H.: Improved polynomial-time algorithm for computing translocation distance between genomes. *The Chinese Journal of Computers* 25, 189–196 (2002)
8. Wang, L.S., Zhu, D.M., Liu, X.W., Ma, S.H.: An $O(n^2)$ algorithm for signed translocation. *Journal of Computer and System Sciences* 70, 284–299 (2005)
9. Ozery-Flato, M., Shamir, R.: An $n^{3/2} \sqrt{\log(n)}$ algorithm for sorting by reciprocal translocations. In: Proc. 17th Ann. Symp. Combinatorial Pattern Matching (CPM 2006), pp. 258–269 (2006)

The Two-Guard Polygon Walk Problem

(Extended Abstract)

John Z. Zhang

Department of Mathematics and Computer Science
University of Lethbridge
Lethbridge, AB, Canada T1K 3M4
zhang@cs.uleth.ca

Abstract. Consider a simple polygon. A walk is conducted by two guards on the polygon boundary. They start at a boundary point and walk on the boundary. It is required that the two guards maintain their mutual visibility at all times and eventually meet together again. A polygon may or may not be walkable, depending on where the two guards start their walk or no matter where they start on the boundary. In this work, we characterize the class of walkable polygons by two guards by presenting a set of forbidden patterns.

1 Introduction

Imagine that two *guards* walk on the boundary of a *simple polygon*. In order to protect themselves, each guard is equipped with a vision device such that there is always a light beam between them, i.e., the two guards maintain their mutual visibility at all times. Given a polygon, the two-guard polygon walk problem asks whether it is possible for the two guards to start at a boundary point, walk on its boundary, and eventually meet together again.

The two-guard walk problem was first proposed in [1] in the context of *streets*. Two points on the boundary of a polygon, the *entrance* and the *exit*, are prespecified. The two guards start moving in the opposite directions from the entrance along the boundary, while maintaining their mutual visibility. They may walk backwards from time to time, as long as each of them does not walk beyond the entrance and exit. The search completes when they meet at the exit. A street is said to be *walkable* if it is possible for the two guards to conduct such a walk. Heffernan [2] proposed a linear-time algorithm to check whether a street is walkable. In [3], Crass *et al.* studied an ∞ -searcher in an open-edge “corridor”, which uses edges as the entrance and the exit. An ∞ -searcher has a vision of 360° .

The two-guard walk problem was also studied in the setting of *rooms* [4,5,6,7,8]. A room is a simple polygon with a designated point on its boundary, called the *door*, which is like the entrance in a street walk. However, no exit is prespecified. The two guards start at the door and walk on the room boundary in the opposite direction as in a street walk and eventually meet somewhere again [6]. It is required that no inside intruders shall escape the room through its door. More recent work can be found in [9,10,11].

In a more general sense, the above two problems are under the framework of *polygon search* problems, formulated by Suzuki and Yamashita [12]. A *k-searcher* is equipped with a vision device that emits k light beams. The purpose of a search is to detect any intruders by eventually illuminating them. Among those models, there have been results for searching a polygon by a 1-searcher. For instance, [13,14] introduced a different set of forbidden geometric patterns than those in [15,16,17] that render a polygon non-searchable. LaValle *et al.* [18] proposed an algorithmic approach for deciding searchable polygons.

Following the strain of the two-guard walk problems, we consider in this work the problem of walking a polygon by two guards. To match its counterpart in streets and rooms, we call it the *two-guard polygon walk* problem. Though seemingly similar to the polygon search problem by a 1-searcher, the two-guard polygon search problem has its own characteristics. In Sec. 2, we introduce the notation used throughout the paper and formally define the two-guard polygon walk problem. The paper is structured as follows. In Sec. 3, we present a characterization of walkable polygons by two guards. In particular, we propose a set of forbidden geometric patterns. Sec. 4 compares a 1-searcher and two guards in searching or walking a polygon. Finally in Sec. 5, we summarize our work and discuss some future tasks.

2 Preliminaries

2.1 Notation

A simple polygon P is defined by a clockwise sequence of distinct *vertices* numbered $1, 2, \dots, n$, ($n \geq 3$), and n *edges*, connecting adjacent vertices. The edge between vertices u and v is denoted by (u, v) . The *boundary* of P , denoted by ∂P , consists of all its vertices and edges. We consider that ∂P is part of the polygon. That is, a polygon is composed of its boundary and its interior. The vertices immediately preceding and succeeding vertex v clockwise are denoted by $Pred(v)$ and $Succ(v)$, respectively. For any two points $a, b \in \partial P$, the open and closed portions of ∂P from a to b clockwise are denoted by $\partial P(a, b)$ and $\partial P[a, b]$, respectively.

A vertex whose interior angle between its two incident edges in the polygon is more than 180° is called a *reflex vertex*. Consider a reflex vertex r . Extend edge $(Succ(r), r)$ toward the interior of P , and let $B(r) \in \partial P$ denote the *backward extension point*, where the extension leaves P for the first time. The polygonal area formed by $\partial P[r, B(r)]$ and chord $\overline{rB(r)}$ is called the *clockwise component* associated with r , and is denoted by $C_{cw}(r)$. Similarly, the extension of $(Pred(r), r)$ determines the *forward extension point*, $F(r)$, and the *counterclockwise component*, $C_{ccw}(r)$, associated with r is bounded by $\partial P[F(r), r]$ and chord $\overline{rF(r)}$.

2.2 The Polygon Walk Problem

Two points, u and v , inside polygon P are said to be *mutually visible* if the line segment \overline{uv} is completely contained inside P .

Polygon P is walkable by two guards, represented as L and R , respectively, if there exist a time constant T and two continuous functions, $\ell : [0, T] \rightarrow \partial P$ and $r : [0, T] \rightarrow \partial P$, such that

- (a) $\ell(0) = r(0) \in \partial P$ and $\ell(T) = r(T) \in \partial P$;
- (b) For any $t \in (0, T)$, $\ell(t) \neq r(t)$, and $\ell(t)$ and $r(t)$ are mutually visible. □

Functions $\ell(t)$ and $r(t)$ represent the positions of L and R on ∂P at $t \in [0, T]$, respectively. If a polygon is walkable, for any $t \in [0, T]$, P is partitioned into two portions by the line segment $\overline{\ell(t)r(t)}$ (i.e., the light beam between the two guards), such that one of them is always cleared of any intruders while the other is contaminated, i.e., it contains intruders. We call $(\ell(t), r(t))$, where $t \in [0, T]$, a walk schedule. Essentially, a walk schedule for the two guards is to enlarge the cleared portion and shrink the contaminated one, until the area of the latter becomes zero, i.e., the two guards meet again at some same boundary point.

Initially, $\ell(0)$ and $r(0)$ are at some same point on ∂P . We call it the walk start point (WSP for short). Beyond this point, the guards walk on the boundary and, if possible, eventually meet again at a walk end point (WEP for short) at time T . Note that for a walk, the WSP may or may not be the same as the WEP.

Not every boundary point can be used as a WSP or a WEP. In addition, if a WSP is selected, depending on the geometric structures of ∂P , we may only select a WEP on some specific boundary portion. Also if we select a WSP, it might be that the two guards cannot finish their walk, though the polygon is walkable. We will repeatedly resort to these facts in our discussions.

For technical reasons, for reflex vertex u , we use two imaginary points u_r and u_l that are sufficiently close to u to represent it. Viewed from u towards the interior of P , u_r is to its right while u_l is to its left.

If a polygon is walkable, i.e., there exists a time T , such that Conditions (a) and (b) are satisfied, we can let the time count down from T to 0 and the two conditions are still satisfied. The following is immediate.

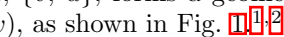
Proposition 1. *If a polygon is walkable by two guards starting at a WSP and ending at a WEP, the polygon is also walkable by the two guards starting at the WEP and ending at the WSP.* □

The following fact is also helpful in our discussions.

Proposition 2. *In order for two guards to walk a component due to a reflex vertex, they should be present in it simultaneously at some time instant.* □

2.3 Basic Geometric Patterns

Some basic geometric patterns play a key role in our characterization of walkable polygons. We discuss them and their properties in this section.

A pair of reflex vertices, $\{v, u\}$, forms a geometric pattern called a trap, if $v \notin C_{cw}(u)$ and $u \notin C_{ccw}(v)$, as shown in Fig. 

¹ In the following discussions, we only mark in figures the relevant vertices for the sake of clear presentation.

² Note that it might be that $C_{cw}(u) \cap C_{ccw}(v) = \emptyset$.

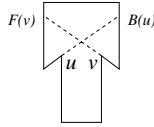


Fig. 1. A trap

Lemma 1. *A point on $\partial P[v_l, u_r]$ in a trap in Fig. 7 can be used as neither a WSP nor a WEP.*

Proof. Consider that L and R start their walk at a point on $\partial P(v_l, u_r)$. Now in order for them to clear component $C_{cw}(u)$ and maintain their mutual visibility, they are required to be in the component simultaneously at some time instant, due to Prop. 2. Suppose that L is at u_r , while R is at v_l . At this moment, in order for R to be inside $C_{cw}(u)$, it has to walk on the boundary counterclockwise, until $B(u)$. However, to maintain mutual visibility, this requires that L be in component $C_{ccw}(v)$ at the same time. This leads to contradictory requirements on the boundary positions of L and R . Note that it cannot be true that L and R are at u at the same time, since it violates Condition (b) in the problem definition of a polygon walk. If a walk ends at a point on $\partial P(v_l, u_r)$, due to Prop. 1, the point can also be used as the corresponding WSP, a contradiction to the above arguments. \square

In Fig. 1, each point on $\partial P[v_l, u_r]$ is said to be covered by a trap.

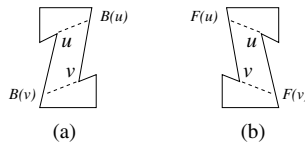


Fig. 2. (a) A DPcw; (b) A DPccw

Two reflex vertices u and v form a *disjoint clockwise component pair* (DPcw, for short), if $C_{cw}(u) \cap C_{cw}(v) = \emptyset$. Analogously, they form a *disjoint counterclockwise component pair* (DPccw, for short), if $C_{ccw}(u) \cap C_{ccw}(v) = \emptyset$. Fig. 2 shows a DPcw and a DPccw, respectively.

Lemma 2. *For a walkable polygon containing a DPcw in Fig. 2(a), for $\partial P[u_l, v_r]$ and $\partial P[v_l, u_r]$, if we select a WSP (WEP, resp.) on one of them, we can only select a WEP (WSP, resp.) on the other. An analogous property holds for a DPccw in Fig. 2(b).*

Proof. Suppose that, for a walkable polygon containing a DPcw, we select a WSP on $\partial P[u_l, v_r]$. If the WSP is on $\partial P[B(u), v_r]$, in order to clear $C_{cw}(v)$, L and R have to be both in it at some time instant. They cannot both walk clockwise, since around reflex vertex v , they have to be on the two adjacent

edges of v , respectively, (otherwise, they would meet again.) which means that they lose their mutual visibility. The only possible way to clear $C_{cw}(v)$ is to let one guard, such as R , go around reflex vertex u counterclockwise. Then the two guards attempt to be present in component $C_{cw}(v)$.

In order to clear $C_{cw}(u)$, the two guards are to be present in it together. After $C_{cw}(u)$ is cleared out, again the two guards cannot both walk counterclockwise, due to reflex vertex u and the same reasoning as above. So it must be true that one guard, say L , walks clockwise, while R walks counterclockwise. In order to clear $C_{ccw}(v)$, the two guards are present together in it at some time instant. After that, either they can meet within the component or they can walk outside and meet at some point on $\partial P[B(v), u_r]$. Note that the two guards cannot meet beyond u clockwise, due to the same reasoning as above. \square

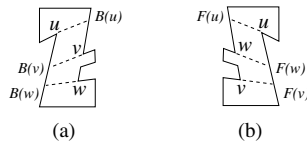


Fig. 3. (a) A 2DPcw; (b) A 2DPccw

Figs. 3 (a) and (b), respectively, show two DPcws and two DPccws coupled together. A *DPcw couple* (2DPcw, for short) consists of three reflex vertices, u, v, w such that $\{u, v\}$ and $\{u, w\}$ each forms a DPcw, as shown in Fig. 3 (a). A *DPccw couple* (2DPccw, for short) is defined analogously and is shown in Fig. 3 (b).

Lemma 3. *For a walkable polygon containing a 2DPcw formed by reflex vertices u, v, w , in Fig. 3 (a), any point on $\partial P[v_l, w_r]$ can be used as neither a WSP nor a WEP. An analogous property holds for a 2DPccw in Fig. 3 (b).* \square

In Figs. 3 (a) and (b), each point on $\partial P[v_l, w_r]$ is said to be *covered* by a 2DPcw or a 2DPccw, respectively.

3 Characterization of Walkable Polygons

We discuss our characterization of walkable polygons. In order to make our presentation clear we categorize polygons into two groups, those that do not contain any DPcws and DPccws (Case I) and those that do (Case II).

3.1 Complex Patterns

Patterns \aleph and Reverse- \aleph . Consider two DPcws, one formed by $\{u, v\}$ and the other formed by $\{w, x\}$, where $u \in C_{cw}(x), w \in C_{cw}(u), v \in C_{cw}(w),$ and $x \in C_{cw}(v)$, as shown in Fig. 4 (a). We call it *pattern \aleph* ³. Symmetrically, as shown

³ It is pronounced “aleph”.

in Fig. 4 (b), *pattern reverse- \aleph* is composed of two DPccws, one due to $\{u, v\}$ and the other due to $\{w, x\}$, such that $u \in C_{ccw}(x)$, $x \in C_{ccw}(v)$, $v \in C_{ccw}(w)$, and $w \in C_{ccw}(u)$. Note that v and x can be the same reflex vertices and u and w can be the same in both patterns.

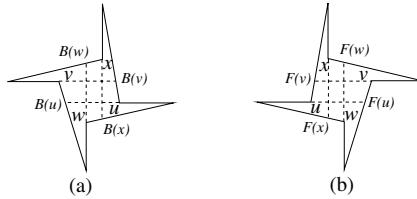


Fig. 4. (a) Pattern \aleph ; (b) Pattern reverse- \aleph

Lemma 4. *In a walkable polygon that contains pattern \aleph in Fig. 4 (a), if we select a WSP (WEP, resp.) on $\partial P[u_l, w_r]$ ($\partial P[x_l, u_r]$, resp.) for a walk, we can only select a WEP (WSP, resp.) on $\partial P[v_l, x_r]$ ($\partial P[w_l, v_r]$, resp.) An analogous property holds for pattern reverse- \aleph in Fig. 4 (b). \square*

Patterns Σ , Reverse- Σ and \bowtie . We next consider together a 2DPcw and a 2DPccw, as shown in Fig. 5. In Fig. 5 (a), $\{u, v\}$ forms a DPcw and $\{x, w\}$ forms a DPccw, while $\{u, x\}$ forms a trap. We call it *pattern Σ* . Similarly in Fig. 5 (b), $\{u, v\}$ forms a DPccw and $\{x, w\}$ forms a DPcw, while $\{u, x\}$ forms a trap. We say that they form *pattern reverse- Σ* .

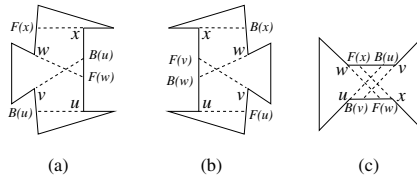


Fig. 5. (a) Pattern Σ ; (b) Pattern reverse- Σ ; (c) Pattern \bowtie

Lemma 5. *For patten Σ in Fig. 5 (a), any point on $\partial P[v_l, w_r]$ can be used as neither a WSP nor a WEP. Analogously, in patten reverse- Σ in Fig. 5 (b), any point on $\partial P[w_l, v_r]$ can be used as neither a WSP nor a WEP. \square*

Fig. 5 (c) shows pattern \bowtie ⁴, where one DPcw, formed by $\{u, v\}$, and one DPccw, formed by $\{w, x\}$, are together such that $u \in C_{ccw}(w)$, $w \in C_{cw}(u)$, $v \in C_{ccw}(x)$, and $x \in C_{cw}(v)$. Pattern \bowtie has the following property.

Lemma 6. *For pattern \bowtie shown in Fig. 5 (c), if we select a WSP (WEP, resp.) on $\partial P[v_l, x_r]$ for walk, then we can only select a WEP (WSP, resp.) on $\partial P[u_l, w_r]$. \square*

⁴ It is pronounced as “bow tie”.

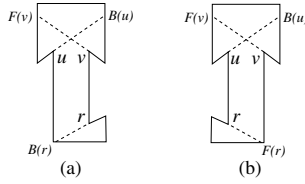


Fig. 6. (a) Pattern τ ; (b) Pattern $\text{reverse-}\tau$

Patterns τ and $\text{Reverse-}\tau$. Pattern τ couples a DPcw, formed by $\{u, r\}$, and a trap, formed by $\{u, v\}$, as shown in Fig. 6 (a). Similarly, pattern $\text{reverse-}\tau$ couples a DPccw and a trap together, as shown in Fig. 6 (b).

3.2 Case I

Consider a polygon that does not contain any DPcws and DPccws, i.e., no patterns in Figs. 3, 5, 4, and 6 are present. The only possible ones are those due to traps in Fig. 1. For example, the polygon in Fig. 7 has three traps, namely by $\{9, 3\}$, $\{3, 6\}$, and $\{6, 9\}$. Due to them, there are contradictory requirements on the WSP and WEP of a walk. Thus the polygon is not walkable.

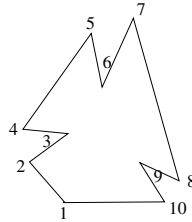


Fig. 7. A polygon that has its entire boundary covered by traps

Theorem 1. *Suppose that a polygon does not contain any DPcws and DPccws. It is walkable by two guards if and only if there is at least boundary point that is not covered by any traps.*

Proof. We omit the proof for the sake of space in this extended abstract. □

3.3 Case II

In Case II, a polygon contains DPcws and/or DPccws.

Lemma 7. *In polygon P containing pattern \aleph in Fig. 4 (a), if one of the portions $\partial P[u_l, v_r]$, $\partial P[w_l, x_r]$, $\partial P[v_l, u_r]$ and $\partial P[x_l, w_r]$ is covered, P is not walkable. An analogous property holds for pattern $\text{reverse-}\aleph$ in Fig. 4 (b). □*

We call pattern \aleph and pattern $\text{reverse-}\aleph$ that satisfy the conditions in Lemma 7 *forbidden pattern \aleph* and *forbidden pattern $\text{reverse-}\aleph$* , respectively.

Lemma 8. *In polygon P containing pattern Σ in Fig. 5 (a), if one of $\partial P[w_l, x_r]$ and $\partial P[u_l, v_r]$ is covered, P is not walkable by two guards. For pattern reverse- Σ in Fig. 5 (b), if one of $\partial P[x_l, w_r]$ and $\partial P[v_l, u_r]$ is covered, P is not walkable. For pattern \bowtie in Fig. 5 (c), if one of $\partial P[u_l, w_r]$ and $\partial P[v_l, x_r]$ is covered, P is not walkable. \square*

We call patterns in Fig. 5 that satisfy the conditions in Lemma 8 *forbidden pattern Σ* , *forbidden pattern reverse- Σ* , and *forbidden pattern \bowtie* , respectively.

Lemma 9. *If polygon P contains pattern τ , it is not walkable by two guards. An analogous property holds for a polygon containing pattern reverse- τ . \square*

Thus pattern τ and pattern reverse- τ are themselves both forbidden patterns.

Theorem 2. *Suppose that a polygon contains DPcws and/or DPccws. It is walkable by two guards if and only if it does not contain any of the above forbidden patterns.*

Proof. The necessity is due to Lemmas 7, 8, and 9. The sufficiency is similar to the proof of Theorem 1 by focusing on some portions of the boundary. For instance, if the polygon contains pattern \aleph , shown in Fig. 4 (a), because it is not forbidden, there must be a boundary point that is not covered. Suppose that it is on $\partial P[v_l, x_r]$. Then accordingly, there exists a point on the boundary portion $\partial P[u_l, w_r]$ that is not covered. If a selected WSP is on one of them, we can only select a WEP on the other. The different situations we need to discuss are similar to those in Theorem 1 and thus omitted in this extended abstract. \square

4 A Comparison between 1-Searcher and Two Guards

A 1-searcher holds a flashlight that emits a light beam. The vision of the searcher is restricted to the light beam [12]. The beam head hits the polygon boundary.

If a polygon is walkable by two guards, at any time $t \in [0, T]$, we can replace one guard by the beam head and the other by the searcher. At time T , the searcher and the beam head meet again, finishing searching the polygon. Therefore, the polygon is also searchable by a 1-searcher.

However, the reverse is not true, as shown by a polygon in Fig. 8 (adapted from [19]). This is due to the fact that in the 1-searcher search, we can make

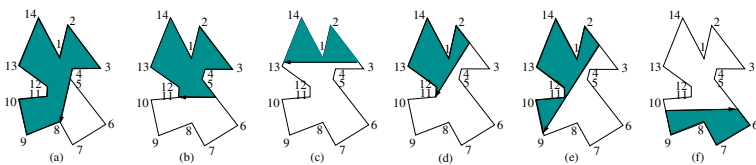


Fig. 8. A polygon that is searchable by a 1-searcher but not walkable by two guards

use of *recontamination*. In the figure, the white area is clear while the shaded is contaminated.

In the figure, the 1-searcher starts at vertex 6 and points its beam at vertex 6. It walks counterclockwise towards vertex 5 as it rotates its beam to the right until vertex 8. After that, Figs. 8(b), (c) and (d) show the following situations. In Fig. 8(e), the beam head jumps from reflex vertex 11 to a point close to vertex 9. The previous cleared area around vertex 10 now becomes contaminated. We call this *recontamination*. It is necessary for the 1-searcher in this polygon to make use of it. Now the 1-searcher can proceed to finish searching the polygon.

On the other hand, $\{4, 11\}$ forms a DPccw (a degenerate pattern reverse- \aleph). Due to the two traps formed by $\{11, 1\}$ and $\{1, 5\}$, any point on $\partial P[11_l, 1_r] \cup \partial P[1_l, 5_r]$ is covered. Therefore, according to Theorem 2, the polygon is not walkable.

5 Conclusion

Our characterization of walkable polygons by two guards completes the strain of the two-guard walk problem, i.e., the two-guard street problem, the two-guard room walk problem, the two-guard general polygon walk problem. Though simple, our characterization involves non-trivial geometric structures in a polygon. It would be an interesting problem to see whether the walkability of a polygon by two guards can be tested in linear time in terms of the number of vertices of the polygon. Some work has been done for the linear-time searchability testing of a polygon by a 1-searcher [19]. In addition, the generation of “optimal walk schedules” to walk a polygon by two guards is also an interesting problem. Our future attempts will focus on these directions.

References

1. Icking, C., Klein, R.: The two guards problem. *Int'l. J. of Computational Geometry and Applications* 2(3), 257–285 (1992)
2. Heffernan, P.: An optimal algorithm for the two-guard problem. *Int'l. J. of Computational Geometry and Applications* 6, 15–44 (1996)
3. Crass, D., Suzuki, I., Yamashita, M.: Searching for a mobile intruder in a corridor: the open edge variant of the polygon search problem. *Int'l. J. of Computational Geometry and Applications* 5(4), 397–412 (1995)
4. Lee, J., Park, S.M., Chwa, K.Y.: Searching a polygonal room with one door by a 1-searcher. *Int'l. J. of Computational Geometry and Applications* 10(2), 201–220 (2000)
5. Lee, J.H., Shin, S.Y., Chwa, K.Y.: Visibility-based pursuit-evasions in a polygonal room with a door. In: *Proc. ACM Symp. on Computational Geometry*, pp. 281–290 (1999)
6. Park, S.M., Lee, J.H., Chwa, K.Y.: Characterization of rooms searchable by two guards. In: *Proc. Int'l. Symp. on Algorithms and Computation*, pp. 515–526 (2000)

7. Park, S.M., Lee, J.H., Chwa, K.Y.: Searching a room by two guards. *Int'l. J. of Computational Geometry and Applications* 12(4), 339–352 (2002)
8. Tan, X.: Efficient algorithms for searching a polygonal room with a door. In: Akiyama, J., Kano, M., Urabe, M. (eds.) *JCDCG 2000*. LNCS, vol. 2098, pp. 339–350. Springer, Heidelberg (2001)
9. Bhattacharya, B., Zhang, J.Z., Shi, Q.S., Kameda, T.: An optimal solution to room search problem. In: *Proc. 18th Canadian Conf. on Computational Geometry*, August 2006, pp. 55–58 (2006)
10. Zhang, J.Z., Kameda, T.: Where to build a door. In: *Proc. IEEE/RSJ Int'l. Conf. on Intelligent Robots and Systems*, October 2006, pp. 4084–4090 (2006)
11. Zhang, J.Z., Kameda, T.: A linear-time algorithm for finding all door locations that make a room searchable (extended abstract). In: Agrawal, M., Du, D.-Z., Duan, Z., Li, A. (eds.) *TAMC 2008*. LNCS, vol. 4978, pp. 502–513. Springer, Heidelberg (2008)
12. Suzuki, I., Yamashita, M.: Searching for a mobile intruder in a polygonal region. *SIAM J. on Computing* 21(5), 863–888 (1992)
13. Tan, X.: Searching a simple polygon by a k-searcher. In: *Proc. of Int'l. Symp. on Algorithms and Computation* 2000, pp. 503–514 (2000)
14. Tan, X.: A Characterization of Polygonal Regions Searchable from the Boundary. In: Akiyama, J., Baskoro, E.T., Kano, M. (eds.) *IJCCGGT 2003*. LNCS, vol. 3330, pp. 200–215. Springer, Heidelberg (2005)
15. Park, S.M., Lee, J.H., Chwa, K.Y.: A characterization of the class of polygons searchable by a 1-searcher. Technical Report CS/TR-2000-160, Korea Advanced Institute of Science and Technology (December 2000)
16. Kameda, T., Zhang, J.Z., Yamashita, M.: Simple characterization of polygons searchable by 1-searcher. In: *Proc. the 18th Canadian Conf. on Computational Geometry*, August 2006, pp. 113–116 (2006)
17. Zhang, J.Z., Burnett, B.: Yet another simple characterization of searchable polygons by 1-searcher. In: *Proc. IEEE Int'l. Conf. on Robotics and Biomimetics*, December 2006, pp. 1244–1249 (2006)
18. LaValle, S.M., Simov, B., Slutzki, G.: An algorithm for searching a polygonal region with a flashlight. *Int'l. J. of Computational Geometry and Applications* 12(1-2), 87–113 (2002)
19. Bhattacharya, B., Zhang, J.Z., Kameda, T.: Exploring polygonal area by robot: Searching testing. In: *Proc. Int'l. Conf. on Robotics and Automation* (May 2009) (to appear)

Approximation and Hardness Results for Label Cut and Related Problems

Peng Zhang^{1,*}, Jin-Yi Cai^{2,**}, Linqing Tang^{3,***}, and Wenbo Zhao⁴

¹ School of Computer Science and Technology, Shandong University,
Ji'nan 250101, China
algzhang@sdu.edu.cn

² Computer Sciences Department, University of Wisconsin, Madison, WI 53706, USA
jyc@cs.wisc.edu

³ State Key Laboratory of Computer Science, Institute of Software,
Chinese Academy of Sciences, Beijing 100080, China
linqing@ios.ac.cn

⁴ Dept. of Computer Science and Engineering, University of California, San Diego,
La Jolla, CA 92093, USA
w3zhao@ucsd.edu

Abstract. We investigate a natural combinatorial optimization problem called the *Label Cut* problem. Given an input graph G with a source s and a sink t , the edges of G are classified into different categories, represented by a set of *labels*. The labels may also have weights. We want to pick a subset of labels of minimum cardinality (or minimum total weight), such that the removal of all edges with these labels disconnects s and t . We give the first non-trivial approximation and hardness results for the Label Cut problem. Firstly, we present an $O(\sqrt{m})$ -approximation algorithm for the Label Cut problem, where m is the number of edges in the input graph. Secondly, we show that it is **NP**-hard to approximate Label Cut within $2^{\log^{1-1/\log \log^c n} n}$ for any constant $c < 1/2$, where n is the input length of the problem. Thirdly, our techniques can be applied to other previously considered optimization problems. In particular we show that the Minimum Label Path problem has the same approximation hardness as that of Label Cut, simultaneously improving and unifying two known hardness results for this problem which were previously the best (but incomparable due to different complexity assumptions).

1 Introduction

In many graph optimization problems, it is natural to associate edges with *labels* (or *colors*) which partition the set of edges into categories. There are several classical optimization problems considered under this model, such as the Minimum

* Supported by NSFC 60325206 and China Postdoctoral Science Foundation No. 20080441144.

** Supported by NSF CCF-0511679.

*** Supported by NSFC 60325206 and NSFC 60310213.

Label Spanning Tree problem [2] and the Minimum Label s - t Path problem (Label Path, for short) [7]. In this paper we consider the Minimum Label s - t Cut problem (Label Cut, for short), in which we are asked to pick labels at minimum total cost to disconnect a source s and a sink t by removing edges from the graph whose labels are picked. The Label Cut problem is a natural generalization of the classical Minimum s - t Cut problem, in the sense that every edge in the latter has a unique label. This problem is also a generalization of the Minimum Set Cover or Minimum Hitting Set problems. In fact it can be considered as a version of the Minimum Hitting Set problem where the collection of sets to be “hit” is implicitly represented (and could be exponentially many).

This problem is sufficiently natural that it can appear in many contexts. We came across this problem from the work of Jha, Sheyner and Wing [8], and of Sheyner, Haines, Jha, Lippmann, and Wing [11,10] in computer security, in particular on intrusion detection and on generation and analysis of *Attack Graphs*[1]. In this application, an attack graph G has nodes representing various states, and directed edges representing state transitions and are labeled by possible “atomic attacks”. A pair of special nodes s and t are also given representing the initial state and the success state (for the intruder). To disable an “atomic attack” incurs some cost (a unit or a weighted cost). Then the computational task is to find a subset of “atomic attacks” of minimum cardinality (or of minimum total weight), such that the removal of all edges labeled by these “atomic attacks” disconnects s and t . This is precisely the Minimum Label Cut problem.

Formally, an instance of the Minimum Label Cut problem consists of a graph $G = (V, E)$ (directed or undirected) with one source node $s \in V$ and one sink node $t \in V$, and a label (or color) set $L = \{1, 2, \dots, q\}$. Each edge $e \in E$ has a unique label $l(e) \in L$, but different edges may have the same label. Each label $l \in L$ has a nonnegative weight $w(l)$. A subset $L' \subseteq L$ of labels is called a *label cut* if the removal of all edges of labels in L' disconnects s and t . The problem asks to find a label cut L' such that the total weight of L' is minimized.

1.1 Related Works

In [8] it was observed that Label Cut is **NP**-hard by reducing Minimum Hitting Set to it. There is a simple duality between Minimum Hitting Set and Minimum Set Cover. It is well known that Minimum Set Cover has a greedy polynomial time $(1 + \ln |U|)$ -approximation algorithm, where $|U|$ is the size of the underlying set. The same algorithm can be translated to Hitting Set by duality. In [8] the authors express the Label Cut problem as an *implicit* Minimum Hitting Set problem, and then translate the Set Cover algorithm to get an approximation algorithm. This implicit Minimum Hitting Set problem is as follows: Let $\mathcal{D} = \{L' \mid L' \text{ is the set of labels appearing on a path from } s \text{ and } t\}$. This is the set to hit. A set of labels is a label cut iff it intersects every $L' \in \mathcal{D}$. The approximation algorithm for Set Cover translates to an approximation algorithm for Label Cut with approximation guarantee of $1 + \ln |\mathcal{D}|$.

¹ We are indebted to Jeannette Wing who gave an overview talk on their interesting work at Tsinghua University.

However, as observed by Jha, Sheyner and Wing [8], $|\mathcal{D}|$ is typically exponentially larger than the input size (number of edges in G). Since the removal of *all* edges is certainly a feasible solution to Label Cut, the worst-case approximation ratio $1 + \ln |\mathcal{D}|$ is useless. There is an additional problem. This concerns the need to compute in polynomial time the next “greedy” step in the approximation algorithm for Set Cover. For implicit Minimum Hitting Set, we would need to find the next label which “hits” the most remaining paths from s to t , and it is not known how to do this in general. The authors in [8] consider special cases where this is feasible.

We mention another problem, dual to Label Cut, called the Label Path problem. Given an edge-labeled graph, the Label Path problem finds an s - t path with minimum total number of labels on the path. Label Path is **NP**-hard by a reduction from 3SAT. Through an approximation preserving reduction from the Red-Blue Set Cover problem [3], Wirth [12, Theorem 2.16] proved that unless $\mathbf{NP} \subseteq \mathbf{DTIME}(n^{\text{polylog}(n)})$ Label Path can not be approximated within ratio $O(2^{\log^{1-\epsilon} n})$. Recently, Hassin, Monnot and Segev [7] gave an $O(\sqrt{n})$ -approximation algorithm for the weighted Label Path problem, and an approximation hardness result $O(\log^k n)$ (for any fixed $k \geq 1$) for the problem assuming $\mathbf{P} \neq \mathbf{NP}$. Notice that the two hardness results for Label Path are incomparable.

1.2 Our Results

In this paper, we give the first non-trivial approximation algorithm as well as prove hardness results for the Label Cut problem. Firstly, we give a polynomial time $O(\sqrt{m})$ -approximation algorithm, where m is the number of edges in the input graph. The idea of the algorithm is that we first make the s - t cut of the graph small by picking labels at *appropriately inexpensive* cost, and when the s - t cut of the graph is sufficiently small (which will definitely be true when the graph is sufficiently sparse) we directly use all the labels in the minimum s - t cut as the label cut in the resulting graph. The solution to the problem is then the set of labels picked in the first stage and the labels in the minimum s - t cut computed in the second stage. Our idea is inspired by the work of [7]. We can extend the algorithm to deal with related cut problems in edge-classified graphs, such as Label Multi-cut, Label Multiway Cut, and Label k -Cut (we call them generally the Label γ -Cut problem), thus obtaining $O((m\rho)^{1/2})$ -approximation for these problems, where ρ is the known approximation ratio for γ -Cut.

Secondly, we show that unless $\mathbf{P} = \mathbf{NP}$, Label Cut can not be approximated within $2^{\log^{1-1/\log \log^c n} n}$ for any constant $c < 1/2$, where n is the input length of the problem. We prove this by a gap-preserving reduction from the Label Cover problem, for which Dinur and Safra [6] proved such a hardness result. Our method of the reduction also gives the same approximation hardness for the Label Path problem as that of Label Cut, thus simultaneously improving and unifying two known hardness results [7,12] for this problem which were previously the best. The approximation hardness proof for Label Path is similar to that for Label Cut, and will be given in the full version of the paper.

The proof of hardness for Label Cut and Label Path relies on two ingredients. Firstly we show that the approximation hardness $2^{\log^{1-1/\log \log^c n} n}$ (under complexity assumption $\mathbf{P} \neq \mathbf{NP}$) is preserved even for a special type of instances of Label Cover, in which the two parts of an input bipartite graph have the same number of vertices. Secondly, we show that the mapping relation between labels in the Label Cover problem can be represented by some sub-structures in the Label Cut (resp. Label Path) problems, and thus the cut requirement in Label Cut (resp. the connectivity requirement in Label Path) is equivalent to the edge-covering requirement in Label Cover.

We have just learned that Coudert et al. [4] [Theorem 5.1] have independently proved a slightly weaker hardness result. Their result is that the Label Path and Label Cut problems are \mathbf{NP} -hard to approximate within a factor of the same expression as our lower bound, except the input length n in our bound is replaced by the quantity $|L|^{1/2}$. When the label set L has essentially the same cardinality as the input length n (or the number of edges) the bound of Coudert et al. [4] is only slightly weaker but very close to the same as ours. However this may not be the case in general since essentially L is a partition of the edge set of the input graph, and it is especially so in our application with attack graphs, where $|L|$ is typically much smaller. In such cases, the lower bound in this paper is stronger.

2 An $O(\sqrt{m})$ -Approximation Algorithm for Label Cut

Our plan is to successively remove edge sets from the graph which are relatively large, yet defined by small label sets. When the s - t cut of the graph becomes sufficiently small (which is implied when the graph becomes sufficiently sparse) we can directly remove the minimum s - t cut and its corresponding label set. To do this we make use of the Budgeted Maximum Coverage problem (BMCP, for short). We are given a ground set U , a collection \mathcal{S} of subsets of U , and a budget B . Each subset $S_i \in \mathcal{S}$ has a nonnegative weight. The problem is to find a sub-collection $\mathcal{S}' \subseteq \mathcal{S}$ such that the total weight of \mathcal{S}' is at most B and the number of elements in U covered by \mathcal{S}' is maximized. This problem can be approximated within a factor of $1 - \frac{1}{e}$ [9].

Now we give a detailed outline of our approximation algorithm for Label Cut. Let $m = |E|$ and $w(L') = \sum_{l \in L'} w(l)$ for any label subset $L' \subseteq L$. Suppose $L^* \subseteq L$ is an optimal solution to Label Cut. We start with an estimate Δ on the optimum $\text{OPT} = w(L^*)$ such that $\text{OPT} \leq \Delta < 2 \cdot \text{OPT}$. We try a sequence of Δ as guesses (see Theorem [1]). Suppose we have a correct Δ . Starting with $H_0 = G$, we will perform a sequence of edge removal operations. This creates graphs $H_0, H_1, \dots, H_{m'}$, where H_{i+1} is obtained from H_i by the removal of some subset of edges. Each step i also produces a subset of labels L_i . The final output is the union of all these L_i .

At step i , we are given H_i . Keeping only edges in H_i with label weight $> \Delta$ (and delete others) defines a subgraph H'_i . Suppose that G is undirected, then we find connected components of H'_i . For directed G we find strongly connected components of H'_i . Then we merge the nodes from H_i within each (strongly)

connected component defined above to obtain H_i^* . Edges within a component disappear. Other edges remain; in particular between the merged nodes there may be multiple edges in H_i^* . All edges retain their labels, and all label weights in H_i^* are $\leq \Delta$. If $\text{OPT} \leq \Delta$, then vertices s and t remain distinct in H_i^* ; otherwise an s - t path using only edges of weight $> \Delta$ exists which has no intersection with the cut of total weight $\text{OPT} \leq \Delta$.

Now we use a Maxflow-Mincut algorithm to find the minimum cut size $c_i^* = |\text{mincut}_{H_i^*}(s, t)|$ of H_i^* . Here each edge in H_i^* counts as one, and multiple edges count with multiplicity. If $c_i^* \leq m^{1/2}$, then we find a minimum cut of H_i^* , and use all labels on this cut. This is our L_i , and with this we terminate with $m' = i$. Note that all edges of H_i of labels in L_i also form a cut in H_i . Now suppose $c_i^* > m^{1/2}$. We consider the following instance of BMCP where the ground set is the set of all edges $E(H_i)$ of the graph H_i , and the collection of subsets \mathcal{S} in BMCP consists of all $S_l = \{e \in E(H_i) \mid l(e) = l\}$ for $l \in L$, and where the weight of S_l is $w(l)$. We apply the approximation algorithm in [9] on this instance for budget Δ to get a sub-collection $\mathcal{S}' \subseteq \mathcal{S}$. Then we use all labels coming from \mathcal{S}' as our L_i in this case. This gives the approximation algorithm for Label Cut, as presented by Algorithm \mathcal{A} below. For any subset $E' \subseteq E$, let $L(E')$ denote the set of all labels in E' .

Algorithm \mathcal{A}

1. **let** $i \leftarrow 0$, $H_0 \leftarrow G$. Define H'_0 and H_0^* as described above.
2. **while** $|\text{mincut}_{H_i^*}(s, t)| > m^{1/2}$ **do**
 - a. Construct a BMCP instance I' as follows: the ground set $U = E(H_i)$, the collection \mathcal{S} consists of $S_l = \{e \in E(H_i) \mid l(e) = l\}$ with weight $w(l)$ for every $l \in L$, and the budget $B = \Delta$.
 - b. Approximately solve I' and let E_i be the set of covered edges.
 - c. **let** $L_i \leftarrow L(E_i)$, $H_{i+1} \leftarrow H_i \setminus E_i$. Define H'_{i+1} and H_{i+1}^* as described above.
 - d. **let** $i \leftarrow i + 1$.
3. **let** $L_i \leftarrow L(\text{mincut}_{H_i^*}(s, t))$, **return** $L' = \bigcup_i L_i$.

Let m' be the number of iterations of step [2], i.e., m' equals the value of i when step [2] terminates.

Lemma 1. *For Algorithm \mathcal{A} , if $\text{OPT} \leq \Delta$, then $m' < 2m^{1/2}$.*

Proof. Suppose that an optimal solution to the Label Cut instance I is $L^* \subseteq L$ with total weight $\text{OPT} = w(L^*) \leq \Delta$, and that $E^* \subseteq E(G)$ is the set of edges corresponding to the label subset L^* . In each iteration of step [2], $E^* \cap E(H_i^*)$ is a feasible solution to the BMCP instance I' , since the total weight of labels in $E^* \cap E(H_i^*)$ is at most $w(L^*)$, which, in turn, is at most Δ .

A crucial observation is that $E^* \cap E(H_i^*) \subseteq E^* \cap E(H_i)$ is also an s - t cut in H_i^* . This is because any s - t path in H_i^* can be extended to an s - t path in H_i by appending only edges of label weight $> \Delta$. The path in H_i must intersect $E^* \cap E(H_i)$, but all edges in E^* have label weight $\leq \Delta$, therefore the s - t path in H_i^* must intersect $E^* \cap E(H_i^*)$. So, we know that $|E^* \cap E(H_i^*)| \geq |\text{mincut}_{H_i^*}(s, t)| > m^{1/2}$.

By the work of [9], we know that E_i from 2.b. covers at least $(1 - \frac{1}{e})\text{OPT}_{\text{BMCP}}(I')$ edges, where $\text{OPT}_{\text{BMCP}}(I')$ is the optimum of instance I' . So in each iteration of step [2] at least $(1 - \frac{1}{e})\text{OPT}_{\text{BMCP}}(I') \geq (1 - \frac{1}{e})|E^* \cap E(H_i^*)| > \frac{1}{2}m^{1/2}$ edges are removed from H_i . Since there are at most m edges in total in graph H_0 , it follows that the number of iterations $m' < 2m^{1/2}$. □

Theorem 1. *The Label Cut problem can be approximated within a factor of $O(m^{1/2})$ in polynomial time, where m is the number of edges in the input graph of the problem.*

Proof. Let w_{\min} denote the minimum (nonzero) label weight. Then we know $w_{\min} \leq \text{OPT} \leq w(L)$. We try Algorithm \mathcal{A} with $\Delta = 2^k \cdot w_{\min}$ for every $k = 0, \dots, \lceil \log_2 \frac{w(L)}{w_{\min}} \rceil$, where Δ is used as an estimate of OPT . Let k be such that $2^{k-1} \cdot w_{\min} < \text{OPT} \leq 2^k \cdot w_{\min}$. Then for the guess $\Delta = 2^k \cdot w_{\min}$, we have $\text{OPT} \leq \Delta < 2 \cdot \text{OPT}$.

With this Δ , we know that the number of iterations $m' < 2m^{1/2}$ by Lemma [1]. Moreover, when step [2] terminates, $|\text{mincut}_{H_{m'}^*}(s, t)|$ is at most $m^{1/2}$, implying that $w(L_{m'}) \leq |\text{mincut}_{H_{m'}^*}(s, t)| \cdot \max\{w(l) : l \in L(E(H_{m'}^*))\} \leq m^{1/2}\Delta$. Thus the total weight of the output L' is at most $m'\Delta + w(L_{m'}) \leq 2m^{1/2} \cdot \Delta + m^{1/2} \cdot \Delta = O(m^{1/2}) \cdot \text{OPT}$. We note that, from the label set $L_{m'}$ derived from $\text{mincut}_{H_{m'}^*}(s, t)$, all edges of labels from $L_{m'}$ in $H_{m'}$ do constitute an s - t cut in $H_{m'}$, and therefore the output L' is an s - t cut in G .

Finally, for every k , we run Algorithm \mathcal{A} . (We never run beyond $m' \geq 2m^{1/2}$, and verify each time s and t remain distinct in the merging operations; if these are unsatisfied then we go to the next Δ .) Among all tries with different Δ we pick the label subset with minimum total weight as our final solution. □

3 Approximation Hardness for Label Cut

In this section we prove that it is **NP**-hard to approximate Label Cut within $2^{\log^{1-o(1)} n}$ via a reduction from the Minimum Label Cover problem (Label Cover, for short). The current best known approximation hardness result for Label Cover is due to Dinur and Safra [6], which states that the problem is **NP**-hard to approximate within $2^{\log^{1-1/\log \log^c n} n}$ for any constant $c < 1/2$.

In the Minimum Label Cover problem, we are given a bipartite graph $G = (U, V, E)$ where $E \subseteq U \times V$, two sets of possible labels B_1 and B_2 for vertices in U and V respectively, and a relation $\Pi \subseteq E \times B_1 \times B_2$ that consists of admissible pairs of labels for each edge $e \in E$. A *labeling* is a pair of functions $(\mathcal{P}_1, \mathcal{P}_2)$ that assigns a non-empty set of labels to every vertex in $U \cup V$, where $\mathcal{P}_1 : U \rightarrow 2^{B_1}$ and $\mathcal{P}_2 : V \rightarrow 2^{B_2}$. It is said to *cover* an edge $e = (u, v)$ (where $u \in U$ and $v \in V$) if for every label $b \in \mathcal{P}_2(v)$ there is some label $a \in \mathcal{P}_1(u)$ such that $(e, a, b) \in \Pi$. The l_1 -cost of a labeling is the l_1 norm of the vector $(|\mathcal{P}_1(u_1)|, \dots, |\mathcal{P}_1(u_{|U|})|)$, that is, the total number $\sum_{u \in U} |\mathcal{P}_1(u)|$ of labels assigned to vertices in U by function \mathcal{P}_1 counted with multiplicities. A *total-cover* (a solution) of G is a labeling that covers all the edges in G . The problem is to find a total-cover with

minimal l_1 -cost. We always implicitly assume that the only instances we shall consider for Minimum Label Cover are such that a total-cover exists (see [1], p.403, line -8).

Dinur and Safra [6] proved the approximation hardness result $2^{\log^{1-o(1)} n}$ for Label Cover via a reduction from the following version of the PCP Theorem.

Theorem 2 ([5]). *Let $\Psi = \{\psi_1, \dots, \psi_n\}$ be a system of local-tests over variables $X = \{x_1, \dots, x_{n'}\}$ such that each local-test depends on $D = \log \log^c n$ variables (for any constant $c < 1/2$), and each variable ranges over a field \mathcal{F} with $|\mathcal{F}| = O(2^{\log^{1-1/O(D)} n})$, where n stands for the number of tests in Ψ and n' stands for the number of variables in X . It is NP-hard to distinguish between the following two cases: Yes - there is an assignment to the variables such that all ψ_1, \dots, ψ_n are satisfied, and No - no assignment can satisfy more than $\frac{2}{|\mathcal{F}|}$ fraction of the ψ_i 's.*

Define $g_c(n) = 2^{\log^{1-1/\log \log^c n} n}$. Dinur and Safra proved that

Theorem 3 ([6]). *Let Ψ be a local-test system defined in Theorem 2. Then an instance $I = ((U, V, E), B_1, B_2, \Pi)$ of Label Cover can be constructed from Ψ in polynomial time such that for any constant $c < 1/2$, (soundness) Ψ is satisfied $\implies \text{OPT}(I) = |U|$, and (completeness) no assignment can satisfy more than $\frac{2}{|\mathcal{F}|}$ fraction of $\Psi \implies \text{OPT}(I) > g|U|$, where $g = g_c(|I|)$ and $\text{OPT}(I)$ denotes the optimum of Label Cover on the instance I .*

We shall prove the approximation hardness of Label Cut via a gap-preserving reduction from Label Cover. For succinctness, denote by $E(v) \subseteq E$ all edges that are incident to v , and by $N(v) \subseteq U$ all endpoints of edges in $E(v)$ that lie in U , i.e., $N(v)$ is the neighborhood of vertex v . Without loss of generality we assume $|N(v)| \geq 1$ for all $v \in V$. Similarly every u is in some neighborhood $N(v)$, i.e., $U = \bigcup_{v \in V} N(v)$. First we claim a simple lemma about labels in B_2 .

Lemma 2. *For any vertex $v \in V$ and label $b \in B_2$, if there is a vertex $u \in N(v)$ such that for all $a \in B_1$, $((u, v), a, b) \notin \Pi$, then vertex v can not be assigned with label b by any solution.*

If a label $b \in B_2$ satisfies the condition in Lemma 2, we say this label b is *excluded* (by the vertex v). A label b is *non-excluded* (by the vertex v) iff for all $u \in N(v)$ there exists some label $a \in B_1$, such that $((u, v), a, b) \in \Pi$. Since we only consider instances where a total-cover exists, we must have for every v , it has a non-empty set of non-excluded labels. The following Lemma 3 gives a transformation on the instances of Label Cover, which is useful for the gap-preserving reduction from Label Cover to Label Cut (its proof is omitted).

Lemma 3. *Without loss of generality, we may assume that instances of Label Cover satisfy $|U| = |V|$.*

Lemma 4. *There is a polynomial-time gap-preserving reduction τ from Label Cover to Label Cut that transforms any instance $I = ((U, V, E), B_1, B_2, \Pi)$ of Label Cover to an instance $I' = ((V', E'), s, t, L)$ of Label Cut, such that,*

$$\begin{aligned} \text{OPT}_{\text{COVER}}(I) = |U| &\implies \text{OPT}_{\text{CUT}}(I') \leq |U| + |V|, \\ \text{OPT}_{\text{COVER}}(I) > g|U| &\implies \text{OPT}_{\text{CUT}}(I') > g|U| + |V|, \end{aligned}$$

where $\text{OPT}_{\text{COVER}}()$ (resp. $\text{OPT}_{\text{CUT}}()$) returns the optimum of any instance of Label Cover (resp. Label Cut).

Proof. Given an instance $I = ((U, V, E), B_1, B_2, \Pi)$ of Label Cover, the instance $I' = (G', s, t, L)$ of Label Cut is constructed as follows. For every vertex v and non-excluded label b , there is a gadget $G_{v,b}$, as shown in Figure 1. $G_{v,b}$ consists of a series of parallel chains, each of which corresponds to an edge $(u, v) \in E(v)$. For such an edge (u, v) , suppose that the set of labels mapped to $b \in B_2$ by Π is $\{a_1, a_2, \dots, a_p\} \subseteq B_1$, i.e., this is the subset of labels $a \in B_1$ such that $((u, v), a, b) \in \Pi$. (Since b is a non-excluded label by v , for every neighbor $u \in N(v)$, this number $p \geq 1$. This p depends on v, b and u . Note also that $|N(v)| \geq 1$ as noted earlier.) Then in $G_{v,b}$ the chain corresponding to edge (u, v) contains p consecutive diamonds, with the i th diamond ($1 \leq i \leq p$) having label (u, a_i) for its two top edges, and label (v, b) for its two bottom edges (see Figure 1). All other chains are constructed in the same manner, one chain for each $u \in N(v)$. The dashed lines (not belonging to the graph) at the two ends of $G_{v,b}$ mean that the ends of each chain are merged.

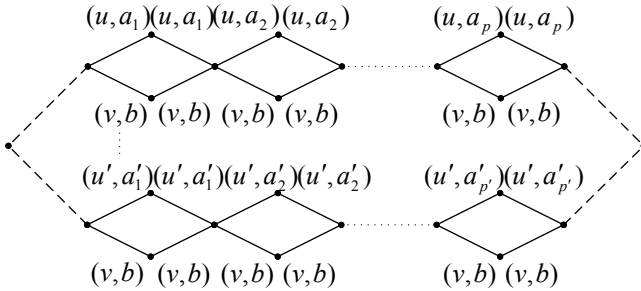


Fig. 1. Gadget $G_{v,b}$ for vertex $v \in V$ and label $b \in B_2$

Next, a gadget G_v for every vertex $v \in V$ is constructed by linking several gadgets $G_{v,b}$ together, where each gadget $G_{v,b}$ corresponds to a non-excluded label b by v (see Figure 2). Note that only $G_{v,b}$ for non-excluded labels b for the vertex v appear in G_v . Its number is at least one but $\leq |B_2|$. (In Figure 2 for notational simplicity we assume that for the given example of G_v , all $|B_2|$ labels are non-excluded labels for v and are denoted as $1, 2, \dots, |B_2|$.) Finally, graph $G' = (V', E')$ consists of a series of parallel gadgets G_v for every vertex $v \in V$. All the heads of gadgets G_v are merged into one vertex, that is, the source vertex s , meanwhile all the tails of gadgets G_v are merged into one vertex, that is, the sink vertex t . The label set L contains all labels of the forms (u, a) and (v, b) used in the construction. Every label in L has unit weight. Since each diamond of G' corresponds to a tuple of Π , we know that I' can be constructed in polynomial time.

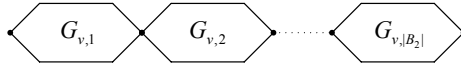


Fig. 2. Gadget G_v for vertex $v \in V$

Suppose $(\mathcal{P}_1^*, \mathcal{P}_2^*)$ is an optimal solution to the Label Cover instance I whose l_1 -cost is $|U|$. For any $v \in V$, since $\mathcal{P}_2^*(v)$ is non-empty, there exists some $b \in \mathcal{P}_2^*(v)$. This b is obviously a non-excluded label, by the definition of a solution in Label Cover. So (v, b) is a label in L . Now we pick the label (v, b) . Next, for all $u \in U$, pick all labels (u, a) for $a \in \mathcal{P}_1^*(u)$. Since the l_1 -cost of $(\mathcal{P}_1^*, \mathcal{P}_2^*)$ is $|U|$, this counts $|\mathcal{P}_1^*(u)|$ for different $u \in U$ with multiplicity, it follows that we have picked a total of $|U| + |V|$ labels from L . We claim that removing all edges with the picked labels disconnects s and t in G' . This is equivalent to disconnecting every G_v . Fix any $v \in V$. We consider the picked label (v, b) , and show that the gadget $G_{v,b}$ is disconnected. This in turn is equivalent to disconnecting every chain in $G_{v,b}$, one chain for each $u \in N(v)$. Consider such a chain defined by a particular $u \in N(v)$. Since all edges labeled (v, b) have been removed, we just need to prove that at least one diamond has been completely removed. Suppose there are p diamonds, where the upper path is labeled by $(u, a_1), \dots, (u, a_p)$, and these a_i are those satisfying $((u, v), a_i, b) \in \Pi$. (see Figure 1). Since $(\mathcal{P}_1^*, \mathcal{P}_2^*)$ is a total-cover, for some $1 \leq i \leq p$, $a_i \in \mathcal{P}_1^*(u)$. But we did pick labels (u, a) , for all $a \in \mathcal{P}_1^*(u)$. It follows that $\text{OPT}_{\text{CUT}}(I') \leq |U| + |V|$.

Now we prove the second part of the lemma by its contrapositive. Suppose the optimum of the Label Cut instance I' is $\leq g|U| + |V|$, and an optimal solution is $L^* \subseteq L$. For clarity let us call a label in L of the form (u, a) a u -label, and a label of the form (v, b) a v -label. Denote by n_{ul} (resp. n_{vl}) the number of u -labels (resp. v -labels) in L^* . Since G' consists of $|V|$ subgraphs G_v in parallel, the removal of edges with labels in L^* must disconnect each G_v . By construction of G_v as a chain of $G_{v,b}$, it must disconnect some $G_{v,b}$, and therefore must pick at least one v -label (v, b) for every $v \in V$. This defines our assignment \mathcal{P}_2 on V , that is, $\mathcal{P}_2(v) = \{b \mid (v, b) \text{ is a picked } v\text{-label in } L^*\}$. Then we know that $n_{\text{vl}} \geq |V|$. Next we define \mathcal{P}_1 on U as follows: $\mathcal{P}_1(u) = \{a \mid (u, a) \text{ is a picked } u\text{-label in } L^*\}$. Since $n_{\text{vl}} \geq |V|$, it follows that $n_{\text{ul}} \leq g|U|$. Since L^* disconnects $G_{v,b}$, for every $u \in N(v)$, L^* must include at least one u -label (u, a) such that $((u, v), a, b) \in \Pi$. In particular $\mathcal{P}_1(u)$ is non-empty. Since $U = \bigcup_{v \in V} N(v)$, this is true for all $u \in U$. This also shows that $(\mathcal{P}_1, \mathcal{P}_2)$ is a solution to the Label Cover instance I . Since $n_{\text{ul}} \leq g|U|$, we know that the l_1 -cost of $(\mathcal{P}_1, \mathcal{P}_2)$ is $\leq g|U|$. This completes the proof of the contrapositive. \square

Theorem 4. *For any constant $c < 1/2$, the Label Cut problem can not be approximated within $2^{\log^{1-1/\log \log^c n} n}$ in polynomial time unless $\mathbf{P} = \mathbf{NP}$, where n is the input length of the problem.*

Proof. Given any Label Cover instance I , first we apply the transformation presented in Lemma 3 to get an instance $\hat{I} = ((U, V, E), B_1, B_2, \Pi)$ of Label Cover such that $|U| = |V|$. Denote by \bar{m} the length of the Label Cover instance

I , and by \bar{n} the length of the Label Cut instance $I' = (G', s, t, L)$ generated by the reduction τ in Lemma 4 on instance \widehat{I} . By Lemma 4, the gap between the two cases in the lemma is at least $g/2$, where $g = g_c(\bar{m})$. By Theorems 2 and 3, it is **NP**-hard to approximate Label Cover within ratio $g_c(\bar{m})$ for any constant $c < 1/2$. Since essentially the edges (together with their labels) in G' reflect the relation Π , we know that $\bar{n} \leq \bar{m}^k$ for some positive constant k , i.e., $\bar{m} \geq \bar{n}^{1/k}$. So we have that it is **NP**-hard to approximate Label Cut (instances generated by the reduction τ of Lemma 4) within ratio $g_c(\bar{m})/2$ for any constant $c < 1/2$. This implies the approximation hardness $g_{c'}(\bar{n})$ of Label Cut for any constant $c' < 1/2$ under the complexity assumption $\mathbf{P} \neq \mathbf{NP}$ (the coefficient $1/2$ and exponential $1/k$ can be absorbed in the exponent of the hardness $g_{c'}(\bar{n})$). \square

References

1. Arora, S., Lund, C.: Hardness of Approximation. In: Hochbaum, D. (ed.) Approximation Algorithms for NP-hard Problems, pp. 399–446. PWS Publishing Company (1997)
2. Broersma, H., Li, X.: Spanning trees with many or few colors in edge-colored graphs. *Discussiones Mathematicae Graph Theory* 17(2), 259–269 (1997)
3. Carr, R., Doddi, S., Konjevod, G., Marathe, M.: On the red-blue set cover problem. In: Proc. of SODA, pp. 345–353 (2000)
4. Coudert, D., Datta, P., Perennes, S., Rivano, H., Voge, M.-E.: Shared risk resource group: complexity and approximability issues. *Parallel Processing Letters* 17(2), 169–184 (2007)
5. Dinur, I., Fischer, E., Kindler, G., Raz, R., Safra, S.: PCP characterizations of NP: towards a polynomially-small error-probability. In: Proc. of STOC, pp. 29–40 (1999)
6. Dinur, I., Safra, S.: On the hardness of approximating Label Cover. *Information Processing Letters* 89(5), 247–254 (2004)
7. Hassin, R., Monnot, J., Segev, D.: Approximation algorithms and hardness results for labeled connectivity problems. *Journal of Combinatorial Optimization* 14(4), 437–453 (2007)
8. Jha, S., Sheyner, O., Wing, J.M.: Two formal analyses of attack graphs. In: Proceedings of the 15th IEEE Computer Security Foundations Workshop, Nova Scotia, Canada, June 2002, pp. 49–63 (2002)
9. Khuller, S., Moss, A., Naor, J.: The budgeted maximum coverage problem. *Information Processing Letters* 70(1), 39–45 (1999)
10. Sheyner, O., Wing, J.M.: Tools for Generating and Analyzing Attack Graphs. In: Proceedings of Workshop on Formal Methods for Components and Objects, pp. 344–371 (2004)
11. Sheyner, O., Haines, J., Jha, S., Lippmann, R., Wing, J.M.: Automated Generation and Analysis of Attack Graphs. In: Proceedings of the IEEE Symposium on Security and Privacy, Oakland, CA, May 2002, pp. 273–284 (2002)
12. Wirth, H.: Multicriteria Approximation of Network Design and Network Upgrade Problems. PhD thesis, Department of Computer Science, Würzburg University (2001)

An Observation on Non-Malleable Witness-Indistinguishability and Non-Malleable Zero-Knowledge

Zongyang Zhang, Zhenfu Cao*, and Rong Ma

Department of Computer Science and Engineering
Shanghai Jiao Tong University, Shanghai, P.R. China
{zongyangzhang, zfc, marong}@sjtu.edu.cn

Abstract. Ostrovsky et al. [1] gave the first definition of non-malleable witness-indistinguishable argument systems. A surprising result given by them showed this notion was incomparable with the notion of non-malleable zero-knowledge. However, they only discussed their relations in the interactive setting. In this paper, we make an observation on relation between the two notions in the non-interactive setting. We show the two notions are still incomparable: that is, there are non-malleable non-interactive zero-knowledge proof systems that are not non-malleable non-interactive witness-indistinguishable, and vice versa.

1 Introduction

The notion of zero-knowledge proof systems was first introduced by Goldwasser et al. [2]. Roughly speaking, a zero-knowledge proof allows a prover to convince a verifier the validity of a statement without disclosing anything else to the verifier. A frequently useful relaxation of zero-knowledge, called witness-indistinguishability, was proposed by Feige et al. [3]. A proof system is said to be witness-indistinguishable (WI) if a verifier cannot distinguish interactions with a prover using different witnesses for the some statement.

When a proof system is executed multi-times especially over the internet, consider a scenario where a man-in-the-middle adversary A is simultaneously participating in two interactions, which are called the left and the right interaction, controls all the messages in the interactions, and can schedule the messages at its wishes. In the left interaction, the adversary plays the role of verifier and interacts with an honest prover P on a statement x . In the right interaction, the adversary plays the role of prover and interacts with an honest verifier V on an adaptively chosen statement \tilde{x} which might be related with x . A may modify the proof received from the left interaction and give an accepting proof for \tilde{x} on the right. Non-malleability was introduced by Dolev et al. [4] to describe security requirements in the above scenario. Informally speaking, a proof is said to be non-malleable if, whenever $x \neq \tilde{x}$, the left interaction does not “help” the adversary in convincing the verifier in the right interaction.

* Corresponding author.

Non-malleable zero-knowledge (NMZK) was first constructed by Dolev et al. [4]. Constant round protocols were given in [5,6] using non-black-box techniques. Under new hardness assumptions (adaptive one-way functions), a 4-round NMZK argument system was constructed in [7]. Other results required either the existence of trusted key-registration functionalities [8], or common reference strings (CRS) [9,10].

It is well known that when non-malleability is not considered, zero-knowledge implies witness-indistinguishability [3]. Ostrovsky et al. [1] first combined witness-indistinguishability and non-malleability and gave the first definition of non-malleable witness-indistinguishable (NMWI) argument systems. They showed the existence of a constant-round NMWI argument system for every NP language in the plain model (i.e., without set-up assumptions) and explored the relation between the notions of NMWI and NMZK. Moreover, a surprising result from their work was that these two notions were incomparable. However, they only showed this result in the interactive setting. In this paper, in order to deeply clarify the two notions, we further explore relation between the two notions in the non-interactive setting and show the following result:

Theorem 1. *Under standard complexity-theoretic assumptions, there exist proof systems that are non-malleable non-interactive zero-knowledge (NIZK) but not non-malleable non-interactive witness-indistinguishable (NIWI), and vice versa.*

Section 2 contains basic definitions of non-malleable NIZK and non-malleable NIWI. In section 3, we will prove Theorem 1.

2 Preliminaries

Consider an NP language L as given by a natural witness relation R_L , i.e., where $x \in L$ iff there is a w such that $R(x, w)$ holds. For $x \in L$, any such w is said to be a witness. Let $R_L(x)$ denote the set of witnesses for the membership $x \in L$, i.e., $R_L(x) = \{y \mid \exists y \text{ s.t. } (x, y) \in R_L\}$. A function $\mu(\cdot)$, where $\mu : \mathbb{N} \rightarrow [0, 1]$ is called *negligible* if for every positive polynomial $p(\cdot)$ and for all sufficiently large n , $\mu(n) < \frac{1}{p(n)}$. In the rest of this paper, we will use $k = |x|$ as the security parameter.

2.1 Non-Malleable Non-Interactive Zero-knowledge

Intuitively, in a NIZK proof system, non-malleability tries to capture the requirement that whatever an adversary can prove after getting polynomially many NIZK proofs for statements of its choosing from an honest prover in the left interaction, it could have proven without seeing them, except for the ability to duplicate proofs. In this work, we use tag-based non-malleable NIZK proof systems. Owing to the page limit, the formal definition of tag-based non-malleable NIZK proof systems are presented in the full version.¹

¹ There are standard ways to transform a non-malleable NIZK proof system into tag-based non-malleable by simply concatenating the tag to the statement being proved.

2.2 Non-Malleable Non-Interactive Witness-Indistinguishability

A proof system is WI if the verifier cannot tell which of the witnesses is being used by the prover to carry out the proof, even if the verifier knows both witnesses. A special kind of WI proof systems is zaps as proposed by Dwork and Naor [11]. Groth et al. [12] constructed a *non-interactive zap* for every NP language in the plain model based on standard cryptographic assumption. For page limited, the formal definitions of WI proofs and non-interactive zap are presented in the full version.

When considering man-in-the-middle attacks, we define non-malleable NIWI proof systems. We require the witness “encoded” in the right proof given by a man-in-the-middle adversary A is independent from the witness used by the honest prover in the left proof.² In order to well define the “encoded” witness in a proof, we focus on commit-and-prove proof systems [9]. We adjust the definition to the non-interactive setting. A *non-interactive commit-and-prove* proof system (K, P, V) for NP language L comprises two phases. On input x , P sends to V a commitment to witness w in the first phase. In the second phase, P proves to V that the committed w is a witness for the membership of x in L .

Definition 1 (Non-Malleable Non-Interactive Witness-Indistinguishability). *A non-interactive commit-and-prove proof system $\Pi = (K, P, V)$ for NP language L with witness relation R_L is a non-malleable NIWI proof system, if for all common reference string $\sigma \leftarrow K(1^k)$, for all probabilistic polynomial-time (PPT) adversary A , and for all PPT algorithms D , there exists a negligible function $\alpha(\cdot)$ such that for all $x \in L$, all witnesses $w_0, w_1 \in R_L(x)$ and all $z \in \{0, 1\}^*$, we have that*

$$\left| \Pr[D(x, w_0, w_1, z, \text{WIMIM}_V^A(\sigma, x, w_0, z)) = 1] - \Pr[D(x, w_0, w_1, z, \text{WIMIM}_V^A(\sigma, x, w_1, z)) = 1] \right| < \alpha(k)$$

where the experiment $\text{WIMIM}_V^A(\sigma, x, w, z)$ is defined as follows:

```

WIMIM_V^A(σ, x, w, z)
  π ← P(σ, x, w)
  (x', π') ← A(σ, π, z)
  Let w' be the witness encoded in proof π'.
  (i.e., the witness committed to by the first message of A.)
  If x = x' or V(σ, x', π') = 0, set w' = “⊥”.
  return w'.
    
```

Since we cannot prevent from copying, we rule out the cases that the statement proved in the right interaction is the same as that in the left interaction. However, in many situations it might be important to protect against an attacker that attempts to prove even the same statement. In order to deal with this problem, we define a tag-based variant of non-malleable non-interactive witness-indistinguishability.

² The reason why we define non-malleable NIWI in this way is similar to that of [1].

Definition 2 (Tag-Based Non-Malleable Non-Interactive Witness Indistinguishability). A non-interactive commit-and-prove proof system $\Pi = \{(K, P_{\text{tag}}, V_{\text{tag}})\}_{\text{tag}}$ for NP language L with witness relation R_L is a tag-based non-malleable NIWI proof system with tags of length $l(k)$, if for all common reference string $\sigma \leftarrow K(1^k)$, for all PPT adversary A , and for all PPT algorithms D , there exists a negligible function $\alpha(\cdot)$ such that for all $x \in L$, for all tags $\text{tag} \in \{0, 1\}^l$, for all witnesses $w_0, w_1 \in R_L(x)$ and all $z \in \{0, 1\}^*$, we have that

$$\left| \Pr[D(x, w_0, w_1, z, \text{TWIMIM}_V^A(\text{tag}, \sigma, x, w_0, z)) = 1] - \Pr[D(x, w_0, w_1, z, \text{TWIMIM}_V^A(\text{tag}, \sigma, x, w_1, z)) = 1] \right| < \alpha(k)$$

where the experiment $\text{TWIMIM}_V^A(\text{tag}, \sigma, x, w, z)$ is defined as follows:

$\text{TWIMIM}_V^A(\text{tag}, \sigma, x, w, z)$
 $\pi \leftarrow P_{\text{tag}}(\sigma, x, w)$
 $(\text{tag}', x', \pi') \leftarrow A(\text{tag}, \sigma, \pi)$
 Let w' be the witness encoded in proof π' .
 (i.e., the witness committed to by the first message of A).
 If $\text{tag}' = \text{tag}$ or $V_{\text{tag}'}(\sigma, x', \pi') = 0$, set $w' = \perp$.
 return w' .

3 Separations between Non-Malleable NIWI and Non-Malleable NIZK

We present our main result in this section. We show the notion of non-malleable NIWI and the notion of non-malleable NIZK are incomparable. We design protocols that distinguish between the two notions.

3.1 Non-Malleable NIZK Proof System

Next, we construct a protocol that is a non-malleable NIZK proof but not a non-malleable NIWI proof. The construction is similar to that of [1]. We will prove the following theorem.

Theorem 2. Assume that there exist trapdoor permutations and dense public-key cryptosystems. Then for any non-trivial NP language there exists a non-malleable NIZK proof system that is not non-malleable NIWI.

Let $\text{CS}=(\text{Com},\text{Ver})$ be a non-interactive statistically binding commitment scheme constructed from one-way permutations [13]. Note the commitment scheme CS uses $6k$ bits to commit to one bit. Let L be an NP language and W be its witness relation. Let L' be the language consisting of pairs (x, com) for which there exist (w, dec) such that $\text{Ver}(w, \text{com}, \text{dec}) = 1$ and $(x, w) \in W$. Suppose $\Pi = \{(K, P_{\text{tag}}, V_{\text{tag}}, S, \mathcal{E})\}_{\text{tag}}$ be a family of tag-based non-malleable NIZK

proof for L' . Let $SIG = (S\text{Gen}, S\text{Sign}, S\text{Ver})$ be a secure strong one-time signature scheme [14].

Then we construct the family of non-malleable NIZK proof system $\Gamma = \{(K, P_{\text{TAG}}, V_{\text{TAG}}, S, E)\}_{\text{TAG}}$ for language L in Fig. 11. Algorithms S, E will be depicted in the proof.

Γ is a commit-and-prove proof system. Protocol Γ consists of two subprotocols, each of which being a commit-and-prove proof, since in each subprotocol, on input x , a commitment com_i of a witness w for $x \in L$ is first generated, then a non-malleable NIZK protocol proving the committed value is a valid witness for membership of x in L is executed. However, we divide the whole protocol in a different way, which consists of a commitment com_0 to a witness w for $x \in L$ and a residual protocol Γ' . Γ' is a proof system for proving that com_0 is indeed the commitment of a valid witness. The proof of completeness property is straightforward. In the following we will prove the protocol is non-malleable, it also implies knowledge soundness (i.e., proof of knowledge).

Γ is non-malleable NIZK. In order to prove the protocol Γ is non-malleable NIZK. We have to show it is NIZK and non-malleable.

NIZK: To show that Γ is unbounded NIZK, we construct a simulator $S = (S_1, S_2)$ that simulates the output of each PPT adversary A .

S_1 runs \mathcal{S}_1 twice and gets $\sigma = (\sigma_0 \circ \sigma_1), \tau = (\tau_0 \circ \tau_1)$. Whenever A submits query $(x_j, w_j, \text{TAG}_j = (t_j^0, t_j^1))$, for $i = 0, 1$, S_2 computes commitment com_j^i of $|x_j|$, and invokes $S\text{Gen}(1^k)$ to generate a verification/signing key pair (vk_j^i, sk_j^i) for the signature scheme SIG . Then S_2 invokes \mathcal{S}_2 on query $(\sigma_0, \tau_0, vk_j^0 \circ 0, x_j \circ \text{com}_j^0)$ and $(\sigma_1, \tau_1, vk_j^1 \circ 1, x_j \circ \text{com}_j^1)$. Upon receiving proof tr_j^0, tr_j^1 in response, S_2 computes $s_j^i = S\text{Sign}(sk_j^i, t_j^i \circ \text{com}_j^i \circ tr_j^i \circ i)$ for $i = 0, 1$, and returns A the value $(\text{com}_j^0, tr_j^0, vk_j^0, s_j^0; \text{com}_j^1, tr_j^1, vk_j^1, s_j^1)$. Finally output whatever A outputs.

We next show the function $|\Pr[\text{Expt}_A(k) = 1] - \Pr[\text{Expt}_A^S(k) = 1]|$ is negligible. Recall $\text{Expt}_A^S(k)$ is the simulation experiment. Denote by $X = (x_1, \dots, x_m)$ all the inputs query by A and $W = (w_1, \dots, w_m)$ the witness such that $w_1 \in R_L(x_1), \dots, w_m \in R_L(x_m)$. Denote by $\text{Expt}_{\text{HYB1}}(X, W, z)$ the experiment as $\text{Expt}_A^S(k)$ except that com_j^0 is a commitment to w_j for all $i \in [m]$. The fact that $|\Pr[\text{Expt}_A^S(k) = 1] - \Pr[\text{Expt}_{\text{HYB1}}(X, W, z) = 1]|$ is negligible follows from the hiding property of the commitment scheme CS . Next define experiment $\text{Expt}_{\text{HYB2}}(X, W, z)$ exactly as $\text{Expt}_{\text{HYB1}}(X, W, z)$ with the exception that com_j^1 is a commitment to w_j for all $j \in [m]$. That $|\Pr[\text{Expt}_{\text{HYB2}}(X, W, z) = 1] - \Pr[\text{Expt}_{\text{HYB1}}(X, W, z) = 1]|$ is negligible follows from the hiding property of CS . Next denote by $\text{Expt}_{\text{HYB3}}(X, W, z)$ the experiment as $\text{Expt}_{\text{HYB2}}(X, W, z)$ except that the proof for (x_j, com_j^0) is generated using actual witness (w_j, dec_j^0) for all $j \in [m]$. The fact that the function $|\Pr[\text{Expt}_{\text{HYB2}}(X, W, z) = 1] - \Pr[\text{Expt}_{\text{HYB3}}(X, W, z) = 1]|$ is negligible follows from the computational ZK property of Π . Recall that $\text{Expt}_A(k)$ is the real experiment of P_{TAG} interacting with A . Note $|\Pr[\text{Expt}_{\text{HYB3}}(X, W, z) = 1] - \Pr[\text{Expt}_A(k) = 1]|$ is negligible following from the

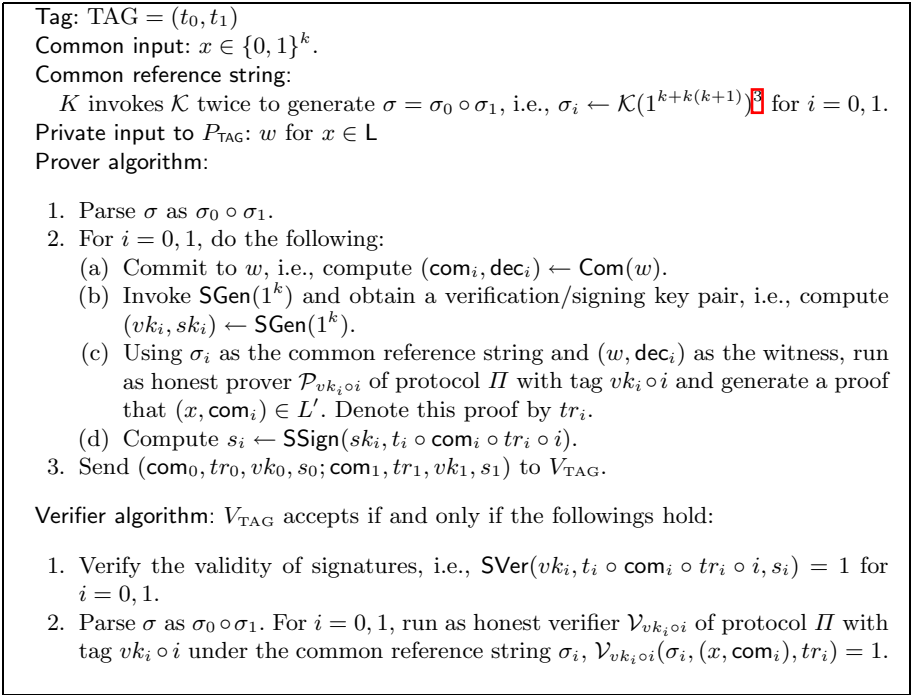


Fig. 1. The non-malleable NIZK proof

computational ZK property of Π . Finally, we have shown that $|\Pr[\text{Expt}_A^S(k) = 1] - \Pr[\text{Expt}_A(k) = 1]|$ is negligible. This proves the ZK property of protocol Γ .

Non-malleability: According to the non-malleability definition (shown in the full version), we have to design an algorithm $E = \{E_1, E_2\}$ for protocol Γ . Recall $\mathcal{E} = \{\mathcal{E}_1, \mathcal{E}_2\}$ is the extractor of protocol Π .

E_1 invokes \mathcal{E}_1 twice and gets $(\sigma_0 \circ \sigma_1, \tau_0^1 \circ \tau_0^2, \tau_1^1 \circ \tau_1^2)$. Whenever the adversary A asks for proof of some instance x_j with tags TAG = (t_j^0, t_j^1) , for $i = 0, 1$, S_2 computes commitment com_i of $|x_j|$, generates verification/signing keys $(vk_j^i, sk_j^i) \leftarrow \text{SGen}(1^k)$, and runs \mathcal{S}_2 on inputs (x_j, com_j^i) with auxiliary input τ_i^1 under common reference string σ_j^i and tag t_j^i . Upon receiving tr_j^i in response, S_2 invokes the signing algorithm and computes $s_j^i = \text{SSign}(sk_j^i, t_j^i \circ \text{com}_j^i \circ tr_j^i \circ i)$ for $i = 0, 1$ and sends $(\text{com}_j^0, tr_j^0, vk_j^0, s_j^0; \text{com}_j^1, tr_j^1, vk_j^1, s_j^1)$ to A . Finally, if A outputs $(\text{TAG}', x', (\text{com}'_0, tr'_0, vk'_0, s'_0; \text{com}'_1, tr'_1, vk'_1, s'_1))$, E_2 invokes \mathcal{E}_2 twice to extract the witness w' for x' .

Recall the experiments $\text{TExpt}_{A,R}^S(k)$ and $\text{TExpt}_{A,R}^E(k)$ in the non-malleability definition. We have to show $|\Pr[\text{TExpt}_{A,R}^S(k)] - \Pr[\text{TExpt}_{A,R}^E(k)]|$ is negligible.

³ Note for each NP language, the witness length is polynomial bounded to the length of its instance. Here, for simplicity, we assume the witness length is equal to the length of its instance. The length of the commitment com below is $k(k+1)$.

Note E actually outputs a witness for an accepting right proof whenever A uses for the right proof tag (t'_0, t'_1) different from the tag (t_0, t_1) used by S in the left proof. Indeed, E fails only in the case that the tag used by A for the subprotocol is equal to one of the tags used by S in the left proof. As the tags in the subprotocol are concatenated with a bit 0 or 1, the only way E fails in extracting the witness is the case that A picks the same signature verification key used by S in the left proof. Since the proof generated by A is accepting, we succeed in generating a signature of a new message for a public verification key chosen by S . This contradicts the security of the signature scheme SIG.

Γ is not non-malleable NIWI. We show Γ is not a tag-based non-malleable NIWI proof system. We design a special PPT adversary A that always produces an accepting right proof that encodes the same witness as the one encoded in the left proof. Furthermore, A generates the right proof with a tag different from the one used in the left proof.

The adversary A runs as honest verifier in the left proof. The common reference string in both interactions are the same. When receiving from a left proof messages $(\text{com}_0, tr_0, vk_0, s_0; \text{com}_1, tr_1, vk_1, s_1)$ on input x with tag (t_0, t_1) , A tries to start a right proof on input x and tag (t'_0, t'_1) where $t'_0 = t_0$ and $t'_1 \neq t_1$. The adversary A who has the witness w as auxiliary input first generates a pair of signature keys $(vk'_1, sk'_1) \leftarrow \text{SGen}(1^k)$, commits to w and computes $(\text{com}'_1, \text{dec}'_1) \leftarrow \text{Com}(w)$. Then for the first subprotocol of the right proof, A using $(\text{com}_0, tr_0, vk_0, s_0)$ as the proof. For the second subprotocol, A runs honest prover $\mathcal{P}_{vk'_1 \circ 1}$'s strategy and uses witness (w, dec'_1) to generate the proof tr'_1 . In the end, A invokes SSign with signing key sk'_1 to generates a signature s'_1 for the message $t'_1 \circ \text{com}'_1 \circ tr'_1 \circ 1$. The final proof for x is $(\text{com}_0, tr_0, vk_0, s_0; \text{com}'_1, tr'_1, vk'_1, s'_1)$. It is obvious that, if the left proof is accepting, then the right proof is also accepting. It is clear that the tags in both proofs are different while the witness encoded in both proofs are the same.

Note that Γ uses a tag-based non-malleable NIZK proof system Π which is based on the existence of dense public-key cryptosystem in addition to trapdoor permutations [10,15]. The commitment scheme CS assumes the existence of one-way permutations [13]. The signature scheme SIG can be constructed from one-way functions [14]. Thus, we prove Theorem 2.

3.2 Non-Malleable NIWI Proof System

In this part, we try to find a protocol that is non-malleable NIWI but not non-malleable NIZK. We prove the following theorem.

Theorem 3. *Assume that there exist trapdoor permutations and dense public-key cryptosystems. Assume the subgroup decision assumption holds. Then for any non-trivial NP language there exists a non-malleable NIWI proof system that is not non-malleable NIZK.*

Let $\text{CS}=(\text{Com},\text{Ver})$ be a non-interactive statistically binding commitment scheme [13]. Let L be an $\text{NP}\setminus\text{BPP}$ language and W be its witness relation. Let Σ be

⁴ The witness w is not necessarily be the same as that encoded in the left proof.

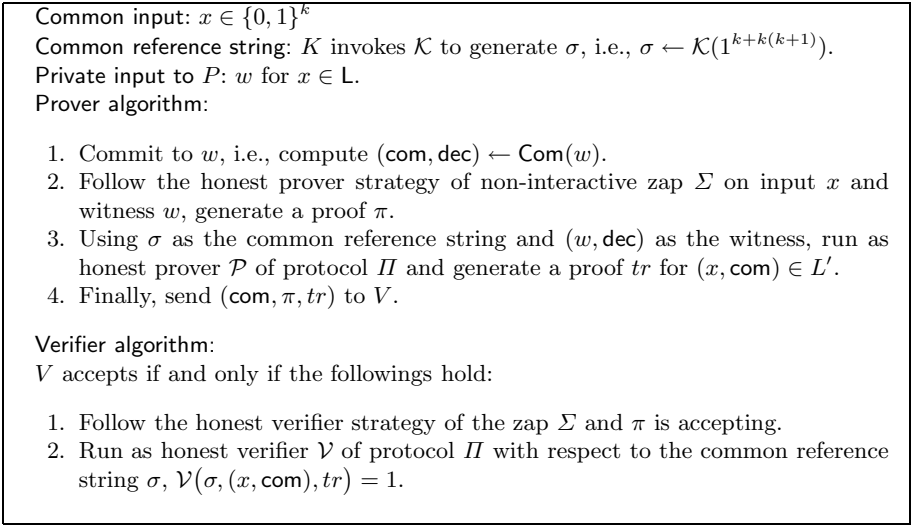


Fig. 2. The non-malleable NIWI proof

a non-interactive zap for L [12]. Define language $\mathsf{L}' = \{(x, \text{com})\}$ where there exist (w, dec) such that $\text{Ver}(w, \text{com}, \text{dec}) = 1$ and $(x, w) \in \mathsf{W}$. Suppose $\Pi = (\mathcal{K}, \mathcal{P}, \mathcal{V}, \mathcal{S}, \mathcal{E})$ be a non-malleable NIZK proof of knowledge for language L' [10].

Consider the following proof system $\Gamma = (K, P, V)$ in Fig. 2.⁵

Γ is a commit-and-prove proof system. Γ is composed of two parts, one of which is a commitment to w , and the other of which forms a proof system Γ' for proving that the committed value is a witness for the given statement. Completeness follows directly from the completeness of Σ and Π . Soundness follows from the proof of knowledge property of Π .

Γ is non-malleable NIWI. We prove by hybrid argument. Let $x \in \mathsf{L}$ and $w_1, w_2 \in \mathsf{W}(x)$. Fix a man-in-the-middle adversary A that violates the non-malleable witness-indistinguishability property of Γ . We design a sequence of experiments $\text{Expt}_i(\sigma, x, w_0, w_1, z)$ for $0 \leq i \leq 5$. The output of Expt_i is the witness encoded in the accepting right proof given by A .

1. The first experiment $\text{Expt}_0(\sigma, x, w_0, w_1, z)$ is simply the experiment in which A receives proof from an honest prover with w_0 as a witness in the left interaction.

The output of $\text{Expt}_0(\sigma, x, w_0, w_1, z)$ is identical to that of the experiment $\text{WIMIM}_V^A(\sigma, x, w_0, z)$.

2. The second experiment $\text{Expt}_1(\sigma, x, w_0, w_1, z)$ is simply the experiment in which A receives proof from simulator \mathcal{S} of Π in the left proof.

⁵ We can also construct a tag-based non-malleable NIWI proof system Γ' by letting Π be a tag-based non-malleable NIZK proof system. Γ' is the same as Γ except that in the third step of the prover's algorithm, P runs honest prover of Π with the same tag of Γ' .

The outputs of experiments Expt_0 and Expt_1 are computationally indistinguishable since Π is non-malleable and ZK.

3. The third experiment $\text{Expt}_2(\sigma, x, w_0, w_1, z)$ proceeds as in the second, except that for commitment in the left proof, it computes $(\text{com}, \text{dec}) \leftarrow \text{Com}(0^{|w_0|})$.

The outputs of experiments Expt_1 and Expt_2 are computationally indistinguishable following from the hiding property of the commitment scheme CS and the strong witness-indistinguishability (implied by ZK) of protocol Π .

4. The fourth experiment $\text{Expt}_3(\sigma, x, w_0, w_1, z)$ proceeds as in the third, except that for proof π in the left proof, it runs as the honest prover of Σ with witness w_1 instead of w_0 to obtain π .

The outputs of experiments Expt_2 and Expt_3 are computationally indistinguishable since the non-interactive zap Σ is WI.

5. The fifth experiment $\text{Expt}_4(\sigma, x, w_0, w_1, z)$ proceeds as the fourth, except that in the left proof com is computed as $(\text{com}, \text{dec}) \leftarrow \text{Com}(w_1)$.

The outputs of experiments Expt_3 and Expt_4 are computationally indistinguishable following from the hiding property of CS and strong witness-indistinguishability of Π .

6. The sixth experiment $\text{Expt}_5(\sigma, x, w_0, w_1, z)$ proceeds as the fifth, except that in the left proof A interacts with an honest prover who uses w_1 as witness.

The outputs of experiments Expt_4 and Expt_5 are computationally indistinguishable following from the non-malleable ZK property of Π . The output of Expt_5 is identical to that of the experiment $\text{WIMIM}_V^A(\sigma, x, w_1, z)$.

Finally we have proven that the output of $\text{WIMIM}_V^A(\sigma, x, w_0, z)$ and that of $\text{WIMIM}_V^A(\sigma, x, w_1, z)$ are computationally indistinguishable. This reaches a contradiction. So Γ is non-malleable NIWI.

Γ is not non-malleable NIZK. Assume on the contrary Γ is non-malleable NIZK. Using the ZK property, we can find a simulator S that simulates the view of A . Note the simulated transcript consists mainly two parts, one of which is the simulated proof for non-interactive zap Σ and the other being the simulated proof for protocol Π . From the simulated transcripts computed by S , we extract the part of transcripts π for the zap Σ . π is accepting for x . Furthermore, by the definition of zap, Σ is a non-interactive proof system for L in the plain model. We conclude that Σ is a one-round zero-knowledge proof system for non-trivial NP language L in the plain model, which is impossible unless $\text{BPP} = \text{NP}$ [16].

The non-malleable NIZK protocol Π is based on trapdoor permutation and dense public-key cryptosystem [10]. The non-interactive zap Σ exists when the subgroup decision assumption holds [12]. The commitment scheme CS assumes the existence of one-way permutations [13]. Thus, we have proved Theorem 3.

Finally Theorem 2 in addition to Theorem 3 imply Theorem 1.

Acknowledgement. We thank the anonymous reviewers for their suggestions to improve the paper. This work is supported by National Nature Science Foundation of China No.60673079 and No.60773086, and National 973 Program No. 2007CB31120.

References

1. Ostrovsky, R., Persiano, G., Visconti, I.: Constant-round concurrent nmwi and its relation to nmzk. Technical Report ECCC report TR06-095 (2006)
2. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. *SIAM J. Comput.* 18, 186–208 (1989)
3. Feige, U., Shamir, A.: Witness indistinguishable and witness hiding protocols. In: *STOC*, pp. 416–426. ACM, New York (1990)
4. Dolev, D., Dwork, C., Naor, M.: Nonmalleable cryptography. *SIAM J. Comput.* 30, 391–437 (2000)
5. Barak, B.: Constant-round coin-tossing with a man in the middle or realizing the shared random string model. In: *FOCS*, pp. 345–355. IEEE Computer Society, Los Alamitos (2002)
6. Pass, R., Rosen, A.: New and improved constructions of nonmalleable cryptographic protocols. *SIAM J. Comput.* 38, 702–752 (2008)
7. Pandey, O., Pass, R., Vaikuntanathan, V.: Adaptive one-way functions and applications. In: Wagner, D. (ed.) *CRYPTO 2008*. LNCS, vol. 5157, pp. 57–74. Springer, Heidelberg (2008)
8. Barak, B., Canetti, R., Nielsen, J.B., Pass, R.: Universally composable protocols with relaxed set-up assumptions. In: *FOCS*, pp. 186–195. IEEE Computer Society, Los Alamitos (2004)
9. Canetti, R., Lindell, Y., Ostrovsky, R., Sahai, A.: Universally composable two-party and multi-party secure computation. In: *STOC*, pp. 494–503. ACM, New York (2002)
10. Santis, A.D., Crescenzo, G.D., Ostrovsky, R., Persiano, G., Sahai, A.: Robust non-interactive zero knowledge. In: Kilian, J. (ed.) *CRYPTO 2001*. LNCS, vol. 2139, pp. 566–598. Springer, Heidelberg (2001)
11. Dwork, C., Naor, M.: Zaps and their applications. In: *FOCS*, pp. 283–293. IEEE Computer Society, Los Alamitos (2000)
12. Groth, J., Ostrovsky, R., Sahai, A.: Non-interactive zaps and new techniques for nizk. In: Dwork, C. (ed.) *CRYPTO 2006*. LNCS, vol. 4117, pp. 97–111. Springer, Heidelberg (2006)
13. Goldreich, O.: *The Foundations of Cryptography*, vol. 1. Cambridge University Press, US (2001)
14. Sahai, A.: Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In: *FOCS*, pp. 543–553. IEEE Computer Society, Los Alamitos (1999)
15. Santis, A.D., Crescenzo, G.D., Persiano, G.: Necessary and sufficient assumptions for non-iterative zero-knowledge proofs of knowledge for all np relations. In: Montanari, U., Rolim, J.D.P., Welzl, E. (eds.) *ICALP 2000*. LNCS, vol. 1853, pp. 451–462. Springer, Heidelberg (2000)
16. Goldreich, O., Oren, Y.: Definitions and properties of zero-knowledge proof systems. *J. Cryptology* 7, 1–32 (1994)

Author Index

- Abu-Khizam, Faisal N. 81
Ambos-Spies, Klaus 88
Ando, Ei 98
- Bampas, Evangelos 108
Bereg, Sergey 118
Berendsen, Jasper 128
- Cai, Jin-Yi 138, 460
Cao, Zhenfu 470
Carbone, Alessandra 6
Cenzer, Douglas 420
Chang, Maw-Shang 150
Chang, Ruei-Yuan 158
Chen, Jianer 168
Chen, Taolue 128
Cooper, S. Barry 18
- Das Sarma, Atish 178
Dib, Linda 6
Doberkat, Ernst-Erich 192
- Elberfeld, Michael 201
- Feng, Qilong 211
Fernau, Henning 59
Fiala, Jiří 221
Franklin, Johanna N.Y. 420
Fu, Bin 231
- Göbel, Andreas-Nikolas 108
Golovach, Petr A. 221
Grumbach, Stéphane 430
Guo, Jiong 39
- Heggernes, Pinar 241
Hennessy, Matthew 4
Hinkelmann, Markus 251
Hung, Ling-Ju 150
- Jakoby, Andreas 251
Jansen, David N. 128
Jiang, Bo 345
Jiang, Guohong 390
Jiang, Minghui 118
- Kanj, Iyad A. 49, 168
Kao, Ming-Yang 231
Kaplan, Marc 261
Kloks, Ton 150
Kräling, Thorsten 88
Kratochvíl, Jan 221
- Lafitte, Grégory 271
Laplante, Sophie 261
Lee, Guanling 158
Lipton, Richard J. 178
Liu, Jiang 420
Liu, Yang 211
Lokshtanov, Daniel 281
Lu, Pinyan 138
Lu, Songjian 211
- Ma, Rong 470
Martens, Maren 291
Meister, Daniel 241
Mizuki, Takaaki 301
Mnich, Matthias 281
- Nanongkai, Danupon 178
Nishizeki, Takao 301
- Ono, Hirotaka 98
- Pagourtzis, Aris 108
Papadopoulos, Charis 241
Peng, Sheng-Lung 150, 158
Pinto, Paulo Eustáquio Duarte 311
Protti, Fábio 311
- Raible, Daniel 59
- Sadakane, Kunihiko 98
Saurabh, Saket 281
Schnoor, Ilka 201
Schubert, Christoph 325
Sorbi, Andrea 29, 335
Swegles, Kyle 380
Szwarcfiter, Jayme Luiz 311

- Tan, Xuehou 345
Tang, Linqing 460
Tantau, Till 201
Tentes, Aris 108
Toulouse, Sophie 360
Tsubata, Hitoshi 301
- van Zuylen, Anke 370
Valiant, Leslie G. 1
Vardi, Moshe Y. 3
- Wang, Fangju 380
Wang, Jianxin 211, 390
Wang, Lusheng 231, 400
Weiss, Michael 271
Witt, Carsten 410
Wolfer Calvo, Roberto 360
- Wu, Guohua 335, 420
Wu, Zhilin 430
- Xia, Ge 168
Xia, Mingji 138
- Yamashita, Masafumi 98
Yang, Boting 118
Yang, Yue 335
Yin, Xiao 440
- Zhang, John Z. 450
Zhang, Peng 460
Zhang, Zongyang 470
Zhao, Wenbo 460
Zhu, Binhai 71, 118, 400
Zhu, Daming 440