

Zbornik 19. mednarodne multikonference

INFORMACIJSKA DRUŽBA - IS 2016

Zvezek H

Proceedings of the 19th International Multiconference

INFORMATION SOCIETY - IS 2016

Volume H

Middle-European Conference on Applied Theoretical Computer Science (MATCOS 2016)

Uredil / Edited by
Prof. Andrej Brodnik

<http://is.ijs.si>

12.-13. oktober 2016 / 12-13 October 2016
Ljubljana, Slovenia

Zbornik 19. mednarodne multikonference
INFORMACIJSKA DRUŽBA – IS 2016
Zvezek H

Proceedings of the 19th International Multiconference
INFORMATION SOCIETY – IS 2016
Volume H

**Middle-European Conference on Applied Theoretical
Computer Science (MATCOS 2016)**

Uredil / Edited by

Prof. Andrej Brodnik

12.-13. oktober 2016 / 12-13 October 2016
Ljubljana, Slovenia

Urednik:

Andrej Brodnik

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko

Univerza na primorskem, Fakulteta za matematiko, naravoslovje in informacijske tehnologije

Založnik: Institut »Jožef Stefan«, Ljubljana

Priprava zbornika: Mitja Lasič, Vesna Lasič, Lana Zemljak

Oblikovanje naslovnice: Vesna Lasič

Dostop do e-publikacije:

<http://library.ijs.si/Stacks/Proceedings/InformationSociety>

Ljubljana, oktober 2016

CIP - Kataložni zapis o publikaciji
Narodna in univerzitetna knjižnica, Ljubljana

004(082) (0.034.2)

MIDDLE-European Conference on Applied Theoretical Computer Science (2016 ;
Ljubljana)

Middle-European Conference on Applied Theoretical Computer Science (MATCOS
2016) [Elektronski vir] : zbornik 19. mednarodne multikonference Informacijska
družba - IS 2016, 12.-13. oktober 2016, [Ljubljana, Slovenija] = proceedings of
the 19th International Multiconference Information Society - IS 2016, 12.-13
October 2016, Ljubljana, Slovenia : zvezek H = volume H / uredil, edited by
Andrej Brodnik. - El. zbornik. - Ljubljana : Institut Jožef Stefan, 2016

Način dostopa (URL):

[http://library.ijs.si/Stacks/Proceedings/InformationSociety/2016/IS2016_Volume_H%
20-%20MATCOS.pdf](http://library.ijs.si/Stacks/Proceedings/InformationSociety/2016/IS2016_Volume_H%20-%20MATCOS.pdf)

ISBN 978-961-264-104-7 (pdf)

1. Gl. stv. nasl. 2. Brodnik, Andrej 3. Mednarodna multikonferenca Informacijska
družba (19 ; 2016 ; Ljubljana)
29879847

PREDGOVOR MULTIKONFERENCI INFORMACIJSKA DRUŽBA 2016

Multikonferenca Informacijska družba (<http://is.ijs.si>) je z devetnajsto zaporedno prireditvijo osrednji srednjeevropski dogodek na področju informacijske družbe, računalništva in informatike. Letošnja prireditev je ponovno na več lokacijah, osrednji dogodki pa so na Institutu »Jožef Stefan«.

Informacijska družba, znanje in umetna inteligenca so spet na razpotju tako same zase kot glede vpliva na človeški razvoj. Se bo eksponentna rast elektronike po Moorovem zakonu nadaljevala ali stagnerala? Bo umetna inteligenca nadaljevala svoj neverjetni razvoj in premagovala ljudi na čedalje več področjih in s tem omogočila razcvet civilizacije, ali pa bo eksponentna rast prebivalstva zlasti v Afriki povzročila zadušitev rasti? Čedalje več pokazateljev kaže v oba ekstrema – da prehajamo v naslednje civilizacijsko obdobje, hkrati pa so planetarni konflikti sodobne družbe čedalje težje obvladljivi.

Letos smo v multikonferenco povezali dvanajst odličnih neodvisnih konferenc. Predstavljenih bo okoli 200 predstavitev, povzetkov in referatov v okviru samostojnih konferenc in delavnic. Prireditve bodo spremljale okrogle mize in razprave ter posebni dogodki, kot je svečana podelitev nagrad. Izbrani prispevki bodo izšli tudi v posebni številki revije *Informatica*, ki se ponaša z 39-letno tradicijo odlične znanstvene revije. Naslednje leto bo torej konferenca praznovala 20 let in revija 40 let, kar je za področje informacijske družbe častitljiv dosežek.

Multikonferenco Informacijska družba 2016 sestavljajo naslednje samostojne konference:

- 25-letnica prve internetne povezave v Sloveniji
- Slovenska konferenca o umetni inteligenci
- Kognitivna znanost
- Izkopavanje znanja in podatkovna skladišča
- Sodelovanje, programska oprema in storitve v informacijski družbi
- Vzgoja in izobraževanje v informacijski družbi
- Delavnica »EM-zdravje«
- Delavnica »E-heritage«
- Tretja študentska računalniška konferenca
- Računalništvo in informatika: včeraj za jutri
- Interakcija človek-računalnik v informacijski družbi
- Uporabno teoretično računalništvo (MATCOS 2016).

Soorganizatorji in podporniki konference so različne raziskovalne institucije in združenja, med njimi tudi ACM Slovenija, SLAIS, DKZ in druga slovenska nacionalna akademija, Inženirska akademija Slovenije (IAS). V imenu organizatorjev konference se zahvaljujemo združenjem in inštitucijam, še posebej pa udeležencem za njihove dragocene prispevke in priložnost, da z nami delijo svoje izkušnje o informacijski družbi. Zahvaljujemo se tudi recenzentom za njihovo pomoč pri recenziranju.

V 2016 bomo četrtoč podelili nagrado za življenjske dosežke v čast Donalda Michija in Alana Turinga. Nagrado Michie-Turing za izjemen življenjski prispevek k razvoju in promociji informacijske družbe bo prejel prof. dr. Tomaž Pisanski. Priznanje za dosežek leta bo pripadlo prof. dr. Blažu Zupanu. Že šestič podeljujemo nagradi »informacijska limona« in »informacijska jagoda« za najbolj (ne)uspešne poteze v zvezi z informacijsko družbo. Limono je dobilo ponovno padanje Slovenije na lestvicah informacijske družbe, jagodo pa informacijska podpora Pediatrične klinike. Čestitke nagrajencem!

Bojan Orel, predsednik programskega odbora
Matjaž Gams, predsednik organizacijskega odbora

FOREWORD - INFORMATION SOCIETY 2016

In its 19th year, the Information Society Multiconference (<http://is.ijs.si>) remains one of the leading conferences in Central Europe devoted to information society, computer science and informatics. In 2016 it is organized at various locations, with the main events at the Jožef Stefan Institute.

The pace of progress of information society, knowledge and artificial intelligence is speeding up, but it seems we are again at a turning point. Will the progress of electronics continue according to the Moore's law or will it start stagnating? Will AI continue to outperform humans at more and more activities and in this way enable the predicted unseen human progress, or will the growth of human population in particular in Africa cause global decline? Both extremes seem more and more likely – fantastic human progress and planetary decline caused by humans destroying our environment and each other.

The Multiconference is running in parallel sessions with 200 presentations of scientific papers at twelve conferences, round tables, workshops and award ceremonies. Selected papers will be published in the Informatica journal, which has 39 years of tradition of excellent research publication. Next year, the conference will celebrate 20 years and the journal 40 years – a remarkable achievement.

The Information Society 2016 Multiconference consists of the following conferences:

- 25th Anniversary of First Internet Connection in Slovenia
- Slovenian Conference on Artificial Intelligence
- Cognitive Science
- Data Mining and Data Warehouses
- Collaboration, Software and Services in Information Society
- Education in Information Society
- Workshop Electronic and Mobile Health
- Workshop »E-heritage«
- 3st Student Computer Science Research Conference
- Computer Science and Informatics: Yesterday for Tomorrow
- Human-Computer Interaction in Information Society
- Middle-European Conference on Applied Theoretical Computer Science (Matcos 2016)

The Multiconference is co-organized and supported by several major research institutions and societies, among them ACM Slovenia, i.e. the Slovenian chapter of the ACM, SLAIS, DKZ and the second national engineering academy, the Slovenian Engineering Academy. In the name of the conference organizers we thank all the societies and institutions, and particularly all the participants for their valuable contribution and their interest in this event, and the reviewers for their thorough reviews.

For the fourth year, the award for life-long outstanding contributions will be delivered in memory of Donald Michie and Alan Turing. The Michie-Turing award will be given to Prof. Tomaž Pisanski for his life-long outstanding contribution to the development and promotion of information society in our country. In addition, an award for current achievements will be given to Prof. Blaž Zupan. The information lemon goes to another fall in the Slovenian international ratings on information society, while the information strawberry is awarded for the information system at the Pediatric Clinic. Congratulations!

Bojan Orel, Programme Committee Chair
Matjaž Gams, Organizing Committee Chair

KONFERENČNI ODBORI

CONFERENCE COMMITTEES

International Programme Committee

Vladimir Bajic, South Africa
Heiner Benking, Germany
Se Woo Cheon, South Korea
Howie Firth, UK
Olga Fomichova, Russia
Vladimir Fomichov, Russia
Vesna Hljuz Dobric, Croatia
Alfred Inselberg, Israel
Jay Liebowitz, USA
Huan Liu, Singapore
Henz Martin, Germany
Marcin Paprzycki, USA
Karl Pribram, USA
Claude Sammut, Australia
Jiri Wiedermann, Czech Republic
Xindong Wu, USA
Yiming Ye, USA
Ning Zhong, USA
Wray Buntine, Australia
Bezalel Gavish, USA
Gal A. Kaminka, Israel
Mike Bain, Australia
Michela Milano, Italy
Derong Liu, Chicago, USA
Toby Walsh, Australia

Organizing Committee

Matjaž Gams, chair
Mitja Luštrek
Lana Zemljak
Vesna Koricki
Mitja Lasič
Robert Blatnik
Aleš Tavčar
Blaž Mahnič
Jure Šorn
Mario Konecki

Programme Committee

Bojan Orel, chair
Nikolaj Zimic, co-chair
Franc Solina, co-chair
Viljan Mahnič, co-chair
Cene Bavec, co-chair
Tomaž Kalin, co-chair
Jozsef Györkös, co-chair
Tadej Bajd
Jaroslav Berce
Mojca Bernik
Marko Bohanec
Ivan Bratko
Andrej Brodnik
Dušan Caf
Saša Divjak
Tomaž Erjavec
Bogdan Filipič

Andrej Gams
Matjaž Gams
Marko Grobelnik
Nikola Guid
Marjan Heričko
Borka Jerman Blažič Džonova
Gorazd Kandus
Urban Kordeš
Marjan Krisper
Andrej Kuščer
Jadran Lenarčič
Borut Likar
Janez Malačič
Olga Markič
Dunja Mladenič
Franc Novak

Vladislav Rajkovič Grega
Repovš
Ivan Rozman
Niko Schlamberger
Stanko Strmčnik
Jurij Šilc
Jurij Tasič
Denis Trček
Andrej Ule
Tanja Urbančič
Boštjan Vilfan
Baldoimir Zajc
Blaž Zupan
Boris Žemva
Leon Žlajpah

KAZALO / TABLE OF CONTENTS

<i>Middle-European Conference on Applied Theoretical Computer Science (MATCOS 2016)</i>	1
PREDGOVOR / FOREWORD.....	3
PROGRAMSKI ODBORI / PROGRAMME COMMITTEES.....	4
Industrial and Medical Applications.....	5
Customizing Hybrid Optimization for Microwave Tomography / Subotić Miloš, Palfi Laszlo, Pjevalica Nebojša.....	5
Schedule Assignment for Vehicles in Inter-City Bus Transportation over a Planning Period / Dávid Balázs.....	9
A Self-Bounding Branch & Bound Procedure for Truck Routing and Scheduling / Csehi Csongor Gy., Farkas Márk, Tóth Ádám.....	13
Improving Flow Lines by Unbalance / Mihály Zsolt, Lelkes Zoltán.....	16
Process Network Solution of a Soleplate Manufacturer's Extended CPM Problem with Alternatives / Vincze Nándor, Ercsey Zsolt, Kovács Zoltán.....	20
Algorithm design and evaluation.....	24
ON NIST Test of a Novel Cryptosystem Based on Automata Compositions / Dömösi Pál, Gáll József, Horváth Géza, Tihanyi Norbert.....	24
ALGator - An Automatic Algorithm Evaluation System / Dobravec Tomaž.....	28
A Graph to the Pairing strategies of the 9-in-a-Row Game / Györfly Lajos, London András, Makay Géza.....	32
Construction of Orthogonal CC-Set / Brodnik Andrej, Jovičić Vladan, Palangetić Marko, Silai Daniel.....	36
Usage of Hereditary Colorings of Product Graphs in Clique Search Programs / Depolli Matjaž, Konc Janez, Szabo Sandor, Zavalnij Bogdan.....	40
Algorithms Optimization.....	44
Testing the Markowitz Portfolio Optimization Method with Filtered Correlation Matrices / Gera Imre, Bánhelyi Balázs, London András.....	44
Tight Online Bin Packing Algorithm with Buer and Parametric Item Sizes / Békési József, Galambos Gábor.....	48
A Branch-and-Cut Algorithm for the Multi-Depot Rural Postman Problem / Fernández Elena, Laporte Gilbert, Rodríguez Pereira Jessica.....	51
Allocation and Pricing on a Network in Presence of Negative Externalities / Pekec Saša.....	54
Graph Theory.....	57
The Vertex Sign Balance of (Hyper)graphs / Miklos Dezso.....	57
Packing Tree Degree Sequences / Berczi Kristof, Kiraly Zoltan, Liu Changshuo, Miklós István.....	61
Benchmark Problems for Exhaustive Exact Maximum Clique Search Algorithms / Szabo Sandor, Zavalnij Bogdan.....	65
On Embedding Degree Sequences / Csaba Bela, Vasarhelyi Balint.....	68
Algorithms Complexity.....	72
Computational Complexity of the Winner Determination Problem for Geometrical Combinatorial Auctions / Goossens Dries, Vangerven Bart, Spieksma Frits.....	72
Diploid Genome Rearrangement / Miklós István.....	76
Team Work Scheduling / Dosa Gyorgy, Kellerer Hans, Tuza Zsolt.....	80
Incremental 2-D Nearest-Point Search with Evenly Populated Strips / Podgorelec David, Špelič Denis.....	83
Miscellaneous.....	87
Exploratory Equivalence on Hypercube Graphs / Mihelič Jurij, Čibej Uroš, Fürst Luka.....	87
Partitioning Polyominoes into Polyominoes of at Most 8 Vertices, Mobile vs Point Guards / Gyori Ervin, Mezei Tamas.....	91
On Linear Grammars with Exact Control / Angyal Dávid, Nagy Benedek.....	95
Some Computable Functions without Brouwer Fixed-Points / Potgieter Petrus H.....	99
<i>Indeks avtorjev / Author index</i>	103

Zbornik 19. mednarodne multikonference
INFORMACIJSKA DRUŽBA – IS 2016
Zvezek H

Proceedings of the 19th International Multiconference
INFORMATION SOCIETY – IS 2016
Volume H

**Middle-European Conference on Applied Theoretical
Computer Science (MATCOS 2016)**

Uredil / Edited by

Prof. Andrej Brodnik

12.-13. oktober 2016 / 12-13 October 2016
Ljubljana, Slovenia

FOREWORD

MATCOS, Middle European Conference on Applied Theoretical Computer Science, took place at the University of Primorska on October 12th and 13th. This was its fifth overall edition and in these years got wide acceptance as a forum where theory and practice meet in a fruitful dialogue. Moreover, the dialogue does not spawn only between theory and practice, but also between senior and junior researchers. As it is already custom, the first day was devoted student papers, invited talk and some of the regular papers.

Overall at the conference were presented four student papers and 26 regular papers. With MATCOS was this year also collocated a conference StuCosRec (3rd Student Computer Science Research Conference) where additional ten student papers were presented.

The invited talk at MATCOS was titled Algorithms for robot navigation: From optimizing individual robots to particle swarms and given by Sándor Fekete. The regular papers were grouped into six sessions spanning from graph theory all the way to algorithm design and use of theoretical Computer Science results in practice.

This is far the largest MATCOS conference, and we hope to make the next MATCOS even a bigger event.

Koper, Ljubljana, Szeged, October 2016

Programme committee Chairs
Gábor Galambos and Andrej Brodnik

PROGRAMSKI ODBOR / PROGRAMME COMMITTEE

Jacek Blazevicz (Poznan, Poland)
Andrej Brodnik (Koper, Ljubljana, Slovenia) co-chair
Angel Corberan (Valencia, Spain)
Ruben Dorado Vicente (Jaén, Spain)
Elena Fernandez (Barcelona, Spain)
Gábor Galambos (Szeged, Hungary) co-chair
Gabriel Istrate (Timisoara, Romania)
Miklós Krész (Szeged, Hungary)
Silvano Martello (Bologna, Italy)
Benedek Nagy (Famagusta, Cyprus, Turkey)
Gerhard Reinelt (Heidelberg, Germany)
Giovanni Rinaldi (Rome, Italy)
Borut Žalik (Maribor, Slovenia)

Customizing Hybrid Optimization for Microwave Tomography

Milos Subotic*
RT-RK Institute for Computer
Based Systems
Narodnog fronta 23a, 21000
Novi Sad, Serbia
milos.subotic@rt-
rk.uns.ac.rs

Laszlo Palfi
Faculty of Technical Sciences
Trg Dositeja Obradovica 6,
21000 Novi Sad, Serbia
laslo.palfi@rt-rk.com

Nebojsa U. Pjevalica
Faculty of Technical Sciences
Computing and control
engineering dept. Trg Dositeja
Obradovica 6, 21000 Novi
Sad, Serbia
pjeva@uns.ac.rs

ABSTRACT

Microwave tomography is an inverse scattering problem, typically solved through optimization methods. The underlying objective function is ill-posed and expensive for evaluation, making microwave tomography a hard optimization problem. This paper presents a novel optimization heuristic for use in microwave tomography. Landscape analysis of objective function is made. Results from landscape analysis helped creating novel optimization heuristic. Significant acceleration is obtained.

Categories and Subject Descriptors

G.1.6 [Numerical Analysis]: Optimization—*global optimization, unconstrained optimization*

General Terms

Algorithms

Keywords

Hybrid optimization, optimization heuristic, fitness landscape analysis, microwave tomography, inverse scattering problem

1. INTRODUCTION

Microwave tomography (MWT) [13] is imaging modality which obtains an image of dielectric properties of scanned object from scattered microwaves. MWT is an inverse problem - the input and output values, i.e. the input and output wave, are known, and the function, i.e. the dielectric properties, have to be found.

MWT is an imaging method which does not produce radiation like X-Ray CT and promise to be much cheaper than MRI. It is multidisciplinary field involving electromagnetism, antenna design, numeric simulation, optimization methods, computing acceleration and parallelization.

According to the No Free Lunch theorem [20] it is impossible to make a universal meta-heuristic which would optimize equally good on all problems. In other words, optimization heuristic should be customized for every problem at hand. This makes MWT optimization problem interesting for research.

*Corresponding author.

The best images are obtained by means of quantitative MWT, where connection between dielectric properties and waves are described by non-linear scattering equations. As all inverse problems, this one is also ill-posed i.e. multimodal and ill-conditioned. MWT is build on top of a forward solver, a numeric algorithm for solving sets of scattering equations or simulating propagation and scattering of electromagnetic waves. One evaluation of the forward solver could last for seconds [17] or even minutes [4]. This makes MWT a hard optimization problem.

One solution is linearization of scattering equations using Born or Rytov approximations [14]. Also, many regularization methods are used for linearization of inverse problems [18]. This makes the objective function convex i.e. unimodal, which then could be solved by some local optimization method, usually Gauss-Newton [14] or similar method. All these methods are criticized because they could stuck in a local minimum, work only if the contrast is small, and tend to over-smooth the resulting image [13].

Other methods use global optimization techniques. The direct optimization approach tries to find the values of every pixel (or voxel) of an image [17] [16]. The scanned object is partitioned to a grid of pixels and every pixel is an optimization variable. Since only a few degrees of freedom (pixels) are possible to optimize, the resulting images are limited to lower resolution. To downsize the search space, some constraints could be set on the image structure or pixel values [2] [17] [5]. The indirect optimization approach optimizes the shapes and dielectric properties of some objects [4]. This technique needs a priori knowledge of objects being scanned. The indirect optimization approach has a smaller amount of optimization variables in comparison to direct optimization approach.

This paper describes a novel hybrid optimization heuristic for solving the inverse problem in quantitative MWT, without any approximation, with quantized pixel values. First, the landscape of the objective function is analyzed. The landscape is described by measures used in literature [19] such as: ruggedness, deceptiveness, neutrality, number of local minima. Next, some proposed countermeasures [19] are used to decrease the difficulty of the objective function, which results in hybridization of a global optimization method known as Abstract Bee Colony (ABC) [10] with lo-

cal optimization methods Hooke-Jeeves (HJ) [7] [8] and custom hill-climbing (HC) with memory. Finally, parameters of ABC method are tuned, after experimenting with the MWT objective function. The method is even more interesting because it uses quantized (integer) values are used for search space, instead of real values.

The limitations of the presented method are perfect conditions for the objective function. Noise is not modeled and inversion crime is made.

Chapter 2 describes the problem of optimization in MWT in detail. Method for the landscape analysis and results of analysis are presented in Chapter 3. Chapter 4 describes proposed hybrid optimization algorithm. Performance analysis and comparison of the proposed hybrid algorithm and its variants is given in Chapter 5. Chapter 6 includes final conclusions and future work.

2. PROBLEM DESCRIPTION

This paper deals with the quantitative MWT problem, which is inverse problem, as described in the introduction. The inverse problem is ill-conditioned and multimodal. Global optimization methods are needed to find a solution. Neither approximation nor regularization methods are used. Direct optimization approach is used, where every grid cell, i.e. pixel, is a separate optimization variable.

The cost (objective, fitness) function consists of:

- A search space \mathbb{G} to problem space \mathbb{X} conversion function. A candidate solution needs to be converted from search space variable vector to problem space grid of dielectric permittivities.
- Measurement and simulation forward solvers. The forward solver takes a grid of dielectric permittivities as the input and calculates scattered waves.
- A cost calculation formula. Waves from the measurement and simulation forward solver are compared and the cost of the objective function is calculated.

FDTD [6], a numeric algorithm for simulating electromagnetic wave propagation, is used as a forward solver. Two separate grids for two forward solver are used: a fine and a coarse grid. The fine grid is used by the measurement forward solver for calculating measured waves, which in real MWT would be obtained by measuring waves on antennas. The fine grid measurement forward solver is evaluated only once at the start of inversion. Its input is later used as a target for optimization. The coarse grid simulation forward solver is used by the optimization algorithm, for calculating waves from the candidate solution. The waves are compared in frequency domain with Mean Absolute Error (MAE), a cost calculation formula.

The search space is quantized, i.e. optimization variables are represented with fixed-point numbers. The implementation presented in this paper uses integers to represent fixed-point numbers. The cost function converts the candidate solution from integer search space to floating-point problem space of

dielectric permeability values, which are later used as input for the forward solver.

Because the lack of derivatives, only derivative-free optimization methods are used. Also, cache for the cost function is used, to avoid unnecessary repeated and costly evaluation of the forward solver.

3. LANDSCAPE ANALYSIS

A list of landscape metrics and characteristics, issues in optimization which rise with characteristics occurrence and some solutions for these issues are given in literature [19]. Landscape analysis of objective function is performed in order to obtain metrics such as: ruggedness (bumpiness), deceptiveness (useless gradient information), multimodality (number of local minima), neutrality (slow convergence). The analysis results are used for choosing the right strategy to improve the optimization process.

The problem being solved in this paper has a large amount of variables, so visualization is impossible. Smaller problems objective function could be completely sampled. Note that the search space is quantized, so the sampling could be complete. For larger problems complete sampling is too expensive. Instead, analysis is done on statistic from local minima searches. In this research, Hooke-Jeeves (HJ), a pattern based local search method, is used. Quantum of search space, 1, is set as minimum step for HJ.

As expected from an ill-posed problem, the analysis found many local minima. Also, the landscape is very rigged, which is typical for ill-conditioned problems with weak causality. The most interesting landscape features are valleys, such as the one found in Rosenbrock function. Valleys which are not oriented (stretched) along the coordinate axes are easily found by HJ. HJ easily converge to the valley's floor. Since HJ searches only along the coordinate axes it cannot move towards the next point in the valley's floor, so it halts there notifying that a local minimum is found. Unlike the Rosenbrock function, which have curved valley, valleys found here are straight. This makes tracking valleys easier. Note that a valley's floor is very neutral i.e. it has much smaller gradient in comparison to the valley's walls.

The change of the quantization shows another interesting behaviour. In order to obtain meaningful results, the quantization of the coarse grid has to be finer than the quantization of the the fine grid i.e. the coarse grid has to be quantized in more quantization levels then the fine grid. If the difference of the coarse grid's and the fine grid's number of the quantization levels is below a certain lower threshold, a true minimum cannot be found. This can be considered as a needle-in-a-haystack problem. Beyond the aforementioned lower threshold, a true minimum can be found. As the number of the coarse grid's quantization levels becomes larger, the number of local minima increases. New local minima mostly appear along the valleys. At very fine quantization, above a certain upper threshold, HJ could break through some valleys. Similar behaviour occurs when the Rosenbrock function is optimized by HJ. HJ needs to lower its step to very small values, so it could pass Rosenbrock's valleys, at the cost of very large number of small steps. For example, for one problem, the lower threshold is on 2^6 quantization

levels i.e. 6 quantization bits and the upper threshold is on 2^{14} quantization levels, which is a very big difference.

4. HYBRID OPTIMIZATION

One solution for the ruggedness issue is hybridization [19]. HJ is chosen as a local search algorithm. HJ is a simple algorithm and it is not hard to implement in integer arithmetic. Other possible candidates were Powell method and Nelder-Mead method. Other methods are harder to implement in integer arithmetic. ABC [10], a novel popular meta-heuristic, is used as a global search algorithm. GA is also commonly used in literature [17] [16]. ABC is easier to understand than GA. PSO is harder to implement in integer arithmetic. Also, for inverse scattering problems, ABC is more effective than PSO [15].

Evaluation of the one ABC+HJ hybrid approach [9] shows small acceleration on rugged functions like Ackley, Griewank, Rastrigin. The reason for this is that the aforementioned hybrid approach most of the time explore the search space by ABC and occasionally executes HJ only on the best solution. Since ruggedness drags deceptiveness, ABC work most of the time with false gradient information HJ converges to local minima and convergence to the global minimum is slow. A better approach would be to make the global search algorithm use the results of the local search algorithm [19]. The landscape of the local minima could look less rugged from the global search algorithm's point of view. This way, ABC works only with local minima.

To overcome the issue of valleys not oriented along the axes, mentioned in Chapter 3, a modified version of hill-climbing (HC) algorithm is used. The purpose of HC is to squeeze through the valley towards a better minimum, from a point where HJ stopped. The main feature of modified HC is the ability to check the neighbours in other directions besides along the coordinate axes. Neighbours are in directions defined as having the greatest common denominator equal 1 and first (Manhattan) norm equal 2. For example, neighbour is in direction (1,1) but it is not in direction (2,0), because (2,0) even having first norm equal 2, greatest common denominator is not 1 but 2. A simple version of HC is used, where the algorithm moves from the current position to the position of the first neighbour with a smaller cost. Since HC tracks strait valley, it moves in only few directions. The algorithm memorizes the two most recently used different directions. The algorithm tries to search in memorized directions first, starting from the most recently used direction, before searching in all directions. While tracking through a valley, it is common that two directions occur alternately. This indicates that the valley is oriented to the vector sum of these two directions. The algorithm tries to track the pattern from these two directions, before searching in memorized directions. If a pattern movement is not successful, the valley probably changed the orientation and the two directions from the memory are probably stalled.

Additional landscape analysis shows that HC will merge many local minima found by HJ. HC will lower number of the HJ's local minima by order of magnitude. Also, the probability that the true minimum is hit from the first attempt of local search is much larger when using HJ with HC in comparison to HJ only. Modified HC is more expensive

than HJ when solving same problem. In the worst case HC search $2N_d^2$ directions, while HJ just $2N_d$, where N_d is number of optimization variables. On other side, HC is more effective on lower quantization threshold than HJ on upper quantization threshold, because difference between lower and upper quantization threshold is large, as mentioned in Chapter 3, which amortizes the cost of HC probing in larger number of directions.

The ABC algorithm used in this paper is slightly changed. Instead of terminating the algorithm when a certain number of iterations or function evaluation is reached [12], the algorithm is terminated when the best solution is not changed for certain number of iterations i.e. when the algorithm stagnates for a certain number of iterations. The ABC global search algorithm is hybridized with HJ and HC on following way: every ABC's candidate solution that needs cost function evaluation is improved with HJ first, and then with HC next. That way, the bees in ABC work with local minima only. One feature of this hybrid is that almost all exploitation is done by local optimizations, which causes that the values of ABC parameters proposed in literature [1] [9] are too large. A meta-optimization based method for parameter tuning [1] demands trying multiple parameter candidates and evaluating ABC on a same candidate over 30 or more times to obtain a good mean of cost, which is unacceptably slow. A different approach to parameter tuning is tried. Maximum trial and stagnation thresholds are set to higher values. Every time a better solution is found by an employee or onlooker bee or whenever new scout is sent, the count of trials is logged, before it is restarted. Similarly, when a new best solution is found, the number of iterations under stagnation is saved before restarting. The histograms are made from these logs. Histograms could help choosing lower parameter values than initial one. The number of bees is twice as big as the number of optimization variables. A larger number of bees will lead to more exploitation, which is already done by local search. Large number of bees also means more expensive iterations. On other hand, literature [1] neglects the impact of the bee number on the performance of the optimization.

5. EXPERIMENTAL VALIDATION

Table 1 presents a comparison of optimization algorithms. A 3-cell 1D problem is optimized over three case: 2^6 , 2^8 and 2^{14} quantization levels. Every algorithm is tuned to a 100% success rate on the simplest problem with 2^6 quantization levels, in order to have a fair comparison. The first two rows show the maximum number of trial iterations $N_{max.trials}$ and the maximum number of stagnation iterations $N_{max.stagn}$. Number of bees N_{bees} is kept as double the number of optimization variables. Every cell shows cost in function evaluation and success rate, averaged over 50 algorithm runs.

As seen in Table 1, ABC method hybridized with HJ and modified HC with memory, gives the best performance. It also seen that finer quantization demands more function evaluations and degrades success rate. At upper threshold of 2^{14} quantization levels success rates are better. The reason for that is the ability of HJ to break through more valleys, as described in Chapter 3.

Table 1: Comparison of optimization heuristics

Heuristics	ABC	ABC+HJ	ABC+HJ+HC
N_{max_trials}	150	30	12
N_{max_stagn}	2000	60	14
2^6	121 100%	100 100%	79 100%
2^8	1890 50%	1420 80%	669 98%
2^{14}	10030 0%	33141 94%	2044 100%

6. CONCLUSIONS

This paper shows that landscape analysis could help in hybridization, modification and tuning of optimization heuristic. It is important to consider valley structures, when optimizing the MWT objective function landscape. Choosing the right level of quantization is important to perform successful search. If quantization is too coarse, then a true global minimum will never be found. Finer quantization will decrease performance.

One of the future tasks is examining objective function separability. If the objective function is separable, additional acceleration could be obtained [3]. An additional future task is implementing an adaptive local search method [11] which promises faster convergence, especially in valleys. Also, the proposed optimization heuristic should be tested on a more realistic forward solver, using realistic phantoms with human tissues [18].

7. ACKNOWLEDGMENTS

The authors want to thank God for all these great ideas and all people who helped.

This work was partially supported by the Ministry of Education, Science and Technological Development of the Republic of Serbia, under grant number: TR32029.

8. REFERENCES

- [1] B. Akay and D. Karaboga. Parameter tuning for the artificial bee colony algorithm. In *International Conference on Computational Collective Intelligence*, pages 608–619. Springer, 2009.
- [2] A. Ashtari, S. Noghianian, A. Sabouni, J. Aronsson, G. Thomas, and S. Pistorius. Using a priori information for regularization in breast microwave image reconstruction. *IEEE Transactions on Biomedical Engineering*, 57(9):2197–2208, 2010.
- [3] W. Chen, T. Weise, Z. Yang, and K. Tang. Large-scale global optimization using cooperative coevolution with variable interaction learning. In *International Conference on Parallel Problem Solving from Nature*, pages 300–309. Springer, 2010.
- [4] M. Donelli, I. J. Craddock, D. Gibbins, and M. Sarafianou. A three-dimensional time domain microwave imaging method for breast cancer detection based on an evolutionary algorithm. *Progress In Electromagnetics Research M*, 18:179–195, 2011.
- [5] S. Gabriel, R. Lau, and C. Gabriel. The dielectric properties of biological tissues: Ii. measurements in the frequency range 10 hz to 20 ghz. *Physics in medicine and biology*, 41(11):2251, 1996.
- [6] S. Hagness and A. Taflove. *Computational electrodynamics: the finite-difference time-domain method*. Norwood, MA: Artech House, 2000.
- [7] R. Hooke and T. A. Jeeves. Direct search solution of numerical and statistical problems. *Journal of the ACM (JACM)*, 8(2):212–229, 1961.
- [8] M. G. Johnson. Nonlinear optimization using the algorithm of hooke and jeeves, 1994.
- [9] F. Kang, J. Li, Z. Ma, and H. Li. Artificial bee colony algorithm with local search for numerical optimization. *Journal of Software*, 6(3):490–497, 2011.
- [10] D. Karaboga and B. Basturk. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. *Journal of global optimization*, 39(3):459–471, 2007.
- [11] I. Loshchilov, M. Schoenauer, and M. Sebag. Adaptive coordinate descent. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, pages 885–892. ACM, 2011.
- [12] M. Mernik, S.-H. Liu, D. Karaboga, and M. Črepinšek. On clarifying misconceptions when comparing variants of the artificial bee colony algorithm by offering a new implementation. *Information Sciences*, 291:115–127, 2015.
- [13] S. Noghianian, A. Sabouni, T. Desell, and A. Ashtari. *Microwave Tomography: Global Optimization, Parallelization and Performance Evaluation*. Springer Publishing Company, Incorporated, 2014.
- [14] M. Ostadrahimi, P. Mojabi, A. Zakaria, J. LoVetri, and L. Shafai. Enhancement of gauss–newton inversion method for biological tissue imaging. *Microwave Theory and Techniques, IEEE Transactions on*, 61(9):3424–3434, 2013.
- [15] A. Randazzo. Swarm optimization methods in microwave imaging. *International Journal of Microwave Science and Technology*, 2012, 2012.
- [16] A. Sabouni and S. Noghianian. Experimental results for microwave tomography imaging based on fdtd and ga. *Progress In Electromagnetics Research M*, 33:69–82, 2013.
- [17] A. Sabouni, S. Noghianian, and S. Pistorius. A global optimization technique for microwave imaging of the inhomogeneous and dispersive breast. *Electrical and Computer Engineering, Canadian Journal of*, 35(1):15–24, 2010.
- [18] J. D. Shea, P. Kosmas, S. C. Hagness, and B. D. Van Veen. Three-dimensional microwave imaging of realistic numerical breast phantoms via a multiple-frequency inverse scattering technique. *Medical physics*, 37(8):4210–4226, 2010.
- [19] T. Weise, R. Chiong, and K. Tang. Evolutionary optimization: Pitfalls and booby traps. *Journal of Computer Science and Technology*, 27(5):907–936, 2012.
- [20] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 1997.

Schedule assignment for vehicles in inter-city bus transportation over a planning period

Balázs Dávid University of Szeged
davidb@jgypk.u-szeged.hu

ABSTRACT

In this paper, we examine the problem of assigning vehicles to each day of a planning period based on existing theoretical schedules in public transportation. The assignment of a vehicle to daily tasks has to satisfy certain requirements. If the problem addresses long-distance bus transportation, vehicles returning to their starting depots would usually result in a high additional cost. Because of this, we also have to assign a garage to each vehicle where they spend the night and from where they start their next daily schedule. We also want to minimize the arising traveling and operational costs. We give a network-based mathematical model for the problem. We examine solutions both of the model and of heuristic methods, and present their results.

Categories and Subject Descriptors

J.m [Computer Applications]: Miscellaneous

General Terms

Vehicle scheduling, Application, Heuristic

1. INTRODUCTION

Public transportation companies usually create their schedule in advance for a longer planning period. The days of this period belong to different day-types (workdays, holidays, etc.). Days that share a day-type have the same underlying theoretical schedule. Such a daily schedule divides the set of trips into vehicle duties, which also give the execution order of tasks in that duty. These duties are carried out by the vehicles of the company each day.

If several days share the same day-type, the same vehicle duty will exist for all of them, and a duty will always be executed by a single vehicle. However, it does not necessarily have to be the same vehicle every day for the same duty. The goal of this paper is to assign the above given duties to vehicles, thus creating a unique roster for each vehicle over the desired planning period.

The outline of this paper is the following: first, we present the classic problem of vehicle scheduling, and demonstrate the concept of vehicle duties through it. Using this, we define the schedule assignment problem, where we aim to organize the duties of vehicles over a longer planning period. For this problem, we give a mathematical model, and also present a matching based heuristic. The solution of both the model and heuristic are tested on real-life instances.

2. VEHICLE SCHEDULING

For the introduction of the VSP, we refer to our formalization in [5]. We are given a set V of vehicles and T of service trips. Every trip has a departure and arrival time, a starting and ending location, and a set of vehicles that are able to serve the trip. A (t, t') pair of trips are compatible, if a vehicle can service both trips with respect to the running time and distance between the arrival location of t and the departure location of t' (such a journey is called a dead-head trip). A set D of depots can also be introduced for the problem. In this case, every $v \in V$ vehicle has a depot-type $d(v) \in D$. Vehicles that share the same depot-type share the same characteristics, and also have the same costs. If a vehicle v belonging to depot d is used in the solution, it contributes a cost of $dc(d) + tc(d) \times dist(v)$, where $dc(d)$ is a one-time daily cost, and $tc(d)$ is the cost of traveling a unit distance for a vehicle belonging to depot d , while $dist(v)$ is the distance covered by vehicle v in the solution. A binary depot-compatibility vector $\mathbf{v}^t = (v_1, \dots, v_{|D|})$ can also be introduced for every trip $t \in T$. If such a vector exists, a vehicle belonging to depot d can only service trip t , if $v_d^t = 1$. The VSP assigns the trips of the given timetable to vehicles, satisfying the following conditions: for every $v \in V$, the trips assigned to v must be compatible with each other, and every trip $t \in T$ must be executed exactly once. The cost of this assignment has to be minimal.

If the problem has only 1 depot, it is called a single depot vehicle scheduling problem (SDVSP), and can be solved in polynomial time. A formulation for the SDVSP can be seen in [2]. If the number of depots is at least 2, we get a multiple depot vehicle scheduling problem (MDVSP). The MDVSP was introduced by Bodin et al. in [3], and proven to be NP-hard by Bertossi et al. [1]. An overview of different VSP models can be found in [4].

The result given by the above VSP corresponds to a set of vehicle duties for one day. A vehicle duty gives a set of tasks that have to be executed by the same vehicle on the given

day. However, the VSP does not assign a specific vehicle to its duties, only gives the required vehicle type. Because of this, we call the result a "theoretical" schedule, as further steps have to be taken to determine the exact vehicles in service on the current day.

3. SCHEDULE ASSIGNMENT PROBLEM

As seen in Section 2, the resulting schedules of the VSP only give vehicle duties for a single day. However, transportation companies create their schedules in advance for a planning period (eg. several weeks or months). Their usual method is to separate the days of the planning period into different types (eg. workday, Saturday, holiday, etc.), and have a theoretical vehicle schedule for each of these day-types. This means that days belonging to the same day-type will have the exact same vehicle duties throughout the entire planning period. Same duties will always have the same requirement for vehicle types over the planning period. However, they will not necessarily be executed by the same vehicle on different days.

The input for the schedule assignment problem is the n day planning period of the company, with each day i having an assigned day-type $dt(i)$. We are also given the set V of vehicles that are available over the planning period. Similarly to the VSP, a set D of depots is also introduced, and every $v \in V$ vehicle is given a depot-type $d(v) \in D$. Similarly to the VSP, vehicles belonging to the same depot share the same costs and characteristics. Set G represents garages where vehicles can stay for the night between two days of the planning period. For each day-type dt we also have a daily vehicle schedule, which is the set $S(dt)$ of vehicle duties that have to be executed. Similarly to the trips of the VSP, a vehicle duty $j \in S(dt)$ also has a binary depot-compatibility vector $\mathbf{v}^j = (v_1, \dots, v_{|D|})$. A vehicle from depot d can service duty j if and only if $v_d^j = 1$. Vehicles in inter-city transportation do not necessarily return to their starting garages after executing a duty, as that could potentially mean high extra costs depending on the distance they have to travel. Because of this, a garage $g \in G$ also has to be assigned to the vehicle at the end of each day, where they will spend the night and begin the next day of the planning period. The goal of our problem is to assign these duties to the vehicles of the company such that each duty is executed exactly once, and the arising costs are minimal. A vehicle v from depot d contributes $dc(d) \times work_i^v + tc(d) \times dist(v)$ to cost of the problem, where dcd and $tc(d)$ are the one-time daily and unit-distance costs of a vehicle from depot d respectively, $dist(v)$ is the distance travelled by vehicle v during the planning period (either by servicing duties or traveling to/from garages). The binary vector $\mathbf{work}^v = (work_1, \dots, work_n)$ denotes whether vehicle v was working on day i of the planning period, or not.

3.1 Model

In this subsection we will introduce an integer programming model for the schedule assignment problem over a planning period where vehicles have to spend the night at one of the pre-assigned garages. Some notations in this section will be different from the problem introduction above. Let us consider a planning period of n days. Let D be the set of nodes for the vehicle schedules for the planning period. Let

$d_{i,j} \in D$ be the vehicle duty j on day i , where $1 \leq i \leq n$ and $1 \leq j \leq k$ where k is the number of duties on day i . Let G be the set of nodes for the l garages. A garage g has to be considered as a potential night garage for vehicles at the end of every day of the planning period. To denote this, we introduce multiple nodes for each garage. Let $g_{i,j}$ represent a state of garage j on the end of day i , where $0 \leq i \leq n$ and $1 \leq j \leq l$. This node will represent the number of vehicles staying at garage j at the end of day i . The special node $g_{0,j}$ denotes the state of garage j at the beginning of the planning period. Let V be the set of nodes for the m vehicles. We represent vehicle i with two nodes: $v_{i,0}$ represents the vehicle at the beginning of the planning period and $v_{i,1}$ at the end of the planning period. The edges of our network will represent the possible traveling activities of vehicles throughout the planning period. Vehicles staying in their starting garage at the beginning of the planning period are given by edges

$$E^{vb} = \{(v_{i,0}, g_{0,j}) | 1 \leq i \leq m, \text{ vehicle } i \text{ starts at garage } j\}.$$

Vehicles ending the planning period in one of the garages are represented by

$$E^{ve} = \{(g_{n,i}, v_{j,1}) | 1 \leq i \leq l, 1 \leq j \leq m\}.$$

Vehicles leaving the garages to execute a duty at the beginning of a day are represented by edges

$$E^{db} = \{(g_{i-1,j}, d_{i,h}) | 1 \leq i \leq n, 1 \leq j \leq l, 1 \leq h \leq k\}.$$

Vehicles returning to a garage at the end of the day from a duty are represented by edges

$$E^{de} = \{(d_{i,h}, g_{i,j}) | 1 \leq i \leq n, 1 \leq j \leq l, 1 \leq h \leq k\}.$$

Vehicles staying at a garage for a given day are represented by edges

$$E^g = \{(g_{i-1,j}, g_{i,j}) | 1 \leq i \leq n, 1 \leq j \leq l\}.$$

Circulation edges should also be added for each vehicle:

$$E^f = \{(v_{i,1}, v_{i,0}) | 1 \leq i \leq m\}.$$

Using the node set $N = \{D \cup V \cup G\}$ and edge set $E = \{E^{vb} \cup E^{ve} \cup E^{db} \cup E^{de} \cup E^g \cup E^f\}$ we can define the multi-commodity network (N, E) . Our network will have m separate commodities, one for every vehicle. The commodities of this network will be denoted by $c \in C$. For each edge e of this network, we give an integer vector x_e . This vector will have one component for every commodity c , which we will denote by x_e^c . The value x_e^c represents if a vehicle c is

assigned the traveling activity connected to edge e . Based on the above network, we can formalize the mathematical model in the following way:

$$\sum_{e:(g_{i-1,j},d_{i,h}) \in E^{db}} x_e^c = 1, \forall (i,h) \text{ pair} \quad (1)$$

$$\sum_{e:(d_{i,h},g_{i,j}) \in E^{de}} x_e^c = 1, \forall (i,h) \text{ pair} \quad (2)$$

$$x_e = 1, \forall e \in E^{vb} \quad (3)$$

$$\sum_{e:(g_{n,i},v_{j,1}) \in E^{ve}} x_e^c = 1, \forall c \in C \quad (4)$$

$$\sum_{e \in n^+} x_e^c - \sum_{e \in n^-} x_e^c = 0, \forall c \in C, \forall n \in N \quad (5)$$

$$x_e^c \in \{0, 1\}, \forall e \in \{E^{vb} \cup E^{ve} \cup E^{db} \cup E^{de} \cup E^f\} \quad (6)$$

$$x_e^c \geq 0 \text{ integer}, \forall e \in E^g \quad (7)$$

$$\sum_{c \in C} \sum_{e \in E} tr_e^c x_e^c \rightarrow \min$$

Constraints (1) and (2) restrict that there should be exactly one vehicle executing a duty, and returning from a duty to a garage. Constraint (3) does the starting setup for each vehicle, assigning them to a garage given by the network. Constraint (4) ensures that every vehicle ends the planning period in exactly one garage. Flow conservation for the vertices of the network is guaranteed by (5), while constraints (6) and (7) provide the binary and integrality constraints for all the variables. The objective function of the model minimizes the arising costs of the executed traveling activities. The value tr_e^c gives the cost of a vehicle from commodity c to service the activity denoted by edge e .

3.2 Extensions of the model

Depending on the requirements and problem size, other constraints can also be added to the model. One of the easiest ways to decrease the problem size is to modify constraint (3). In its current form, there is a separate commodity for each vehicle available for the planning period, which can result in a large graph even for a small number of vehicles. However, vehicles that have the exact same requirements can be classified into groups. If we let V be the set of such vehicle groups, and k_i be the number of vehicles in group $i \in V$, then the following constraint can replace (3):

$$\sum_{j:v_{i,0},g_{0,j}} x_v \leq k_i, \forall i \in V \quad (8)$$

If garages $i \in G$ have a limited capacity m_i , then we have to introduce the following capacity constraint on all of their incoming edges:

$$\sum_{e:d_{i,h},g_{i,j}} x_e \leq m_i, \forall j \in G \quad (9)$$

If the vehicles also have to be refueled at the end of every day, we have to modify the underlying graph, and introduce the set T for refueling stations. To assign a refueling station for every vehicle at the end of a day, we have to introduce two new sets of edges to the model instead of E^{de} . Vehicles heading towards a refueling station at the end of a day from a duty are represented by edges

$$E^{rb} = \{(d_{i,h},t_{i,j}) | 1 \leq i \leq n, 1 \leq j \leq |T|, 1 \leq h \leq k\},$$

and vehicles returning to a garage after refueling are represented by edges

$$E^{re} = \{(t_{i,j},g_{i,j}) | 1 \leq i \leq n, 1 \leq j \leq |T|\},$$

where $t_{i,j}$ represents a state of refueling station j on the end of day i . In this case, constraint (2) is replaced with the following:

$$\sum_{e:(d_{i,h},t_{i,j}) \in E^{rb}} x_e^c = 1, \forall (i,h) \text{ pair} \quad (10)$$

3.3 A matching heuristic

We also present a heuristic solution for the above problem. Given a planning period of n days, this method will sequentially examine all (d_k, d_{k+1}) day pairs ($0 \leq k \leq n-1$) over the planning period. For each such pair, a bipartite graph $G_k = (V_k \cup D_k, E_k)$ is constructed. The graph G_k represents the state of the problem at the beginning of day k . The special value $k=0$ denotes the beginning of the planning period. Let G be the set of garages where vehicles can stay for the night.

Nodes $v \in V_k$ represents the vehicles from the fleet of the company with a status at the end of day k , while nodes $d \in D_k$ represent the vehicle duties of day $k+1$. An edge (v,d) exists in E_k , if the vehicle v is able to execute duty d . The cost of an edge is based on the minimum of the following distances: $s_{v,d} = \min\{s_{v,g} + s_{g,d}\}$, where $s_{v,g}$ is the distance between the location of vehicle v at the end of day k and garage g , while $s_{g,d}$ is the distance between garage g and the starting location of duty d for all $g \in G$ (the smaller the distance, the bigger this value will be). Based on the above graph, we can give the following matching model for our problem (for a big enough number N). The binary variable $x_{v,d}$ represents if duty d is executed by vehicle v in the solution, or not.

$$\sum_{(v,d) \in E} x_{v,d} = 1, \forall i \quad (11)$$

$$\sum_{(v,d) \in E} x_{v,d} = 1, \forall j \quad (12)$$

$$c_{v,d} = \min\{N - s_{v,d}\}, \forall (v,d) \in E \quad (13)$$

$$x_{v,d} \in \{0, 1\}, \forall v, d \quad (14)$$

$$\sum_{(v,d) \in E} c_{v,d} x_{v,d} \rightarrow \max$$

We sequentially solve n such matching models for all day pairs (d_k, d_{k+1}) , $(0 \leq k \leq n - 1)$, which will give us the schedule assignment for all vehicles over the planning period. After solving the last matching problem, the position of the vehicles will be the ending location of their last duty. Because of this, we need to solve a final matching problem, which sends every vehicle to the closest garage.

4. TEST RESULTS

We tested the model and heuristic solution on real-life instances. These instances were part of a "what-if" scenario, trying to coordinate the transportation of three counties in Hungary. These counties organized their transportation semi-independently before. The transportation companies provided the input for a 3-month long planning period. The input consisted of vehicle duties belonging to 4 day-types. A single day had 90-170 vehicle duties depending on its type, and the combined fleet of the companies was separated into 3 vehicle types. One vehicle type was able to execute any of the duties, while the other two vehicle types were restricted to some of the duties (eg. depending on the length of the duty). Using the input data above, we created two main groups of test instances: one with all three vehicle types, and another with the restricted vehicle types merged into one. We ran tests for the entire planning period of 3 months and smaller intervals of it also. The mathematical model was solved using the COIN-OR Symphony MILP solver. The results can be seen in Table 1.

Table 1: Results of the mathematical model

Vehicle types	Planning period	Opt. cost(km)	Opt. time(s)
2	1 week	1 665.10	14.06
	1 month	12 219.50	221.23
	2 months	31 993.40	758.09
	3 months	48 800.00	1 813.57
3	1 week	2 123.40	14.00
	1 month	21 221.20	228.26
	2 months	56 597.40	847.86
	3 months	88 933.30	2 008.83

This table shows both the cost of the instances (measured in the km that the vehicles ran during the planning period) and the time in seconds required for the solution. The results of the heuristic method can be seen in Table 2.

It can be seen from the tables, that the solution of the model is possible even for larger instances. This means that it can be applied for practical problems, especially because the constraints of the model can easily be modified depending on the requirements of the given company. The heuristic performed poorly on small instances due to the large number of sequential matching problems it has to solve, but manages to significantly decrease the running time of bigger instances. The quality of its solutions is far from the optimal value, but the gap gets smaller as the length of the planning period increases. However, one of the main reasons behind developing

Table 2: Results of the heuristic

Vehicle types	Planning period	Heur. cost(km)	Heur. time(s)	Heur. gap(%)
2	1 week	2 824.30	26.53	69.62
	1 month	16 061.400	86.06	31.44
	2 months	37 266.90	168.02	16.48
	3 months	55 717.10	232.90	14.17
3	1 week	5 463.00	24.38	157.28
	1 month	31 768.00	86.64	49.69
	2 months	70 661.90	154.78	24.85
	3 months	106 551.10	227.98	19.81

the heuristic was the ability to generate an adequate solution for the problem in a short time, in case we want to use an initial solution for larger instances with a mathematical programming based solution process. The heuristic fits this requirement well.

5. CONCLUSIONS

In this paper, we examined the application oriented problem of assigning schedules to vehicles over a planning period. We wanted these assignments to take into consideration the requirements of the vehicles themselves, and provide more information than a "theoretical" solution this way. For this, we introduced the schedule assignment problem, and provided a mathematical model for it. The basic model considers the parking requirements of vehicles at the end of each day, but we also gave extensions for garage capacities and refueling at the end of each day. We also devised a matching-based sequential heuristic for the problem, which decreases the running time significantly, but comes at the cost of being far from the optimal solution in quality.

A future extension of the model will include the requirement of regular mechanical inspection: vehicles have to be sent for a daily inspection after executing duties for a given amount of days. For this, we considered a state-expanded version of the current model, but the size of its current version is still too large to yield a solution yet. One natural way to handle such a large problem is column generation, which needs an initial solution with an acceptably quality. The matching heuristic solves the problem quickly, and its results can be applied effectively in such a solution process.

6. REFERENCES

- [1] A. Bertossi, P. Carraraesi, and G. Gallo. On some matching problems arising in vehicle scheduling models. *Networks*, 17(1):271–281, 1987.
- [2] L. Bodin and B. Golden. Classification in vehicle routing and scheduling. *Networks*, 11(1):97–108, 1981.
- [3] L. Bodin, B. Golden, A. Assad, and M. Ball. Routing and scheduling of vehicles and crews: The state of the art. *Computers and Operations Research*, 10(1):63–212, 1983.
- [4] S. Bunte and N. Klierer. An overview on vehicle scheduling models. *Journal of Public Transport*, 1(4):299–317, 2009.
- [5] B. Dávid and M. Krész. A model and fast heuristics for the multiple depot bus rescheduling problem. In *10th International Conference on the Practice and Theory of Automated Timetabling (PATAT)*, pages 128–141, 2014.

A self-bounding Branch & Bound procedure for truck routing and scheduling

[Extended Abstract]

Csongor Gy. Csehi*

Márk Farkas*

Ádám Tóth*

ABSTRACT

In this talk we will study a part of the core algorithm of a complex software solution for truck itinerary construction for one of the largest public road transportation companies in the EU. The problem is to construct a cost optimal itinerary, given an initial location with an asset state, the place and other properties of tasks to be performed. Such an itinerary specifies the location and activity of the truck and the driver until the finish of the last routing task. The calculation of possible itineraries is a branch and bound algorithm. The nodes of the search tree have the following arguments: position, time, driver-state and truck-state. For each node we calculate the cumulated cost for the road reaching that state, and a heuristically lower bound for the cost of the remaining road. In each step the procedure expands the next unexpanded node with the best sum for cumulated and heuristically cost. To make a sharp heuristic we run the same branch and bound algorithm (from each node) but with hypothetical positions (with coarser data and simplified activities: no refuelling, no road costs, etc.). We anticipate significant gains in performance and quality compared to the previous approach.

CCS Concepts

•Computer systems organization → Embedded systems; Redundancy; Robotics; •Networks → Network reliability;

Keywords

Logistics; route optimization; branch and bound

1. INTRODUCTION

We will study a part of the core algorithm of a complex software solution for truck itinerary construction for one of

¹Nexogen Kft., csehi.csongor@nexogen.hu, farkas.mark@nexogen.hu, toth.adam@nexogen.hu

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

the largest public road transportation companies in the EU. A minor improvement on the operational cost of each tour can result huge advantage for the freight services company. The problem is to construct a cost optimal itinerary, given an initial location with an asset state, the place and other properties of tasks (we will call them routing tasks) to be performed. Such an itinerary specifies the location and activity of the truck and the driver until the finish of the last routing task. This means that this itinerary gives every order to the driver, including every turn in the road and every stops with exact durations, etc. The working stops can be done only in the places of the tasks, the refueling and resting stops can be done only in previously fixed places (roughly 4000 fixed parking places and 100 fixed filling stations across Europe). To achieve such an itinerary we use mapping software to construct the routes and calculate the distance, duration and cost between any two places. Clearly the problem is much harder than a path finding in the graph, because we can do many different actions in each place (different amount of fueling liters, different duration of rest, etc).

The software (which also performs the vehicle assignment) is already finished and applied with very good results (from 2015), large cost saving is reached by the company. For more formal definitions of the problem, and more information of the software one must read [2]. The ongoing researches aim to extend the functionality of the software. One goal is to improve optimality by plan the itinerary for longer timespan. That means more routing tasks in each round.

The calculation of possible itineraries is a branch and bound algorithm. For detailed information on the widely used algorithms of operations research the reader should see [1] The nodes of the search tree have the following arguments: position, time, driver-state and truck-state (we will call these data the state). For each node we calculate the cumulated cost for the road reaching that state, and a heuristical lower bound for the cost of the remaining road. Each node has a pointer to its father (this will make it possible to calculate the route from a proper node). In each step the procedure expands the next unexpanded node with the best sum for cumulated and heuristical cost.

The following oversimplified example of [2] with Figure 1 illustrates the tree of the algorithm. Suppose that we are in position 'Start' in the beginning. From 'Start' we can go to different places for example two parking places 'P1' and 'P2' (the state will be different in the two locations if the duration and distance of the drivings are not equal). Supposing that we can rest 9 or 11 hours we get two new nodes from each parking place reaching node. If we can reach place 'P3'

from both 'P1' and 'P2' then this way we get four different nodes in the same place 'P3'. In general none of the four nodes can be bounded in the algorithm, because the states are different and hence we can not predict which will give the best solution in the end.

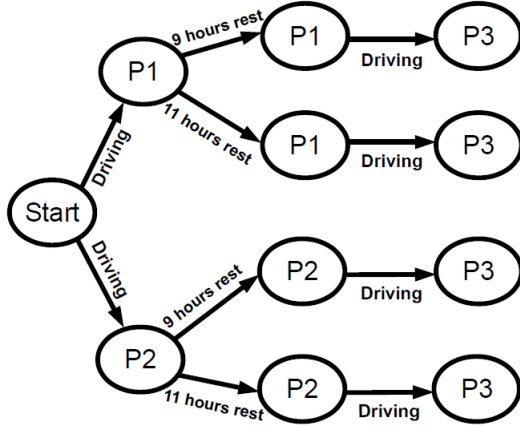


Figure 1: An example subtree of the algorithm

The better the lower bounds are, the less nodes need to be expanded. However, it is always more time-consuming to make better estimations.

The difficulty with the heuristics is the relation of the state of the driver and the opening times of the routing tasks. Both would be easier to handle separately but together it gives an NP-hard problem.

The main steps of the algorithm are the following:

1. Create the starting node of the tree from the initial state and position of the driver. Put it in an empty list L .
2. While L has any element:
 - (a) Pick X from L with the best TotalCost value.
 - (b) If the itinerary given by X is a complete tour (finishing all the routing tasks), then RETURN X .
 - (c) Select the best possible activities (set A) to do from X .
 - (d) BRANCHING: For each element of A create the node (set N) which represents the state and position after that activity.
 - (e) For each element of N calculate the cumulated cost (we can get it by adding the cost of the activity to the cumulated cost of X).
 - (f) For each element of N calculate the heuristical cost (this step will be examined in detail in the next sections).
 - (g) For each element of N compare the lower bound for the reaching time with the limitations (we get the lower bound during the calculation of the heuristics). If the node can not reach the target in time than delete it from N .

- (h) BOUNDING: For each element Y of N , where the place of Y is P , get the list L_P of the previously examined nodes in place P . Compare Y with every element of L_P , and if there exists such Z that every state related variable and the cost are not worse in Z than in Y , then delete Y from N .
- (i) For each element of N put it into L and into the proper L_{P_i} list according to the place of the nodes.

3. RETURN: Unreachable target. The target can not be reached in the given time limit.

The algorithm has many additional logics, but here we focus on the heuristics only. A more detailed description of the algorithm can be found in [2].

2. THE CONCEPT OF THE MAIN BOUNDING METHOD

As we mentioned in the step 2/f of the algorithm we need a good heuristic to bound the remaining cost from every node. For this we need to calculate a minimal road and we have to bound the needed duration.

First we estimate the remaining distance and driving duration. To optimize the running time we do not want to ask the mapping software for all these estimations, but store as much of the possibly needed information as we can. We construct two graphs where the nodes are the possible places of the tours (parking places, filling stations, etc.) and the length of the edges are the minimal distances and driving durations. From these graphs we generate the minimal distances and durations between each pair of nodes with the Floyd-Warshall algorithm (this is a precalculation before the itinerary generator algorithm mentioned before). It is a whole separated topic how we handle the truck positions and places of the routing tasks (since they are not permanent, hence they are not contained in the graphs).

After we have a lower bound for the remaining driving time we estimate the total time needed by constructing a hypothetical itinerary. We suppose that the driver can drive the maximal amount what he can, each time, and then reaches a parking place. In each parking place he rests the minimal amount what is needed and then go further. When he reaches a routing task sometimes he has to wait for the time-window. However, supposing that there are no time-windows the heuristics can be calculated in linear time (we will call it linear heuristic).

On the other hand, if we think about how to include the time-windows in the linear heuristic we face a problem. Namely, sometimes it would be better to rest more, not just the minimal needed amount before making the task. The following example highlights that behavior.

Suppose that the driver arrives at 6 a clock, after 9 hours of driving, but the routing task opens at 10 a clock. To finish the routing task, the driver has to work 1 hour there and we have one more routing task which is 2 hours far from this. When will we finish the last routing task?

1. If we wait for the first opening and work 1 hour, then we cannot drive further because of the daily driving time limit (9 hours). That way we have to rest at least 9 hours. After the rest we can drive to the next routing task and finish it until **23 a clock**.
2. If we rest 9 hours instead of the 4 hours waiting then

we can start the work with a fresh state, drive to the next routing task and finish it until **19 a clock**.

The above example shows that we can not make good lower bounds with such a concept (as in the linear heuristic) if we try to optimize with the driver-state and the time-windows at a time. However, to obtain better estimations for the branch and bound procedure we must include the time-windows in the heuristics. For the best fit (between the heuristics and the algorithm) we apply almost the same logics to calculate a lower bound for the duration as we use in the branch and bound algorithm itself.

3. THE SELF-BOUNDING BRANCH AND BOUND ALGORITHM

As we mentioned before the main branch and bound algorithm works on nodes with position, time, driver-state and truck-state. The positions are real places on the map. To make a sharp heuristic in step 2/f of the algorithm (we will call it B&B heuristic) we run the same branch and bound algorithm (from each node) but with hypothetical positions (with coarser data and simplified activities: no refuelling, no road costs, etc.). This means that we generate those positions which was used by the linear heuristic and let the different cases compete in total duration. The best solution will give the B&B heuristic which will be the lower bound for the remaining cost of the node in the main branch and bound algorithm.

Observe that the B&B heuristic needs a heuristically lower bound too. For this we can use the linear heuristic.

It is easy to see that this extended procedure can give much better lower bounds for the main branch and bound algorithm, but it is in question that if it is worth the extra time consumed during the construction of the nodes (calculating their heuristic values). Observe that it is more likely to get better heuristics this way if we have more routing tasks (with time-windows).

4. RESULTS

We evaluated the differences using a sample pack of 4400 itineraries, containing 2 – 3 routing tasks in average. The original branch and bound procedure (with the linear heuristic) expands about $2.4 * 10^4$ nodes during an itinerary construction. The total time of the algorithm was about $3.2 * 10^3$ minutes, but we use about 100 parallel machines. Hence, it runs in about 30 minutes to construct the 4400 itineraries. The heuristics was calculated in about $2.7 * 10^6$ ms in total. There was 67 cases where the algorithm did not give a solution because of the running time limit.

The new branch and bound procedure (with the B&B heuristic) expands about $2 * 10^4$ nodes in the same sample pack of itineraries, but each node creation needs more time. The heuristics was calculated in about $1.4 * 10^8$ ms in total (50 times more than the original). Fortunately the total time of the algorithm was about $7.6 * 10^3$ minutes, which is just twice the original. That way it is not yet better than the original algorithm with this size. However, since the better lower bound thin the searching tree, it is an exponential reduction in the number of routing tasks, which means that this solution will be better for longer itineraries. We are not capable to make statistics for more than 10 routing tasks in one plan yet.

On the other hand the new algorithm failed to give a solu-

tion only in 35 cases because of the running time limit. We hope that with some modifications to make the heuristic branch and bound faster, we can calculate longer itineraries which can not be calculated with the original heuristics. If we reach a proper running time with the new branch and bound heuristics, we will try to give more tasks for the planning (now it is about 2 – 3 routing tasks in average). It is anticipated that with this method we can plan two times longer itineraries.

5. ACKNOWLEDGMENTS

Research is partially supported by grant No. OTKA 108947 of the Hungarian Scientific Research Fund.

6. REFERENCES

- [1] M. W. Carter, C. C. Price (2000) Operations research: a practical introduction, Crc Press.
- [2] Cs. Gy. Csehi, M. Farkas, (2016) Truck routing and scheduling, Central European Journal of Operations Research, 1-17. doi: 10.1007/s10100-016-0453-8

Improving flow lines by unbalance

Z. Mihály
Optasoft Kft.
1051 Sas utca 10.
Budapest, Hungary
zsolt.mihaly@optasoft.hu

Z. Lelkes^{*}
Pallasz Athéné University
Department of Information Technology
6000 Izsáki út 10.
Kecskemét, Hungary
lelkes.zoltan@gamf.kefo.hu

ABSTRACT

The paper's aim is to provide some insight regarding the performance of balanced and unbalanced discrete manufacturing flow lines. The investigation is based on physical simulation systems. The performance characteristics are gathered with a discrete time simulation program using next-event time advance mechanism. The model has been implemented in AIMMS modelling language.

Categories and Subject Descriptors

H.4.2 [Information Systems Applications]: Types of Systems—*Logistics*; G.3 [Probability and Statistics]: Stochastic processes; J.6 [Computer-Aided Engineering]: Computer-aided manufacturing

General Terms

Management

Keywords

flow line, discrete time simulation, factory physics, throughput, cycle time

1. INTRODUCTION

Discrete manufacturing systems can be classified by several disciplines. Following Govil and Fu [4], the manufacturing systems can be job shops, **flow lines**, flexible manufacturing systems or assembly systems. The research of manufacturing systems uses diverse modelling techniques, e.g., simulation models [13], queueing theory and Petri nets [9].

Companies make great efforts to diminish their ecological footprint, which is highly connected to supply chains. Interest in researches on environmentally benign business practices has been continuously increasing. It is necessary to adopt some of these techniques in order to sustain a green supply chain [12]. The study of flow lines is important as

they are frequently parts of complex supply chains. Manufacture of various products, such as cars, pharmaceutical ingredients and electrical goods are only a few instances where flow lines can be used for modelling.

In this paper, flow lines are investigated using physical experiments and discrete time simulation model. Some examples from the literature contain investigations into flow line with common buffer [16], complex optimization problems where the flow line is only one element in the model [8] or more complicated systems. Huang and Li examined a two-stage hybrid flow shop with multiple product families [7]. Simulation modelling has a wide range of applications in engineering-aided manufacturing regarding system performance. Modelling apparel assembly cells [1], a Mercedes-Benz production facility [10], or analyzing the performance of a Korean motor factory [2] are only some of the examples.

Hopp and Spearman [6] have introduced the concept of factory physics consisting of useful theories. The type of material flows that they investigated is the flow line in which there is only one machine per station, one job class, and no capacity constraint.

Three main modelling measures are proposed by Hopp and Spearman:

- Throughput (TH): the number of entities (cars, apples, people, etc...) coming out from the system during a given time
- Cycle time (CT): the time an entity spends in the system
- Work-in-process (WIP): the number of entities residing in the system at the same time

The higher TH and lower CT the system has, the better the performance will be. These parameters are not independent from each other. Little's law makes connection among them:

$$WIP = TH \times CT$$

The variability of procedures is measured by the coefficient of variation (CV):

$$CV = \frac{\text{standard deviation}}{\text{mean}}$$

Hopp and Spearman use two so called characteristic functions to analyze the performance. The dependent variables

To appear in the Middle-European Conference on Applied Theoretical Computer Science, October 2016, Koper, Slovenia

^{*}to whom all correspondance should be addressed

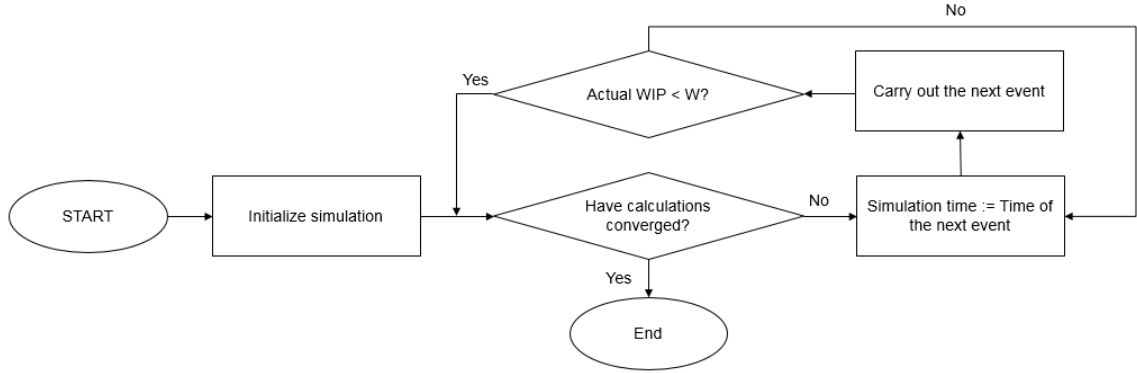


Figure 1: The simplified flow chart of the simulation model

are the TH and the CT, while the independent variable is the WIP level both times. The flow line is modelled as a closed network. It means that the level of WIP is a model parameter [14].

Regarding performance analysis, three important concepts were introduced [6]:

- Best case performance: the best possible performance for a line. It is balanced, and there is no batching.
- Worst case performance: the worst possible performance for a line. All the entities move in one batch.
- Practical worst case (PWC): As the worst case performance is so bad that it is far from practical instances, PWC was introduced to define a realistic worst case.

The paper’s aim is to provide some insight regarding the performance of balanced and unbalanced discrete manufacturing flow lines. The deteriorating effect of variability in balanced and unbalanced systems is examined in a quantitative manner.

2. METHOD OF EXAMINATION

In this research, the same characteristics are used to evaluate the performance as in [5]. Both physical and simulation model experiments are performed to gather data. The models were closed networks containing single machine stations and using CONWIP control. In the physical model experiment, a toy car factory has been realized with the assumption of infinite raw material stock and stochastic demand. The entire process to build a small car takes 4 minutes. In an arbitrary way, the operations could be distributed among the stations where one-one person worked with different abilities.

The simulation model is a discrete time simulation program with next-event time advance mechanism. Comparing with fixed-increment time advance method, it is more complicated, but more efficient regarding computational need [15]. Figure 1 shows the basic mechanics of the model. W notes the WIP level of the model while the actual WIP reflects the state of the simulation. In the model, the process times are stochastic variables with normal distributions.

2.1 Implementation of the model

The simulation program is implemented in AIMMS modelling language [11]. It has already been used in other studies with success. E.g., [3] used it on supply chain optimization with homogenous product transport constraints. It is chosen for a number of reasons. The simulation program can be easily extended in this environment. AIMMS is linked to the most modern solvers, which are easily integratable. Furthermore, it has an advanced graphical user interface, which can be used for creating simply usable and aesthetic softwares.

3. COMPUTATIONAL RESULTS

3.1 The effect of variability

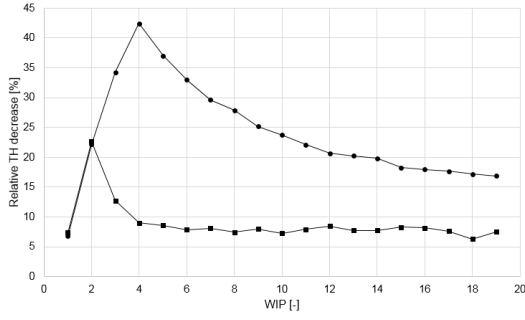
The results of the physical experiments showed performance decrease because of the variability. This effect has already been shown in [6]. Balanced flow lines with different CV’s are compared with the deterministic case (see Figure 3). As the CV grows, the TH decreases, and the CT increases. Comparing the lines on the optimal WIP level of the deterministic case, that is to say on $WIP = 4$, it can be stated that TH gets lower by 13% at $CV = 0.2$, 23% at $CV = 0.4$, 31% at $CV = 0.6$, 37% at $CV = 0.8$ contrasted to the deterministic line. In the meantime, CT increases by 14% at $CV = 0.2$, 30% at $CV = 0.4$, 44% at $CV = 0.6$, and 58% at $CV = 0.8$ compared to the deterministic case.

Contrary, unbalanced systems are less sensitive to the influence of variability. A balanced and an unbalanced line are set against each other on figure 2. Relative changes are displayed on the ordinate, which shows the deteriorating effect of variability from a different aspect. It is easier to see the difference in the drop of performance regarding WIP. These characteristics are calculated in the following way:

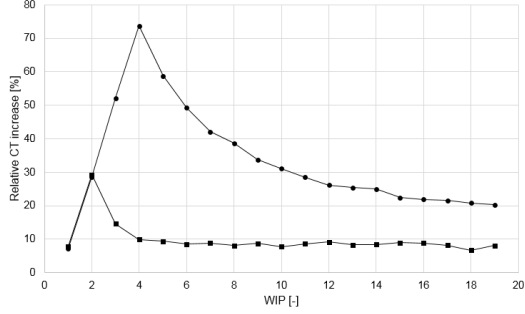
$$TH_{rel} = \frac{abs(TH_{stoch} - TH_{det})}{TH_{det}}$$

Table 1: Comparison of the maximal deteriorations

	Balanced	Unbalanced
TH	42%	23%
CT	174%	129%



(a) Decrease of TH



(b) Increase of CT

■ Unbalanced ● Balanced

Figure 2: The effect of variability on the performance of flow lines.

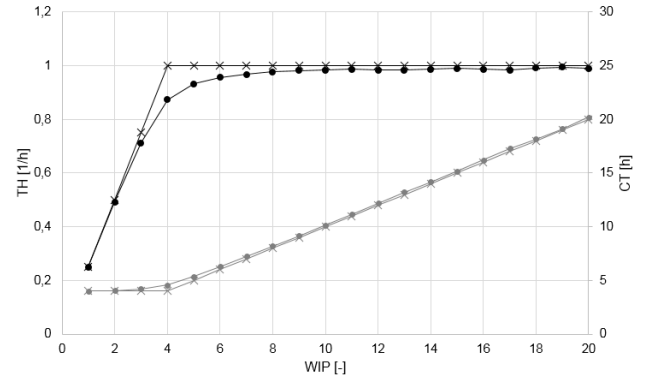
$$CT_{rel} = \frac{abs(CT_{stoch} - CT_{det})}{CT_{det}}$$

The extent of deterioration is bigger when the line is balanced. In this case, the maximal TH decrease is 42% and 23% in the unbalanced one. The maximal CT increase is 74% when the flow line is balanced; 29% when it is unbalanced. It means that the maximal deterioration of TH is twice as high in balanced lines than in unbalanced lines, and the CT maximum is 2.5 times as high. The loss of TH and the growth of CT increase until the deterministically optimal WIP value is reached. The curves of both system move together until the lower deterministically optimal WIP value. After the peak, both functions begin to decrease. At high WIP levels, they will converge into 1. Table 1 sums up the results regarding the peaks.

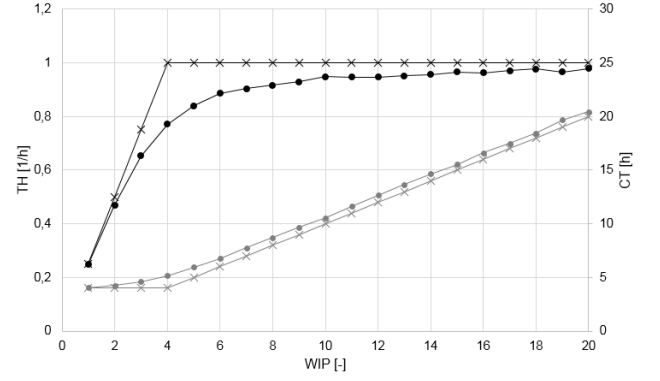
3.2 System unbalancing

In this research, a tradeoff is assumed between performance and stability. Balanced systems usually give better performance while unbalanced systems more stability. However, there are situations when unbalanced system is more efficient.

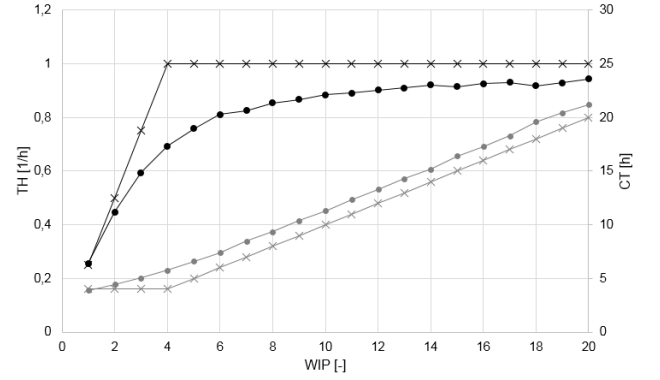
It is illustrated by a case study here. Three systems are compared: a balanced and two unbalanced. The balanced system has uniform process times of 1 hour. The CV of the first three stations equal to 0.1, and the last station's CV is 1. In the unbalanced systems, the process times of three operations are 1.15 hour, and there is one with 0.55 hour. The CV values are the same compared with the balanced system.



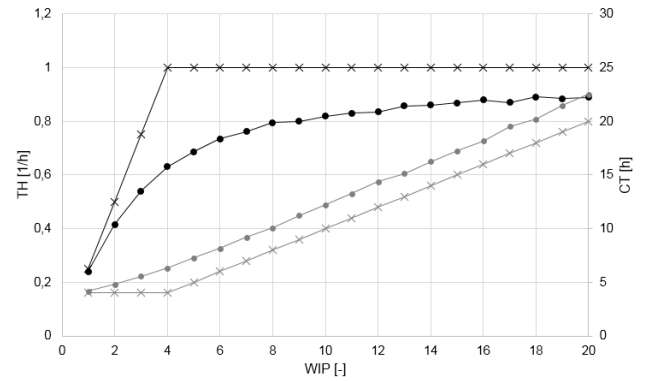
(a) CV = 0.2



(b) CV = 0.4



(c) CV = 0.6



(d) CV = 0.8

× Deterministic TH ● Stochastic TH × Deterministic CT → Stochastic CT

Figure 3: The impairing effect of variance

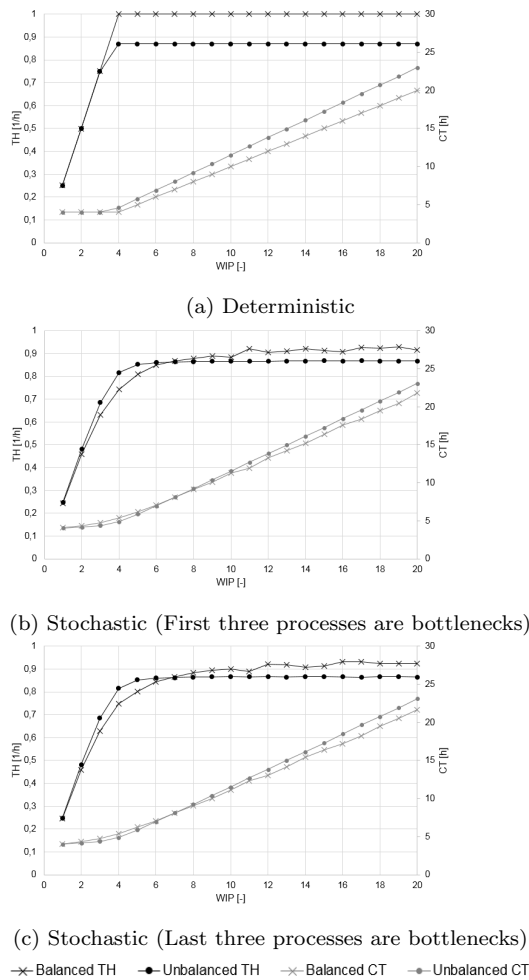


Figure 4: Comparison of the performance of a balanced and two unbalanced systems

The station with the process time of 0.55 hour has $CV = 1$. Two unbalanced cases are examined. They differ in the position of the non-bottleneck process. A balanced and two unbalanced flow lines are examined in order to investigate the tradeoff (see figure 4). While the balanced system has a better performance regarding any WIP level in the deterministic case, it is not true when stochastic processes are investigated. Around the WIP optimum, the unbalanced flow line has a better output. In practical cases, the optimal level of WIP is about where the derivatives of the functions change in the deterministic case. This is the region where unbalanced systems work better (see Figure 4). According to the experiments, the TH of the unbalanced system can be 9-11% higher compared with the balanced line, the CT is 8-9% lower in the earlier case. The positions of the bottleneck procedures have no effect in the investigated cases. The results confirm the assumption that there is a tradeoff between performance and stability, and it can be handled as an optimization problem.

4. CONCLUSIONS

Endeavours are generally made to balance flow lines. This is an intuitive idea, and earlier researches showed examples

where unbalanced systems had worse performance. In this paper, it was showed that **unbalancing the flow line in a small extent achieves better performance on low WIP levels**, that is to say higher TH and lower CT. In the examined case, the TH was 9-11% higher and the CT 8-9% lower on the optimal WIP level of the deterministic case.

5. REFERENCES

- [1] J. T. Black and B. J. Schroer. Simulation of an apparel assembly cell with walking workers and decouplers. *Journal of Manufacturing Systems*, 12(2):170–180, 1993.
- [2] K. Cho, I. Moon, and W. Yun. System analysis of a multi-product, small-lot-sized production by simulation: A korean motor factory case. *Computers & Industrial Engineering*, 30(3):347–356, July 1996.
- [3] T. Farkas, Z. Valentinyi, E. Rév, and Z. Lelkes. Supply chain optimization with homogenous product transport constraints. In *Computer Aided Chemical Engineering*, volume 25, pages 205–210, 2008.
- [4] M. Govil and M.C.Fu. Queueing theory in manufacturing: A survey. *Journal of Manufacturing Systems*, 18(3):214–240, 1999.
- [5] W. J. Hopp. *Supply Chain Science*. Waveland Pr Inc, Long Grove, Illinois, 2011.
- [6] W. J. Hopp and M. L. Spearman. *Factory Physics*. McGraw-Hill Education, New York, New York, 2000.
- [7] W. Huang and S. Li. A two-stage hybrid flowshop with uniform machines and setup times. *Mathematical and Computer Modelling*, 27(2):27–45, January 1998.
- [8] J. Olhager and B. Rapp. Balancing capacity and lot sizes. *European Journal of Operational Research*, 19(3):337–344, March 1985.
- [9] H. T. Papadopoulos and C. Heavey. Queueing theory in manufacturing systems analysis and design: A classification of models for production and transfer lines. *European Journal of Operational Research*, 92(1):1–27, July 1996.
- [10] Y. H. Park, J. E. Matson, and D. M. Miller. Simulation and analysis of the mercedes-benz all activity vehicle (aav) production facility. In *Proceedings of the 1998 Winter Simulation Conference*, pages 921–926, December 1998.
- [11] M. Roelofs and J. Bisschop. *AIMMS The user's guide*. AIMMS B.V., AP Haarlem, The Netherlands, 2016.
- [12] J. Sarkis. A strategic decision framework for green supply chain management. *Journal of Cleaner Production*, 11(4):397–409, June 2003.
- [13] J. S. Smith. Survey on the use of simulation for manufacturing system design and operation. *Journal of Manufacturing Systems*, 22(2):157–171, 2003.
- [14] W. Whitt. Open and closed models for networks of queues. *AT&T Bell Laboratories Technical Journal*, 63(9):1911–1979, November 1984.
- [15] W. L. Winston. *Operations Research: Applications and Algorithms*. Cengage Learning, Boston, Massachusetts, 2003.
- [16] H. Yamashita and S. Suzuki. An approximation method for line production rate of a serial production line with a common buffer. *Computers & Operations Research*, 15(5):395–402, 1988.

Process Network Solution of a Soleplate Manufacturer's Extended CPM Problem with Alternatives

Nándor Vincze,
Department of Applied
Informatics, Faculty of
Education, University of
Szeged, Boldogasszony
u. 6, 6725 Szeged, Hun-
gary,
vincze@jgypk.u-
szeged.hu

Zsolt Ercsey
Department of System
and Software
Technology, Faculty of
Engineering and
Information Technology,
University of Pécs,
Boszorkány u. 2, 7624
Pécs, Hungary,
ercsey@mik.pte.hu

Zoltán Kovács
Department of
Computational
Optimization, Institute of
Informatics, University of
Szeged, Árpád tér 2,
6720 Szeged, Hungary,
kovacs@inf.u-
szeged.hu

ABSTRACT

In this paper a Hungarian soleplate manufacturer's problem is described in details, extended with alternatives and effectively solved. First, after the presentation of the industrial problem, the CPM graph of the problem is given and then it is transformed into a process network. Then the original problem is extended with alternatives specified by various industrial needs, for example an activity is performed in two different ways and resources with different time and costs. Then the corresponding mathematical programming model is formulated: time optimal and time optimal with additional cost constraints mathematical programming models are given. Please note that only the earlier corresponds to the CPM problem and the latter is an extension. The solution illustrates the efficacy of the method.

INTRODUCTION

The CPM (critical path method) is an algorithmic approach of scheduling a set of activities, where the duration times of the activities are known together with their dependencies and the aim is to calculate the longest path of the planned activities. The method originates in the 1950s. For advances

in CPM, please see Chanas and Zielinski, 2001, Li et al. 2015, Madhuri et al. 2013. It is worth mentioning that CPM orders the resources to the activities without a representation in the CPM graph. In case other type of resources are ordered to the activities, the parameters of the problem have to be reset and the problem has to be solved again. This may result in a large number of problems to be solved for a single case. Moreover, it is also not handled by CPM where a given subtask can be solved in different ways.

Process network synthesis is an optimization methodology basically used in the chemical industry. Based on a mathematical rigor, graph theoretical approaches and combinatorial techniques are combined with the first focus on the synthesis step, ie structure generation. This method enables the consideration of alternatives as well as generates all feasible solutions within one model and one time solution process. For details, see Friedler et al. 1992a, 1992b, Tick et al. 2013, Kovács et al. 1999 and 2000, Garcia-Ojeda et al. 2015, Losada et al. 2015.

Transforming CPM problems into process networks is described by Vincze et al 2015. First the two terminologies are mapped: an event corresponds to a material, an activity corresponds

to an operating unit, the dependencies between the activities correspond to the material flows and the CPM graph corresponds to the process network. After the structural mapping and the logical connections establishments, each parameter of the original CPM is represented in the resulting process network.

Industrial examples always raise the question of alternatives, where a given subproblem can be solved by performing more than one activity or more than one series of activities, or various resources can be ordered to the activities with different durations, costs etc. These situations cannot be represented within one CPM problem, but after the transformation these can be handled within the process network model.

SOLEPLATE MANUFACTURING

A manufacturer in the southern region of Hungary produces various types of soleplates for irons. For this research paper the production of Gx3 and EP5 types were investigated. These two soleplate types are identical in terms of shape but they differ in terms of assembling. Gx3 soleplates are assembled by 4 operators in 8 hours shift when the target is 1000 solaplates; and by 5 operators in case of larger quantities. Other operators support this production work with unpacking and material handling; since these workers belong to other work groups, their work tasks were considered to be handled by one additional operator. The production line is linear. One piece of Gx3 soleplate gets finished by 68 seconds.

Table 1. Production steps; with alternatives indicated in brackets

Gx3 soleplate production activities	Time (sec)	Cost
Spider bending	7	5
Fuse welding	7	3
Cut	5	4
Thermostat welding (alternative thermostat welding)	10 (8)	5 (7)
Spider screwdriwing	14	4
Spider soleplate welding	13	5
Edging	3	4
Cut	2	5
Test (alternative test)	7 (9)	5 (4)
Unpacking	7	5
Put on the work table (alternative put on the work table)	5 (3)	4 (6)

Gx3 and EP5 types are different. While Gx3 soleplate is fixed by 2 screws and the inserted part has to be edged, EP5 has a different formation and is fixed in two different phases with 3 screws driven in. Another major difference is that the soleplates arrive to the production line wrapped and thus a separate unpacking activity has to be performed, which requires extra time from the operator. The previously mentioned additional operator supporting this production line performs the unpacking in a serious of steps: soleplates arrive in carton boxes, the boxes are first cut along a mark, then the foils are removed

from the cut boxes, then the soleplates are put on the work table. EP5 soleplates are put on the work table on a plastic tray and with a paper separator; these have to be removed first and the plastic plus the paper have to be separately put away into their containers, then the soleplates have to be put on the work table. These additional packaging tasks result that a piece of EP5 solaplate gets finished by 71 seconds. It can be seen that the two types of soleplates are very similar, yet different in some ways. The differences of the necessary operating times may be indicated for example in Yamazumi tables.

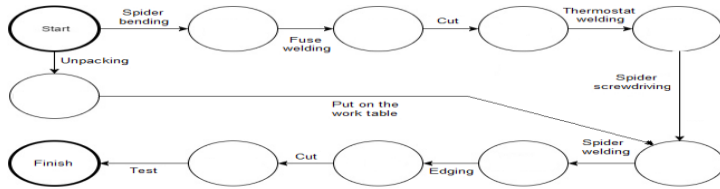


Figure 1. CPM graph of the soleplate manufacturing.

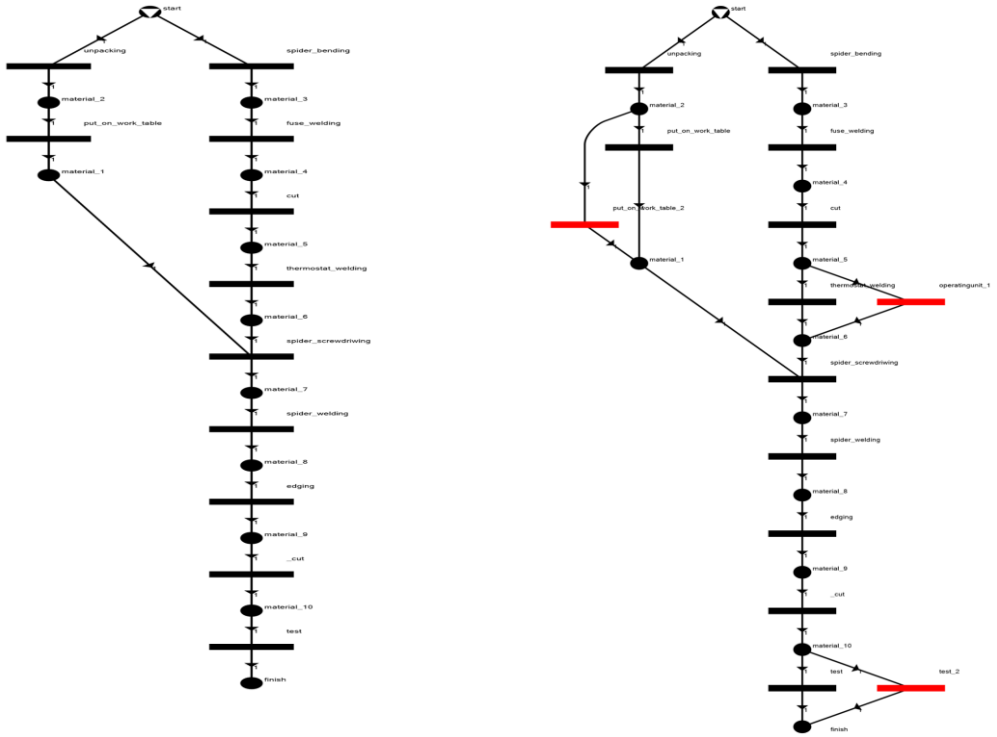


Figure 2. A. Process network of the soleplate manufacturing.

B. Process network of the manufacturing with alternatives.

Please note that Figure 2.A. illustrates the process network model of the soleplate manufacturing and Figure 2.b contains the extended model with alternative activities. Three activities may be performed in two ways, namely

- put on the work table activity may be performed under 3 seconds or as an alternative under 5 seconds when a student or an assistant performs the activity; please note that obviously

their costs also differ, namely the former has a cost of 6 and the latter has a cost of 4;

- thermostat welding activity may be performed under 10 seconds or as an alternative under 8 seconds when a senior welder performs the activity; please note that obviously their costs also differ, namely the former has a cost of 5 and the latter has a cost of 7;

- test activity may be performed under 7

seconds or as an alternative under 9 seconds when a junior worker performs the activity; please note that obviously their costs also differ; namely the former has a cost of 5 and the latter has a cost of 4.

When only time is considered in the mathematical programming model, it corresponds to the original CPM problem extended with the above mentioned alternatives. Since in this industrial case study financial issues were also taken into consideration, therefore the time optimal mathematical programming model was also extended with costs constraints. Please note that these models are detailed in Vincze et al. 2016.

CONCLUDING REMARKS

The CPM gives the longest path of the planned activities together with its overall duration, nevertheless, for industrial real case problems where financial issues also influence the decisions of the production processes it is very

ACKNOWLEDGEMENTS

The present scientific contribution is dedicated to

REFERENCES

- [1] Chanas S. and P. Zielinski, Critical path analysis in the network with fuzzy activity times, *Fuzzy Sets and Systems*, (2001) 122, 195–204
- [2] Friedler, F., K. Tarjan, Y. W. Huang, and L. T. Fan, Graph-Theoretic Approach to Process Synthesis: Axioms and Theorems, *Chem. Engng Sci.*, 47, 1973-1988 (1992a).
- [3] Friedler, F., K. Tarjan, Y. W. Huang, and L. T. Fan, Combinatorial Algorithms for Process Synthesis, *Computers Chem. Engng*, 16, S313-320 (1992b)
- [4] Zhenhong Li, Yankui Liu, Guoqing Yang: A New Probability Model for Insuring Critical Path Problem with Heuristic Algorithm. *Neurocomputing* 148: 129-135 (2015)
- [5] Madhuri, U., Saradhi, P. ve Shankar, R., (2014) "Fuzzy Linear Programming Model for Critical Path Analysis", *Int. J. Contemp. Math. Sciences*, Vol. 8, No. 2, 2013, pp. 93-116

The overall cost constraint of the manufacturer was determined to be 55. Within this constraint the optimal production process contains the original put on the work table activity and the alternative thermostat welding activity and the original test activity. When the manufacturer determined the overall cost constraint to be 50, then the optimal production process contains the alternative put on the work table activity and the alternative thermostat welding activity and the alternative test activity.

Please note that the same method was used when solving the industrial problem of the EP5 type soleplate manufacturing process.

complicated to reach a true goal. Therefore in the present paper a new process network method was used which extends the problem range of CPM problems with alternatives as well as cost constraints.

the 650th anniversary of the foundation of the University of Pécs, Hungary.

[6] Garcia-Ojeda, J.C., B. Bertok, F. Friedler, A. Argoti, and L.T. Fan, A Preliminary Study of the Application of the P-graph Methodology for Organization-based Multiagent System Designs: Assessment, *Acta Polytechnica Hungarica*, 12, 103-122 (2015).

[7] József Tick, Csanád Imreh, Zoltán Kovács, Business Process Modeling and the Robust PNS Problem, *Acta Polytechnica*, DOI: 10.12700/APH.10.06.2013.6.11, Volume 10, Issue Number 6, 193-204, 2013

[8] Losada, J.P., I. Heckl, B. Bertok, F. Friedler, J. C. Garcia-Ojeda, and A. Argoti, Process-network Synthesis for Benzaldehyde Production: P-graph Approach, *Chemical Engineering Transactions*, 45, 1369-1374 (2015).

[9] Vincze, N., Z. Ercsey, T. Kovács, J. Tick, Z. Kovács, Process Network Solution of Extended CPM Problems with Alternatives, *ACTA POLYTECHNICA HUNGARICA* 13:(3) pp. 101-117. (2016)

ON NIST test of a Novel Cryptosystem Based on Automata Compositions

Pál Dömösi
Institute of Mathematics and
Informatics
University of Nyíregyháza
H-4400 Nyíregyháza, Sóstói út
31/B, Hungary
domosi.pal@nye.hu

József Gáll
Faculty of Informatics
University of Debrecen
H-4028 Debrecen, Kassai út
26, Hungary
gall.jozsef@inf.unideb.hu

Géza Horváth
Faculty of Informatics
University of Debrecen
H-4028 Debrecen, Kassai út
26, Hungary
horvath.geza@inf.unideb.hu

Norbert Tihanyi
Faculty of Informatics
Eötvös Loránd University
H-1117 Budapest, Pázmány
Péter sétány 1/C, Hungary
tihanyi.pgp@gmail.com

ABSTRACT

In this paper we discuss on NIST test results of a previously introduced cryptosystem based on automata compositions. Our conclusions based on the statistics confirm that the requirements of NIST test are fulfilled.

Keywords

automata network, NIST, block cipher, statistics

1. INTRODUCTION

The history of cryptography is crowded by examples when supposed to be very safe encryption systems were proved breakable. Based on the simple probability theory and mathematical logic, the one-time pad system (OTP), – which is commonly called Vernam cipher – the only known cryptographic system that is completely unbreakable. Only this system is known to have a mathematical proof on its perfect secrecy [17]. Although the OTP is the most reliable form of encryption, in practice its use is not efficient. Each user must have a copy of the symmetric key and the key exchange can only be accomplished through secure communication channels. The key can not be used more than once and the key size must be at least the size of the encoded text. OTP is a symmetric system, where the decryption and encryption key coincide, or any of them can be easily derived from the other. Therefore, both of the encryption and decryption keys must be secret, and those secret keys should be known only by the sender and the recipient of the message. Another main type of cryptosystems is the asymmetric one –

also called public-key cryptosystem, – where the determination of decryption key is infeasible for anyone knowing only the encryption key. (The principle of public-key cryptography was invented by Diffie and Hellman in 1976.) The discussed novel cipher is a symmetric system.

Several types of cryptosystems based on automata theory have been designed so far. Some of them are based on Mealy automata [11, 16, 19, 20] or their generalization [2], while others are based on cellular automata [5, 10, 14, 13]. Almost all of the best-known automaton based cryptosystems share the common problem of serious realization difficulties: some systems are easy to defeat [3, 4, 15], the technical realization of others result in slow performance [10], still others exhibit difficulties in the choice of the key-automaton [14, 13], some of them has no known rigorous security analysis and the security of some systems is largely unknown [5].

In [6, 7] we introduced new block ciphers based on Gluškov-type product of automata. In this paper we investigate the system [6]. Both systems use the following simple idea: Consider a giant-size permutation automaton such that the set of states and the set of inputs consist of all given length of strings over a non-trivial alphabet as all possible plaintext/ciphertext blocks. Moreover consider a cryptographically secure pseudo random number generator with large periodicity having the property that, getting its really random kernel, it serves a sequence of pseudo random strings as inputs for the automaton. For each plaintext block the system calculates the new state into which the actual pseudorandom string takes the automaton from the state which is identified as the actual plaintext block. The string, identified as the new state, will be the ciphertext block ordered to the considered plaintext block. Of course, the ciphertext will be the concatenation of the generated ciphertext block. The giant size of the automaton makes it infeasible to break the system by brute-force method.

The problem of this idea is that store of the transition matrix of giant-size automata is impossible. Another idea is

that this problem can be overcome considering automata which consists of composition of automata. In this case, we should store only the component-automata and the structure of the composition. Moreover, if the component automata are isomorphic to each others then it is enough to store the transition matrix of one component automaton and the structure of the isomorphisms. By this recognition, the storage of automata having 2^{128} states and 2^{128} input signs can be easily solved. The basic idea of this cipher is to operate on a giant secret square matrix which is compressed into the memory using automata-theoretic methods. The matrix has 2^{128} rows and 2^{128} columns such that each of its rows is a permutation of all bitstrings of 128 bit length. Using automata-theoretic methods, we can easily handle this giant matrix. Because of the giant size of the matrix, there is no hope to attack the system by brute-force method. On the other hand, this giant matrix can be generated unambiguously by a bitstring of 782 bytes length. Note that this less than 1 kilobyte long string can be generated by an appropriate hash function using a secret password of any length.

For all notions and notation not defined here we refer to the monographs [8, 9, 12, 1]. The discussed cryptosystem is a block cipher. Since the key automaton is a permutation automaton, for every ciphertext there exists exactly one plaintext making the encryption and decryption unambiguous. Moreover, there is a huge number of corresponding encoded messages to each plaintext so that several encryptions of the same plaintext yield several distinct ciphertexts.

2. THEORETICAL BACKGROUND

By an *automaton* we mean a deterministic finite automaton without outputs. If all the rows of the transition matrix are permutations of the state set then we have a *permutation automaton*.

Lemma 1. An automaton $\mathcal{A} = (A, \Sigma, \delta)$ is a permutation automaton if and only if for any $a, b \in A, x \in \Sigma, \delta(a, x) = \delta(b, x)$ implies $a = b$.

Let $\mathcal{A}_i = (A_i, \Sigma_i, \delta_i)$ be automata where $i \in \{1, \dots, n\}, n \geq 1$. Take a finite nonvoid set Σ and a *feedback function* $\varphi_i : A_1 \times \dots \times A_n \times \Sigma \rightarrow \Sigma_i$ for every $i \in \{1, \dots, n\}$. The *Gluškov-type product* of the automata \mathcal{A}_i with respect to the feedback functions φ_i ($i \in \{1, \dots, n\}$) is defined to be the automaton $\mathcal{A} = \mathcal{A}_1 \times \dots \times \mathcal{A}_n(\Sigma, (\varphi_1, \dots, \varphi_n))$ with state set $A = A_1 \times \dots \times A_n$, input set Σ , transition function δ given by $\delta((a_1, \dots, a_n), x) = (\delta_1(a_1, \varphi_1(a_1, \dots, a_n, x)), \dots, \delta_n(a_n, \varphi_n(a_1, \dots, a_n, x)))$ for all $(a_1, \dots, a_n) \in A$ and $x \in \Sigma$. In particular, if $\mathcal{A}_1 = \dots = \mathcal{A}_n$ then we say that \mathcal{A} is a *Gluškov-type power*.

We shall use the feedback functions $\varphi_i, i = 1, \dots, n$ in an extended sense as mappings $\varphi_i^* : A_1 \times \dots \times A_n \times \Sigma^*$, where $\varphi_i^*(a_1, \dots, a_n, \lambda) = \lambda$ and $\varphi_i^*(a_1, \dots, a_n, px) = \varphi_i^*(a_1, \dots, a_n, p)\varphi_i(\delta_1(a_1, \varphi_1^*(a_1, \dots, a_n, p)), \dots, \delta_n(a_n, \varphi_n^*(a_1, \dots, a_n, p)), x), a_i \in A_i, i = 1, \dots, n, p \in \Sigma^*, x \in \Sigma$. In the sequel, $\varphi_i^*, i \in \{1, \dots, n\}$ will also be denoted by φ_i .

Let $\mathcal{A}_t = (A, \Sigma_t, \delta_t), t = 1, 2$ be automata having a common state set A . Take a finite nonvoid set Σ and a mapping φ of Σ into $\Sigma_1 \times \Sigma_2$. Then the automaton $\mathcal{A} =$

(A, Σ, δ) is a *temporal product* (*t-product*) of \mathcal{A}_1 by \mathcal{A}_2 with respect to Σ and φ if for any $a \in A$ and $x \in \Sigma, \delta(a, x) = \delta_2(\delta_1(a, x_1), x_2)$, where $(x_1, x_2) = \varphi(x)$. The concept of temporal product is generalized in the natural way to an arbitrary finite family of $n > 0$ automata \mathcal{A}_t ($t = 1, \dots, n$), all with the same state set A , for any mapping $\varphi : \Sigma \rightarrow \prod_{t=1}^n \Sigma_t$, by defining $\delta(a, x) = \delta_n(\dots \delta_2(\delta_1(a, x_1), x_2), \dots, x_n)$ when $\varphi(x) = (x_1, \dots, x_n)$. In particular, a temporal product of automata with a single factor is just a (one-to-many) relabeling of the input letters of some input-subautomaton of its factor.

Lemma 2. Every temporal product of permutation automata is a permutation automaton.

Given a function $f : X_1 \times \dots \times X_n \rightarrow Y$, we say that f is *really independent of its i -th variable* if for every pair $(x_1, \dots, x_n), (x_1, \dots, x_{i-1}, x'_i, x_{i+1}, \dots, x_n) \in X_1 \times \dots \times X_n, f(x_1, \dots, x_n) = f(x_1, \dots, x_{i-1}, x'_i, x_{i+1}, \dots, x_n)$. Otherwise we say that f *really depends on its i -th variable*.

A (finite) *directed graph* (or, in short, a *digraph*) $\mathcal{D} = (V, E)$ (of order $n > 0$) is a pair consisting of sets of *vertices* $V = \{v_1, \dots, v_n\}$ and *edges* $E \subseteq V \times V$. Elements of V are sometimes called *nodes*. Moreover, if $(v, v') \in E$ then it is said that (v, v') is an *outgoing edge* of v , and simultaneously, (v, v') is an *incoming edge* for v' . (In this way, a loop edge (v, v) has both of these properties with respect to the vertex v .) An edge $(v, v') \in E$ is said to have *source* v and *target* v' . If $|V| = n$ then we also say that \mathcal{D} is a digraph of order n . If V can be decomposed into two disjoint (nonempty) subsets V_1, V_2 such that V_1 is the set of all incoming edges and V_2 is the set of all outgoing edges then we say that \mathcal{D} is a *bipartite digraph*.

Let Σ be the set of all binary strings with a given length $\ell \geq 1$ and let n be a positive integer, let $\mathcal{A}_1 = (\Sigma, \Sigma \times \Sigma, \delta_{\mathcal{A}_1})$ be a permutation automaton and let $\mathcal{A}_i = (\Sigma, \Sigma \times \Sigma, \delta_{\mathcal{A}_i}), i = 2, \dots, n$ be state-isomorphic copies of \mathcal{A}_1 such that $\mathcal{A}_1, \dots, \mathcal{A}_n$ are pairwise distinct, and let n be a power of 2. Consider the following bipartite digraphs:

$$\mathcal{D}_1 = (\{1, \dots, n\}, \{(n/2 + 1, 1), (n/2 + 2, 2), \dots, (n, n/2)\}),$$

$$\mathcal{D}_2 = (\{1, \dots, n\}, \{(n/4 + 1, 1), (n/4 + 2, 2), \dots, (n/2, n/4), (3n/4 + 1, n/2 + 1), (3n/4 + 2, n/2 + 2), \dots, (n, 3n/4)\}),$$

...

$$\mathcal{D}_{\log_2 n - 1} = (\{1, \dots, n\}, \{(3, 1), (4, 2), (7, 5), (8, 6), \dots, (n - 1, n - 3), (n, n - 2)\}),$$

$$\mathcal{D}_{\log_2 n} = (\{1, \dots, n\}, \{(2, 1), (4, 3), \dots, (n, n - 1)\}),$$

$$\mathcal{D}_{\log_2 n + 1} = \mathcal{D}_1,$$

...

$$\mathcal{D}_{2\log_2 n} = \mathcal{D}_{\log_2 n}.$$

For every digraph $\mathcal{D} = (V, E)$ with $\mathcal{D} \in \{\mathcal{D}_1, \dots, \mathcal{D}_{2\log_2 n}\}$, let V_1 be the set of all incoming edges and let V_2 be the set of all outgoing edges, and define the Gluškov-type product,

called *two-phase D-product*,

$\mathcal{A}_{\mathcal{D}} = \mathcal{A}_1 \times \dots \times \mathcal{A}_n(\Sigma^n, (\varphi_1, \dots, \varphi_n))$ of $\mathcal{A}_1, \dots, \mathcal{A}_n$ so that for every $(a_1, \dots, a_n), (x_1, \dots, x_n) \in \Sigma^n, i \in \{1, \dots, n\}$, $\varphi_i(a_1, \dots, a_n, (x_1, \dots, x_n)) = (a_j \oplus x_j, x_i)$, if $(j, i) \in V_1$ and $a_j \oplus x_j$ is the bitwise addition modulo 2 of a_j and x_j , $\varphi_i(a_1, \dots, a_n, (x_1, \dots, x_n)) = (a'_j \oplus x_j, x_i)$, if $(j, i) \in V_2$, a'_j denotes the state into which $\varphi_j(a_1, \dots, a_n, (x_1, \dots, x_n))$ takes the automaton from its state a_j , and $a'_j \oplus x_j$ is the bitwise addition modulo 2 of a'_j and x_j .

Let $\mathcal{B} = (\Sigma^n, (\Sigma^n)^{2 \log_2 n}, \delta_{\mathcal{B}})$ be the temporal product of $\mathcal{A}_{\mathcal{D}_1}, \dots, \mathcal{A}_{\mathcal{D}_{2 \log_2 n}}$ with respect to $(\Sigma^n)^{2 \log_2 n}$ and the identity map $\varphi : (\Sigma^n)^{2 \log_2 n} \rightarrow (\Sigma^n)^{2 \log_2 n}$. We say that \mathcal{B} is a *key-automaton* with respect to $\mathcal{A}_1, \dots, \mathcal{A}_n$.¹ Obviously, \mathcal{B} is unambiguously defined by the transition matrix of \mathcal{A}_1 and the bijective mappings $\tau_1 : \Sigma \rightarrow \Sigma, \dots, \tau_n : \Sigma \rightarrow \Sigma$ which represent the state isomorphisms of $\mathcal{A}_1, \dots, \mathcal{A}_n$ to \mathcal{A} .

An important property of key-automata is explained in the following result.

Theorem 1. Every key-automaton is a permutation automaton.

Both of the encryption and decryption algorithms use a pseudo random generator and the above defined key-automaton.

3. THE NIST TEST

The National Institute of Standards and Technology (NIST) published a statistical package consisting of 15 statistical tests that were developed to test the randomness of arbitrarily long binary sequences produced by either hardware or software based cryptographic random or pseudorandom number generators. In case of each statistical test a set of P-values was produced. Given a significance level α , if the P-value is less than or equal to α then test suggests that the observed data is inconsistent with our null hypothesis, i.e. the 'hypothesis of randomness', so we reject it. We used $\alpha = 0.01$ as it is common in such problems in cryptography. An α of 0.01 indicates that one would expect 1 sequence in 100 sequences to be rejected under the null hypothesis. Hence a P-value exceeding 0.01 would mean that the sequence would be considered to be random, and P-value ≤ 0.01 would lead to the conclusion that the sequence is non-random.

One of the criteria used to evaluate the AES candidate algorithms was their demonstrated suitability as random number generators. That is, the evaluation of their output utilizing statistical tests should not provide any means by which to distinguish them computationally from a truly random source. Randomness testing was performed using the same parameters as for the AES candidates in order to achieve the most reliable and comparable results. First the input parameters –such as the sequence length, sample size, and significance level– were fixed. Namely, these parameters were set at 2^{20} bits, 300 binary sequences, and $\alpha = 0.01$, respectively. Furthermore, Table 1 shows the length parameters we used.

¹Recall that n should be a power of 2.

Table 1: Parameters used for NIST Test Suite

Test Name	Block length
<i>Block Frequency</i>	128
<i>Non-overlapping Template</i>	9
<i>Overlapping Template</i>	9
<i>Approximate Entropy</i>	10
<i>Serial</i>	16
<i>Linear Complexity</i>	500

In order to analyze the output of the algorithm we encrypted the Rockyou database, which contains more than 32 millions of cleartext passwords. Applying the NIST test for the encrypted file it has turned out that the output of the algorithm can not be distinguished in polynomial time from true random sources by statistical tests. The exact p-values of the evaluation of the ciphertext are shown in Table (2). We also tested the uniformity of the distribution of the p-values obtained by the statistical tests included in NIST, which is a usual requirement in the literature (see e.g. [18]). The uniformity of p-values provide no additional information about the type of the cryptosystem. We have also shown that the proportions of binary sequences which passed the 0.01 level lie in the required confidence interval (see e.g. [18]).

Table 2: Results for the uniformity of p-values and the proportion of passing sequences

P-value	PROPORTION	STATISTICAL TEST
0.162606	296/300	<i>Frequency</i>
0.407091	298/300	<i>BlockFrequency</i>
0.574903	297/300	<i>CumulativeSums</i>
0.840081	295/300	<i>CumulativeSums</i>
0.205897	297/300	<i>Runs</i>
0.284959	297/300	<i>LongestRun</i>
0.527442	297/300	<i>Rank</i>
0.623240	298/300	<i>FFT</i>
0.958773	295/300	<i>NonOverlappingTemplate</i>
.	.	.
...
0.419021	299/300	<i>OverlappingTemplate</i>
0.220931	298/300	<i>Universal</i>
0.935716	299/300	<i>ApproximateEntropy</i>
0.516465	171/177	<i>RandomExcursions</i>
0.384836	172/177	<i>RandomExcursionsVariant</i>
0.042808	298/300	<i>Serial</i>
0.253551	296/300	<i>Serial</i>
0.039244	295/300	<i>LinearComplexity</i>

3.1 Sárközy and Mauduit randomness test

Different security audits and processes use different statistical tests and methods. In order to fulfill further requirements we performed the Sárközy and Mauduit methods in order to study the behaviour of pseudorandom sequences generated by our cryptosystem. Let $E_N = \{e_1, e_2, \dots, e_N\} \in \{-1, +1\}^N$ represent a finite binary sequence. Let us define

$$U(E_N, t, a, b) = \sum_{j=0}^t e_a + j_b$$

The *well-distribution measure* of E_N is defined by

$$W(E_N) = \max_{M,u,v} |U(E_N, M, u, v)| = \max_{M,u,v} \left| \sum_{j=0}^{M-1} e_{u+jv} \right|$$

where the maximum is taken over all M, u, v with $u + (M-1)v \leq N$. Furthermore let us define

$$V(E_N, M, D) = \sum_{n=1}^M e_{n+d_1} e_{n+d_2} \dots e_{n+d_k}$$

The *correlation measure of order k* of E_N is defined by

$$C_k(E_N) = \max_{M,D} |V(E_N, M, D)| = \max_{M,D} \left| \sum_{n=1}^M e_{n+d_1} e_{n+d_2} \dots e_{n+d_k} \right|$$

where the maximum is taken over all M and $D = (d_1, \dots, d_k)$ such that $0 \leq d_1 \leq \dots \leq d_k \leq N - M$. The goodness of a PRNG is determined by the order of $W(E_N)$ and $C_k(E_N)$. We were not able to distinguish the output of our cryptosystem from true random sources by analyzing the deviation of $W(E_N)$ and $C_k(E_N)$.

4. CONCLUSIONS

The output of our crypto algorithm has passed all statistical tests we performed (NIST test, Sárközy and Mauduit test) and **we were not able to distinguish it from true random sources** by statistical methods. Statistical analyses of a cryptosystem is a must have requirement, and these tests are good indicators that further analyses should be done. Exact cryptoanalyses like chosen-plaintext, known-plaintext and related-key attack will be investigated in order to prove or disprove the strength of this cryptosystem. These problems are the subject of our future research.

5. REFERENCES

- [1] P. C. O. A. J. Menezes and S. A. Vanstone. *Handbook of Applied Cryptography ser. Discrete Mathematics and Its Applications*, doi 10.1201/9781439821916. CRC Press, 1996.
- [2] A. Atanasiu. A class of coders based on gsm. *Acta Informatica*, 29:779–791, 1992.
- [3] F. Bao. Cryptoanalysis of partially known cellular automata. *IEEE Trans. on Computers*, 53:1493–1497, 2004.
- [4] E. Biham. Cryptoanalysis of the chaotic map cryptosystem suggested at eurocrypt’91. In *D. W. Davies, ed., Proc. Conf. Advances in Cryptology, Workshop on the Theory and Application of Cryptographic Techniques, Brighton, UK*, pages 532–534. EUROCRYPT’91, April 8-11 1991.
- [5] A. Clarridge and K. Salomaa. A cryptosystem based on the composition of reversible cellular automata. *LATA, LNCS 5457*, pages 314–325, 2009.
- [6] P. Dömösi and G. Horváth. A novel cryptosystem based on abstract automata and latin cubes. *Studia Scientiarum Mathematicarum Hungarica*, 52(2):221–232, 2015.
- [7] P. Dömösi and G. Horváth. A novel cryptosystem based on gluškov product of automata. *Acta Cybernetica*, 22:359–371, 2015.
- [8] P. Dömösi and C. L. Nehaniv. *Algebraic theory of automata networks: An introduction*. ser. SIAM monographs on Discrete Mathematics and Applications, vol. 11, Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, doi 10.1137/1.9780898718492, "2005".
- [9] F. Gécseg. *Products of automata ser. EATCS Monographs on Theoretical Computer Science, vol. 7*, doi 10.1007/978-3-642-61611-2. Springer-Verlag, Berlin, Heidelberg, New York, Tokyo, 1986.
- [10] P. Guan. Cellular automaton public key cryptosystem. *Complex Systems*, 1:51–56, 1987.
- [11] M. Gysin. One-key cryptosystem based on a finite non-linear automaton. In *E. Dawson and J. Golic, eds., Proc. Int. Conf. Proceedings of the Cryptography: Policy and Algorithms, Lecture Notes in Computer Science 1029, Springer-Verlag, Berlin*, pages 165–163. CPAC’95, Brisbane, Queensland, Australia, July 3-5 1995.
- [12] R. M. J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation, second edition, Addison-Wesley series in computer science*. Addison-Wesley, 2001.
- [13] J. Kari. Reversibility of 2d cellular automata is undecidable. *Physica D*, 45:379–385, 1990.
- [14] J. Kari. Cryptosystems based on reversible cellular automata. *University of Turku, Finland, preprint*, April 1992.
- [15] T. Meskaten. On finite automaton public key cryptosystems. *TUCS Technical Report, Turku Centre for Computer Science, Turku*, No. 408:1–42, 2001.
- [16] V. J. Rayward-Smith. *Mealy machines as coding devices In: H. J. Beker and F. C. Piper, eds., Cryptography and Coding*. Clarendon Press, Oxford, 1989.
- [17] C. Shannon. Communication theory of secrecy systems. *Bell System Technical Journal*, 28(4):656–715, 1949.
- [18] J. Soto. Statistical testing of random number generators. *National Institute of Standards and Technology*, <http://csrc.nist.gov/groups/ST/toolkit/rng/>, downloaded in August 2016.
- [19] R. Tao. *Finite Automata and Application to Cryptography*. Springer-Verlag, Berlin, 2009.
- [20] R. Tao and S. Chen. The generalization of public key cryptosystem fapkc4. *Chinese Science Bulletin*, 44(9):784–790, May 1999.

ALGator– An Automatic Algorithm Evaluation System

Tomaž Dobravec
Faculty of Computer and Information Science
University of Ljubljana, Slovenia
tomaz.dobravec@fri.uni-lj.si

ABSTRACT

In this paper we present an automatic algorithm evaluation system called ALGator, which was developed to facilitate the algorithm design and evaluation process. The system enables unbiased tests of the correctness and quality of implemented algorithms for solving various kinds of problems (e.g. sorting data, matrix multiplication, traveler salesman problem, shortest path problem, and the like). Within the ALGator one can define a problem by specifying the problem descriptors, test sets with corresponding test cases, input parameters and output indicators, algorithm specifications and criteria for measuring the quality of algorithms. When a user of the system submits an algorithm for solving the given problem, ALGator automatically executes this algorithm on predefined tests, measures the quality indicators and prepares the results to be compared with the results of other algorithms in the system. ALGator is meant to be used by algorithm developers to perform independent quality tests for their solutions.

Keywords

algorithm development, automatic execution and evaluation, algorithm testing and analysis

1. INTRODUCTION

Algorithm evaluation is a very important part of an algorithm design and implementation process. The ALGator was designed to facilitate an automatic algorithm evaluation process. It is used to execute an algorithm implementation on the given predefined sets of test cases and to analyze various indicators of the execution. Within every project of the system user can define the problem to be solved, sets of test cases, parameters of the input and indicators of the output data and the criteria for the algorithm quality evaluation. When a project is defined, any number of algorithm implementations (programs) can be added. When requested, system executes all the implemented algorithms, checks the correctness and compares the quality of their results. Using the ALGator user can add additional quality criteria, draw graphs and perform evaluations and comparisons of defined algorithms.

1.1 User roles

The ALGator can be used by users with different roles. An unauthenticated user (guest) can execute only those actions that do not change the system, which basically means that such user can only view the public data. For all the other

activities user must be logged into a system and must have one of the following roles.

System administrator.

The system administrator installs and manages the whole system (the software and hardware part), and has the access to all the resources of the system.

Project administrator.

The project administrator defines the project by a) implementing the predefined java or C++ interfaces that describe the problem and the structure of the algorithm; b) defining sets of test cases on which algorithms will be executed, and c) characterizing the format of the input and the output of algorithms (i.e. defining the parameters of the input and indicators of the output). Project administrator has an access to all the project resources. If the project is made public, project data can be seen by all users, while private projects can be seen only by project and system administrators.

Researcher.

The researcher defines an algorithm within the selected project, runs predefined tests and compares the results with the results of other algorithms. Public algorithms can be seen by every user while the private algorithms can only be seen the owner (i.e. researcher) and the project administrator.

1.2 A typical use case

A typical way of using the ALGator is as follows:

- The system administrator prepares the system by providing the hardware, installing the ALGator software packages, and publishing the internet address of the installed system.
- The project administrator adds a new project and defines all the project's properties. When the project is completely defined and declared as public, the ALGator automatically generates an internet subpage with the project presentation and usage guide sections.
- The project administrator adds some state of the art algorithms for solving the problem of the project, which will be used as a reference for the evaluation process (i.e. the results of the algorithms added by researchers will be compared with the results of these referential algorithms).
- According to the rules, presented at the project's website, the researcher adds a new algorithm. The ALGator will automatically run the new algorithm on prede-

fined tests. The researcher then checks the correctness and compares the results of his algorithm with the results of the other algorithms defined in the project. The researcher can also decide to make the algorithm public (by default, the algorithms are private).

- The guest of the system lists the results and prints the graphs and other data produced by the ALGator. Guest can also perform some actions (like customization of the presentation) that do not alter the project configuration.

2. PROJECT DEFINITION

The main task of the project administrator is to provide the configuration files and to implement corresponding java or C++ interfaces. Besides the definition of the output format (where the sequence of the parameters and indicators in output file is described), the test cases, the test sets and the algorithm structure has to be defined precisely.

The test cases and the test sets

A test case in the ALGator execution environment is defined by a subclass of a `TestCase` class, which contains data structures to hold the test case data. Since these data structures are project-specific (i.e. each problem needs data of its own type) the project administrator has to implement the `[Project]TestCase` class and prepare the data structures. For example, in the data-sorting problem, the `SortTestCase` class could be defined as follows.

```
public class SortTestCase extends TestCase{
    // An array of data to be sorted
    public int [] arrayToSort;
}
```

A test set contains one or more test cases and it is a minimal execution unit. Test set is defined by a single text file in which every line defines one test case. The format of these lines is project-specific and it is defined by a project administrator. If required, additional files can be used to specify the cases. Again, the syntax and the semantics of the content of these files is defined by a project administrator. The following presents an example of the text file defining five test cases for the data-sorting problem.

```
test1:10000:RND
test2:20000:RND
test3:30000:RND
test4:40000:FILE:numbers.txt:12540
test5:50000:FILE:numbers.txt:16534
```

To iterate the test set of a given project (i.e. to read the lines of the text file, to parse and interpret their meaning and to generate a test case for each line) the ALGator uses the `AbstractTestSetIterator` class. The main task of this class is to provide test cases (as `[Project]TestCase` classes) one by one. The `AbstractTestSetIterator` class contains the following methods: `void hasNext()` (returns `true`, if there are some tests left in the test set), `void readNext()` (reads the next test case and stores it in internal data structures) and `TestCase getCurrent()` (returns the last test case read by the `readNext()` method). Since the representation of test cases is project-specific, project administrator has to provide

the correct implementation of the `getCurrent()` method. All the other methods are general and they can be used without modification.

```
public class SortTestSetIterator
    extends DefaultTestSetIterator{
    public TestCase getCurrent() {
        String [] fields = inputLine.split(":");

        probSize = Integer.parseInt(fields[1]);
        String group = fields[2];
        int [] array = new int[probSize];

        switch (group) {
            case "RND":
                Random rnd = new Random();
                for (i = 0; i < probSize; i++)
                    array[i] = rnd.nextInt(1000);
                break;
            //...
        }
        SortTestCase tCase = new SortTestCase();
        tCase.arrayToSort = array;
        return tCase;
    }
}
```

Algorithms

The “heart” of the each project are the implemented algorithms. Each algorithm is represented by a subclass of the `AbsAlgorithm` class with the following methods:

`ErrorStatus init(TestCase test)`. This method takes care for the input of the algorithm; it reads the test case and prepares the data. To enable fast algorithm execution all expensive initial tasks have to be done in this method. When this method is done all the required algorithm’s input data has to be prepared in a proper format.

`void run()`. In this method the `execute(...)` method is called. The parameters of the `execute()` method are project-specific and are provided by project administrator. The ALGator takes the time of the execution of the `run()` method as an algorithm execution time therefore nothing else as the `execute()` method call should be placed in the `run()` method body.

```
public void run() {
    execute(sortTestCase.arrayToSort);
}
```

`ParameterSet done()`. This method collects all the parameters and indicators of the execution and prepares them in the form suitable to be written in to the output file.

The `AbsAlgorithm` class is abstract and the project administrator has to provide the `[Project]AbsAlgorithm` subclass with the above mentioned methods implemented. Besides he has to declare fields for input data (in these fields the input data obtained from the test case will be stored during the execution of the `init()` method) and the `abstract`

`execute()` method with appropriate number and type of parameters. The task of the researcher is to implement a subclass of `[Project]AbsAlgorithm` and implement the `execute(...)` method. In other words, all the “dirty job” of preparing data and collecting the results is done by the project administrator. The researcher who wants to provide an algorithm only has to implement one method which returns a proper result. In the case of data-sorting problem, an algorithm only needs to sort the array of data; a very simple (but technically correct) algorithm that can be executed in the ALGator is as follows.

```
public class JavaSortAlgorithm
    extends SortAbsAlgorithm {
    public void execute(int [] data) {
        Arrays.sort(data);
    }
}
```

3. INDICATORS OF THE ALGORITHM

Since the ALGator was designed to be used for various kinds of problems, the criteria for measuring the quality of algorithms are not defined as a part of the system but they have to be defined by the project administrator. The current version of the system enables measurements of the three different kinds of indicators: a) the indicators to measure the speed and the quality of the algorithm (the so called EM indicators), b) the project-specific counters to count the usage of the parts of the algorithm’s program code (the so called CNT indicators), and c) the counters of the java byte code usage (the so called JVM indicators). These indicators are calculated with independent measurements that are performed as separated tasks so they do not interfere with one another. For example: when the ALGator measures time, the CNT and JVM indicators are disabled. To perform the JVM measurements a dedicated java virtual machine is used.

The EM measurements.

These measurements are used to measure the time and other project-specific metrics. All measurements of the time are performed automatically. To provide as accurate time indicators as possible the ALGator tries to reduce the influence of the uncontrolled computer activities (e.g. sudden increase of a system resource usage) by running each algorithm several times. The system measures the first, the best, the worst and the average time of the execution. The project administrator only needs to specify the phases of algorithm execution (e.g. the pre-processing phase, the main phase, the post-processing phase, ...) and to select which of the time indicators are to be presented as the result of execution. The project-specific indicators are defined by the project administrator. They can be presented as a string or as a number. For example, for exact algorithms, the value of an indicator could be “OK” (is the algorithm produced the correct result) or “NOK” (is the result of the algorithm is not correct). For approximation algorithms the value of an indicator could be the quality of the result (i.e. the quotient of the correct result and the result of the algorithm).

The CNT measurements.

The CNT measurements are used to count the usage of the parts of the program code. This option is used to analyze the usage of a certain system resource or to count the usage

of selected type of commands on the programming language level. Using this one can, for example, measure how many times the memory allocation functions were executed during the algorithm execution and the amount of the memory allocated by these calls. One can also use CNT measurements to detect which part of the algorithm is most frequently used. For example, if the problem in concern would be the data-sorting, using the CNT measurements one could count the number of comparisons, the number of swaps of elements and the number of recursive function calls (which are the measures that can predict the algorithm execution behavior [4]). To facilitate the CNT measurement in the project, the project administrator has to define the names and the meaning of the counters and the researchers have to tag the appropriate places in their code. Everything else is done automatically by the ALGator.

The JVM measurements.

The algorithm written in the java programming language compiles into the java byte code. An interesting option offered by the ALGator is the ability to count how many times each byte code instruction was used while execution the algorithm on a given test case. To facilitate this option the ALGator uses a dedicated java virtual machine which was developed as a part of the ALGator project [2, 3]. Besides counting the usage of each byte code this virtual machine also records the data about the memory usage. In [1] Lambert and Power indicated that the frequency of the usage of each byte code instruction can be used to predict the execution time. Even though the ALGator’s ability to count the byte code instructions usage is quite young, we expect that the data produced by the JMV measurements could be useful not only for the quantitative but also the substantive analysis of the algorithms.

4. ANALYZING THE RESULTS

As a result of the algorithm execution the ALGator produces the text output files. For each tuple (algorithm, test set, measurement) one file is created; each line in this file contains parameters and indicators of one test case.

ID	Testset	TestID	Pass	N	JHoare.Tm...	JWirth.Tmin
1	TestSet3	Test-1	DONE	10000	740	772
2	TestSet3	Test-2	DONE	10000	768	788
3	TestSet3	Test-3	DONE	10000	753	768
4	TestSet3	Test-4	DONE	10000	760	771
5	TestSet3	Test-5	DONE	10000	750	807
6	TestSet3	Test-6	DONE	15000	1160	1181
7	TestSet3	Test-7	DONE	15000	1152	1182
8	TestSet3	Test-8	DONE	15000	1150	1203
9	TestSet3	Test-9	DONE	15000	1144	1245
10	TestSet3	Test-10	DONE	15000	1166	1199
11	TestSet3	Test-11	DONE	20000	1594	1583
12	TestSet3	Test-12	DONE	20000	1598	1673

Figure 2: An example of data query with result.

The data in the output line is separated by semicolons (CSV format). For efficient work with this data ALGator provides the analyzer with its own query language and with the visualization module for presenting data as graphs. For ex-

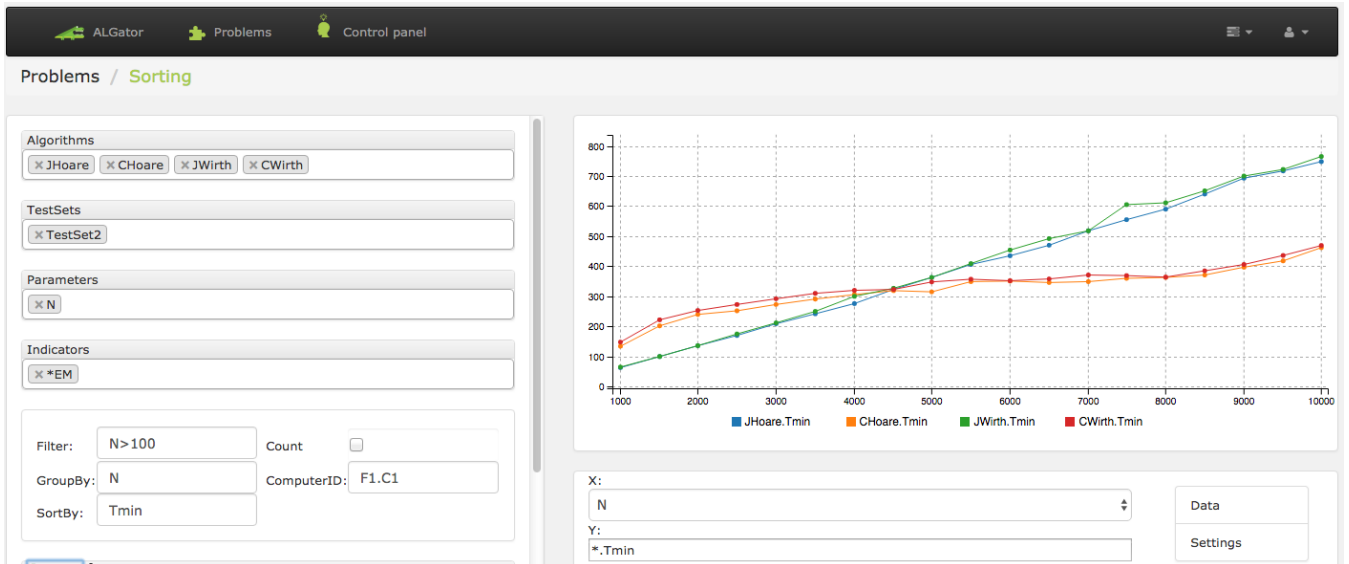


Figure 1: The visualization module of the ALGator.

ample, to get the minimal execution times for algorithms named JHoare and JWirth on the test set called TestSet3, user can run query as depicted in Figure 2.

The ALGator query language is a powerful tool that enables all sorts of data manipulation. An example of a complex query to calculate the quotient of minimal times for the JHoare algorithm running on two different computers (F1.C1 and F1.C2) is presented below.

```
queryF1C1 = FROM TestSet0
  WHERE (algorithm=*) AND ComputerID=F1.C1
  SELECT Tmin AS A1;
queryF1C2 = FROM TestSet0
  WHERE (algorithm=*) AND ComputerID=F1.C2
  SELECT Tmin AS A2;
FROM queryF1C1, queryF1C2
  WHERE (algorithm=JHoare)
  SELECT N, A1/A2 AS Q
```

The visualization module the ALGator can be used to produce graphs as depicted in Figure 1.

5. CONCLUSION

The execution part of the ALGator was developed in both java and C++ programming languages, therefore the algorithms to be tested could be implemented in one of these two languages. Measuring the exact execution time of the algorithms written in java is a challenging task since the system can only measure real time and because there is no way to eliminate the side effects of the java virtual machine's background tasks (e.g. garbage collection). To overcome this problem, the ALGator executes each algorithm several times and reports the first, the minimal, the maximal and the average time of execution. Comparing and analyzing these times one can detect the influence of the execution environment to the overall execution time. In many cases this influence is negligible. Having the java implementation of the algorithm also has some benefits. Namely, the ALGator counts and generates the statistics of the usage of the java byte code

instructions. As stated in [1] these statistics provide enough information to be used for the platform independent timing of the algorithms. Our preliminary tests indicate a great correlation between the number of used java byte code instructions (multiplied by the corresponding weight depending on the type of instruction) and the execution time.

The ALGator is a testing environment, which aims to make the testing process as easy as possible for both, the project administrators and for the researchers. We tried to minimize the effort that has to be used to prepare the project and to prepare the algorithm and we think that this goal was achieved. The biggest challenge for the project administrator is to prepare adequate test cases and to write several lines of java or C++ code (in an average case not more than about 100 lines of code), while the researcher has to write only a few lines of code to call the existing java or C++ implementation of the algorithm. All the other tasks needed to execute the algorithm and to produce the desired indicators are performed automatically by the ALGator, therefore the researchers can focus on the analyses of the results. Furthermore, ALGator uses the same test cases for all the algorithms of the project, therefore the researchers can not tailor the tests to be optimal for their implementations, which makes the results of the evaluation fair and reliable.

6. REFERENCES

- [1] J. M. Lambert and J. F. Power. Platform independent timing of java virtual machine bytecode instructions. *Electronic Notes in Theoretical Computer Science*, 220:79–113, 2008.
- [2] J. Nikolaj. Predelava javanskega navideznega stroja za štetje ukazov zložne kode. *Univerza v ljubljani, Fakulteta za računalništvo in informatiko, diplomsko delo*, 2014.
- [3] J. Nikolaj. Vmep. github.com/nikolai5slo/jamvm, 2014.
- [4] R. Sedgewick. The analysis of quicksort programs. *Acta Informatica*, 7:327–355, 1977.

A Graph to the Pairing strategies of the 9-in-a-row Game*

Lajos Gyórfy[†]
Institute of Mathematics
H-6701 Szeged, Hungary
lgyorffy@math.u-
szeged.hu

András London
Institute of Informatics
H-6701 Szeged, Hungary
london@inf.u-szeged.hu

Géza Makay
Institute of Mathematics
H-6701 Szeged, Hungary
makayg@math.u-
szeged.hu

ABSTRACT

In Maker-Breaker positional games two players, Maker and Breaker, are playing on a finite or infinite board with the goal of claiming or preventing to reach a finite winning set, respectively. For different games there are several winning strategies either for Maker or Breaker. One class of winning strategies are the so-called pairing strategies. Generally, a pairing strategy means that the possible moves of a game are paired up; if one player plays one, the other player plays its pair. In this study we describe all possible pairing strategies for the 9-in-a-row game. Furthermore, as a concept, we define a graph of these pairings in order to find a structure for them. The characterization of that graph will be also given.

Categories and Subject Descriptors

F.2 [Analysis of algorithms and problem complexity]: Nonnumerical Algorithms and Problems; G.2 [Discrete mathematics]: Graph Theory, Combinatorics

Keywords

Positional games, pairing strategies, Hales-Jewett pairing

1. INTRODUCTION

In this work, we study the pairing strategies of the 9-in-a-row Maker-Breaker game. Hales and Jewett [7] gave the first pairing strategy to this game showing Breaker's win. However, the uniqueness of the Hales-Jewett pairing or other examples had not been provided since then, until Gyórfy et al. [6] showed the following. There exist only 8- and 16-toric pairings (i.e. they are simply the repetitions of a pairing on the 8×8 and 16×16 square grids, respectively) where all 16-toric ones can be derived from some 8-toric ones.

*This work was partially supported by the National Research, Development and Innovation Office - NKFIH, SNN-117879.

[†]Corresponding author

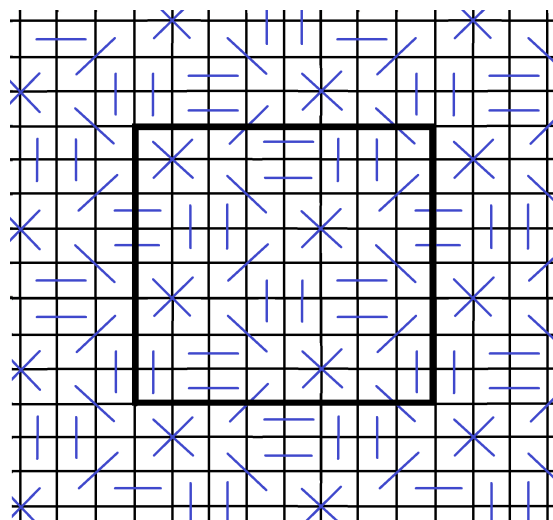


Figure 1: Hales-Jewett pairing blocks the 9-in-a-row

After recalling positional games and pairing strategies in general, we focus on the 9-in-a-row game and its pairings. We provide a computer program which generates and distinguish all (194543) different 8-toric pairings. Finally, we create and analyze a graph of these pairings to have a structure of them.

1.1 Positional games

A positional game can be defined as a game on a hypergraph $\mathcal{H} = (V, E)$, where $V = V(\mathcal{H})$ and $E = E(\mathcal{H}) \subseteq \mathcal{P}(V) = \{S : S \subseteq V\}$ are the set of vertices and edges, respectively. Usually, V can be finite or infinite, but an $A \in E$ edge is always finite. The first and second players take elements of V in turns. In the **Maker-Maker** (M-M) version of the game, the player who first takes all elements of some edge $A \in E$ wins the game. In contrary, in the **Maker-Breaker** (M-B) version, Maker wins by taking every element of some $A \in E$, while the other (usually the second) player, Breaker, wins by taking at least one vertex of every edge in E . Clearly, there is no draw in this game. The M-M and M-B games are closely related, since if Breaker wins as a second player, then the M-M game is a draw. On the other hand, if the first player has a winning strategy for the M-M game, then Maker also wins the M-B version. For more on these, see Berlekamp, Conway and Guy [3] or Beck [2].

In this work we deal with the hypergraph of the k -in-a-row game, which is defined as follows.

Definiton 1. The vertices of the **k -in-a-row hypergraph** \mathcal{H}_k are the squares of the infinite (chess)board, i.e. the infinite square grid. The edges of the hypergraph \mathcal{H}_k are the k -element sets of consecutive squares in a row horizontally, vertically or diagonally. We refer to the whole infinite rows as *lines*.

For k -in-a-row M-B games Maker wins if $k \leq 5$, see Allis et al. [1] and Breaker wins if $k \geq 8$, see Zetters [5]. There is a Breaker winning pairing strategy only if $k \geq 9$, see Csernenszky et al. [4]. For the case of $k = 9$ the first pairing strategy found by Hales and Jewett can be seen on Fig. 1. For $k = 6, 7$ the problem is open.

1.2 Pairing strategies

Given a hypergraph $\mathcal{H} = (V, E)$ and a bijection $\rho : X \rightarrow Y$, where $X, Y \subset V(\mathcal{H})$, $X \cap Y = \emptyset$, is a **pairing** on the hypergraph \mathcal{H} . An $(x, \rho(x))$ pair **blocks** an $A \in E(\mathcal{H})$ edge, if A contains both elements of the pair. If the pairs of ρ block all edges, we say that ρ is a **good pairing** of \mathcal{H} .

Pairings are one way to show that Breaker has a winning strategy in positional games. A good pairing ρ for a hypergraph \mathcal{H} can be turned to a winning strategy for Breaker in the M-B game on \mathcal{H} . Following ρ on \mathcal{H} in a M-B game, for every $x \in X$ chosen by Maker, Breaker chooses $\rho(x)$ or vice versa in case of $x \in Y$ (if $x \notin X \cup Y$ then Breaker can choose an arbitrary vertex). Hence Breaker can block all edges and wins the game. Hereafter we focus on the 9-in-a-row game and its pairings.

2. PAIRINGS FOR 9-IN-A-ROW

Definiton 2. A pairing is a **domino pairing** on the grid, if all pairs consist of only neighboring cells (horizontally, vertically or diagonally).

Note that the pairing on Fig. 1 is a domino pairing. From Györfy et al. [6] it follows that if there is a good pairing for \mathcal{H}_9 then this pairing is a domino pairing in which the dominoes are following each other by 8-periodicity in each line and all squares are covered by a pair. To handle the periodicity we define the concept of k -toric pairings.

Definiton 3. A pairing of the infinite board is **k -toric** if it is an extension of a $k \times k$ square, where k is the smallest possible.

In [6] it was proved that a good pairing of \mathcal{H}_9 is either 8-toric or 16-toric. Furthermore, all 16-toric pairings can derive from two (or more) 8-toric pairings. Fig. 2 shows a 16-toric (but not 8-toric) good pairing. The four 8×8 squares differs from each other only in the colored squares, where the bold pairs show the actual pairs and what the thin line shows is the pairing of the other 8×8 square. From now we only deal with the 8-toric pairings of \mathcal{H}_9 . A good 8-toric pairing is uniquely determined by an 8×8 section of it, by

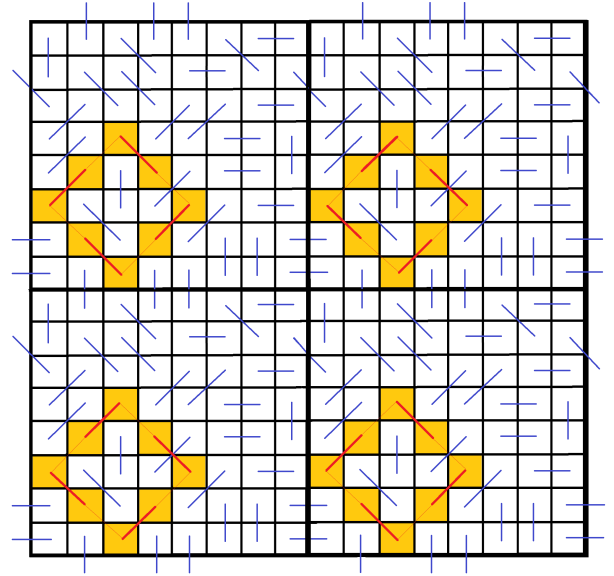


Figure 2: 16-toric pairing for the 9-in-a-row game

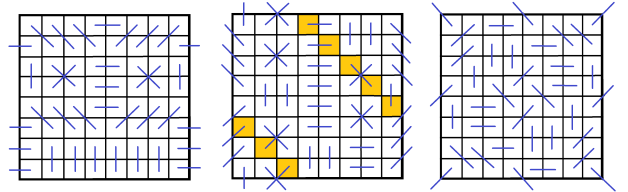


Figure 3: Other 8-toric examples

definition. Furthermore, that 8×8 section contains exactly one pair in each 32 (eight vertical, eight horizontal and 16 diagonal) torus lines. Three examples, other then the Hales-Jewett pairing, can be seen on Fig. 3. A diagonal torus line is colored on the middle one.

2.1 Generate pairings

To find all possible 8-toric pairing strategies of \mathcal{H}_9 on the infinite board we wrote a computer program that will be introduced in this section. The main challenge here is not only finding all pairings, but deciding whether two pairings are the same.

We store a pairing in the 8×8 table such that each cell represents the actual pair of the cell according to the 8 possible pairs: 0 means East, 1 South-East, and so on, 7 North-East. Naturally, if a cell's pair is on the East, then its pair has its own pair on the West, i.e. we fill the table two cells at a time. The algorithm itself is the usual backtracking algorithm: we find possible pairs for the next cell in the table having no pair so far, try all those by recursively calling the table filling function. While checking whether a pair is possible, we also make sure that there can be no overblocking, so we keep track of the blocked edges. A detailed example can be found in webpage [8].

From previous experiences we know, that the running time

is crucial, since there are too many such pairings. We try to reduce the number of cases to be considered. We consider two pairing strategies on the infinite board to be the same, if they can be transformed into each other by translation, mirroring and rotation. Thus, in order not to find the same pairing several times, we apply all transformations for any pairing found on the 8×8 table. From these transformed pairings we select the smallest one with respect to the lexicographical order. That also means that such a pairing must start with 0 and 4 in the first row of the 8×8 table, so we can also reduce the number of searched cases by starting fill the table with these two numbers. Naturally, we keep in mind that the 8×8 table is expanded in (say) an 8-toric way to the whole infinite board while applying these transformations. More precisely:

1. We either mirror or not (2 possible cases) the table to the vertical line between columns 4 and 5.
2. We rotate the table by 0, 90, 180, and 270 degrees (4 possible cases).
3. We try all toric (that is, modulo 8) translation that results in a table starting with 0 and 4.

We select the lexicographically smallest table as a representative for the actual pairing. This method reduce the number of all pairing checked to 6210560, and the program found the 194543 different pairings in about 4 minutes on a desktop computer with a 3.2 GHz Core i7 processor using 12 Mb of memory. The pairings themselves can be downloaded at the page [8]. Interestingly, the number of the different pairings turns out to be a prime number.

Since we have such many different pairings, an obvious way to find a structure can be to store the pairings in a graph. In the next section we will show a natural method to find connections between pairings.

2.2 Graph of pairings

While trying to find pairings by hand one can observe, that we can move a pair along the blocked edge by one step to create a new pairing using the following method.

1. Move the first pair on the table. This move creates a cell (say A) without a pair, and another cell (say B) with two pairs.
2. Move the pair containing cell B which was not the just moved pair so that cell B has one pair after the move. But then another cell may have two pairs.
3. Repeat step 2 as long as it creates a cell with two pairs.
4. This method will end when the last move creates a new pair for cell A , which had no pair before the move.

Naturally, we should keep in mind that we are on an 8-toric pairing and move the pairs accordingly. Since we are on a finite table, this method will either end at step 4, or create a repeating cycle. But the later one is not possible. Note that cell A cannot be part of the cycle, as it has no pair, and

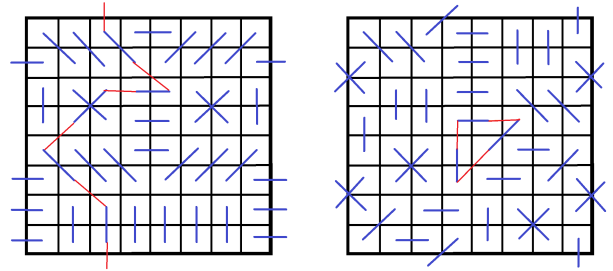


Figure 4: Two examples of connections between pairings

it would break the repetition. The first move that entered the cycle creates a hole “behind” (outside the cycle), and when the cycle comes to the same cell, the pair will move backwards, and the cycle is not entered again. Also, it is easy to see that we get an optimal pairing by this method. Since the original pairing was optimal, moving a pair (an 8-toric way) along the blocked edge keeps that direction (i.e. 8 edges) blocked. Since the method ends in step 4, there are no cells without a pair. We also move the pairs on a torus, so no overblocking is possible. We say that two pairings are connected, if one can obtain the second pairing from the first one by the method described above (of course, we consider only different pairings as it was defined in the previous section). This relation is symmetric: moving back the last pair of the above method gives back the first pairing from the second one. This creates a graph, where the vertices are the pairings and the edges are defined by the moving transition. Fig. 4 shows two examples for this moving transition. In both cases, the first pairing contains only the blue pairs, and the red dominoes show the transition to the other pairing. After computing all possible different pairings our program can easily find this graph. It tries to move all pairings (by trying to free up each cell in the 8×8 board), and use the method described in the previous section to find the lexicographically smallest representative for the new pairing. It takes about 1 minute to finish this task on the same hardware as in the previous section.

In the next section we will investigate the properties of the obtained graph.

2.3 Analyzing the graph

The basic parameters of the obtained graph can be seen in Tab. 1. The graph is not connected, which means that repeating the moving transition described in the previous section we cannot reach an arbitrary pairing from another. One of the 14 components of the graph is a giant component containing almost all (194333) vertices. The diameter of this component is 34, which shows us that even this giant component does not seem to be a “small-world” network. There are 5-5 smaller components of 10 and 16 vertices and 1-1 components of size 6, 26, 48. Note that every graph component containing 16 vertices is the net of a 4-dimensional cube. Fig. 5 shows some small components.

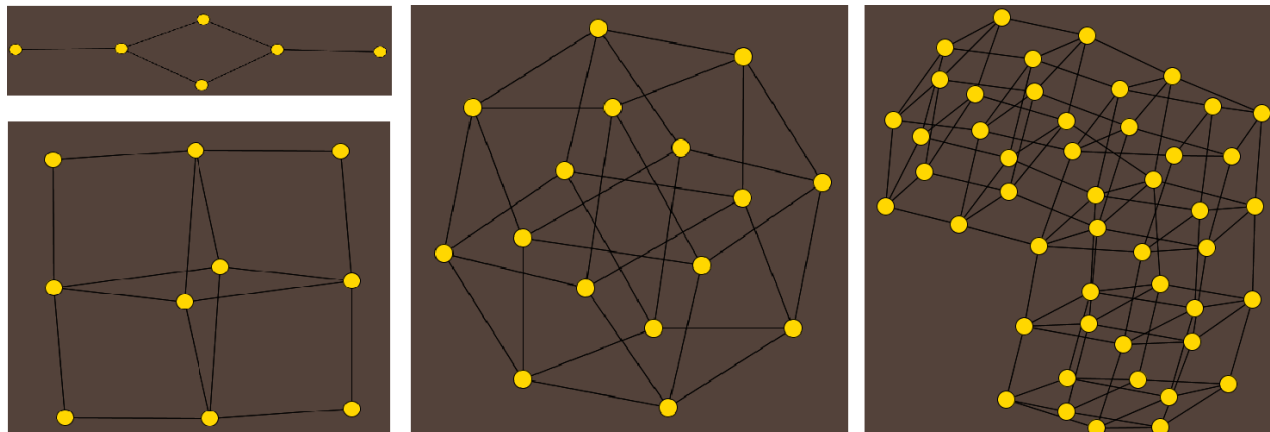
The graph is triangle-free, moreover, the length of all induced cycles is four. The degree distribution of the graph can be seen in Tab. 2.

Table 1: Basic parameters of the constructed graph

vertices	edges	#components	max degree	min degree	avg. degree
194543	532107	14	11	1	5.47

Table 2: Degree distribution of the graph

1	2	3	4	5	6	7	8	9	10	11
17	392	395	39811	66185	53222	25309	7547	1472	183	10

**Figure 5: Some components of the obtained graph**

3. CONCLUSIONS

In this study, we investigated the 9-in-a-row Maker-Breaker positional game focusing on its pairing strategies which guarantee Breaker's win. We found all different 8-toric pairing strategies using a computer program. The main concepts of the program were described in detail. In order to find a structure of the 194543 pairings, we arranged them into a graph where the vertices are the pairings itself and the edges are some moving transitions of pairs. Analyzing the graph and calculating standard parameters may help in a better understanding of pairing strategies in general.

4. REFERENCES

- [1] L. V. Allis, H. J. van den Herik and M. P. Huntjens. Go-Moku solved by new search techniques. *Proc. 1993 AAAI Fall Symp. on Games: Planning and Learning, AAAI Press Tech. Report FS93-02*, pp. 1-9, Menlo Park, CA.
- [2] J. Beck. *Combinatorial Games, Tic-Tac-Toe Theory*. Cambridge University Press, 2008.
- [3] E. R. Berlekamp, J. H. Conway and R. K. Guy. *Winning Ways for your mathematical plays*, Volume 2. Academic Press, New York, 1982.
- [4] A. Csernenszky, R. Martin and A. Pluhár. On the Complexity of Chooser-Picker Positional Games. *Integers* 11, 2011.
- [5] R. K. Guy and J. L. Selfridge, Problem S.10, *Amer. Math. Monthly* **86** (1979); solution T.G.L. Zetters **87** (1980) 575–576.
- [6] L. Györfy, G. Makay and A. Pluhár. Pairing strategies for the 9-in-a-row game. Submitted, 2016. http://www.math.u-szeged.hu/~lgyorffy/predok/9_pairings.pdf downloaded: 28. 08. 2016.
- [7] A. W. Hales and R. I. Jewett. Regularity and positional games. *Trans. Amer. Math. Soc.* **106** (1963) 222–229; M.R. # 1265.
- [8] G. Makay. Personal homepage <http://www.math.u-szeged.hu/~makay/amoba/> downloaded: 06. 04. 2016.

Construction of orthogonal CC-set

Andrej Brodnik
University of Primorska
Koper, Slovenia
andrej.brodnik@upr.si

Vladan Jovičić
Ecole Normale Supérieure
Lyon, France
vladan94.jovicic@gmail.com

Marko Palangetić
University of Primorska
Koper, Slovenia
palangeticmarko95@hotmail.com

Daniel Siladi
University of Primorska
Koper, Slovenia
szilagyi.d@gmail.com

ABSTRACT

In this paper we present a graph-theoretical method for calculating the maximum orthogonal subset of a set of coiled-coil peptides. In chemistry, an orthogonal set of peptides is defined as a set of pairs of peptides, where the paired peptides interact only mutually, and not with any other peptide from any other pair.

The main method used is a reduction to the maximum independent set problem. Then we use a relatively well-known maximum independent set solving algorithm which turned out to be the best suited for our problem. We obtained an orthogonal set consisting of 29 peptides (homodimeric and heterodimeric) from initial 5-heptade set. If we allow only heterodimeric interactions we obtain an orthogonal set of 26 peptides.

Keywords

Algorithms, NP-hard problem, Modeling

1. MOTIVATION

In the last 30 years, impressive 3D structures have been built using DNA, in a field called DNA origami. Complex structures built from proteins would have many advantages, since amino acids provide much more functionality. The main problem is that the simple Watson-Crick pairing rules present in DNA have no simple analogue for proteins. Using a special class of polypeptides, called coiled-coil polypeptides, the orthogonal binding rules of DNA can be emulated. By specifying only the primary structure of those polypeptides (the order of amino acids), complex 3D structures can be built, such as the recent protein tetrahedron [2]. More specifically, that structure is determined by taking the wireframe of the desired object, doubling every edge, and performing an Euler traversal of the obtained graph. Then,

we know that the peptides associated with edges that were initially parallel must bind, and all others must not.

Essential for such designs is that each pair of peptides interacts only mutually, and not with any other pair. Thus, the notion of an *orthogonal set* is introduced. Obviously, the greater our orthogonal set is, the more complex are the structures we can create. Currently the limiting factor in designing larger structures is the small set of available peptides.

In this paper, we describe a method for determining a maximal orthogonal set, from a given set of admissible peptides. Also, in section 6 we present a possible approach for extending an already-calculated orthogonal set.

2. PROBLEM DESCRIPTION

As input we are given a set of peptides $P = p_1, p_2, \dots, p_n$ (their primary structures – given as strings of fixed length) and interaction matrix I . If $I_{i,j} = 1$, then p_i interacts with p_j and if it is 0 they do not interact. We have to construct a set of pairs S , where $(p_i, p_j) \in S$, iff $I_{i,j} = 1$ and for all other p_k that are in any pair of S $I_{i,k} = 0$. Moreover, if $i = j$ in (p_i, p_j) we are talking of *homodimer* and otherwise of *heterodimer*.

We can model this problem as a graph-theoretical one: We create an undirected graph $G = (V, E)$ where V is set of peptides P , and the edge set E containing an edge $p_i p_j$ (or a loop at p_i , denoted by $p_i p_i$) if and only if p_i and p_j interact. Therefore, the problem definition is the following:

DEFINITION . [Maximum Independent Set of Pairs (MISP)]
Let $G = (V, E)$ be undirected graph and let k be positive integer. Does there exist set $S \subseteq E$ such that for $u_1 v_1, u_2 v_2 \in S$

$$\{u_1, v_1\} \cap \{u_2, v_2\} = \emptyset,$$

$$\{\{u, u_1\}, \{u, v_1\}, \{v, u_1\}, \{v, v_1\}\} \cap E = \emptyset$$

and $|S| > k$?

3. HARDNESS OF THE PROBLEM

In order to determine the best possible solution of our problem, in this section we will prove that MISP is NP-complete.

THEOREM 1. *[[Maximum independent set of pairs is NP-complete.*

Algorithm 1 NP certifier

PROOF. 1: $S \leftarrow$ given set of pairs
2: **if** $|S| < k$ **then**
3: **return No**
4: **for** $u_1v_1 \in S$ **do**
5: **for** $u_2v_2 \in S - u_1v_1$ **do**
6: **if** $u_1u_2 \in E \vee u_1v_2 \in E \vee v_1u_2 \in E \vee v_1v_2 \in E \vee u_1v_1 \notin E$ **then**
7: **return No**
8: **return Yes**

It is easy to check that Algorithm 1 is a polynomial certifier for MISP. Now we will reduce the *independent set* problem to MISP in order to show that MISP is NP-hard.

Let $G = (V, E)$ be a graph. We want to check if there exists an independent set of size greater than k . Define a new graph $G' = (V', E')$ as follows. Initially, let $V' = V$ and $E' = E$. Then, for each vertex $v \in V$ add another vertex v' (twin vertex) to V' and add the edge vv' to E' .

LEMMA 1. *[[Every maximal independent set of pairs consists only of the edges of the form vv' .*

PROOF. Let S be a MISP in G' . Suppose the contrary, i.e. there is a pair $uv \in S$ which is not of the form wv' for $w \in V$. Then, for all $u_1v_1 \in S$ we have $u_1u \notin E'$, $u_1v \notin E'$, $v_1u \notin E'$, $v_1v \notin E'$. Then we can delete the pair uv from S and add pairs uu' and vv' where u' and v' are twin vertices of u and v , respectively. We can do this since the only neighbors of u' and v' are u and v , respectively. We obtained an independent set of pairs, with more more than $|S|$ elements, a contradiction. \square

We want to prove now that there is an independent set $|S| \geq k$ of G if and only if there is an independent set of pairs $|S_P| \geq k$ of G' .

(\Rightarrow) Suppose that S is independent set of G and $|S| \geq k$. Then, define the independent set of pairs S_P of G' on the following way:

$$S_P = \{vv' \mid v \in S\}.$$

It is easy to verify that this is independent set of pairs by the above definition. Then $|S_P| = |S| \geq k$.

(\Leftarrow) Suppose that S_P is an independent set of pairs of G' with $|S_P| \geq k$. Then, by previous lemma, we can define the following independent set S of G :

$$S = \{v \in V \mid vv' \in S_P\}.$$

By the construction of graph G' and by the lemma, one can show that S is a independent set of G . Then $|S| = |S_P| \geq k$ which completes proof that MISP is NP-hard.

Combining the NP-hardness with the earlier fact that MISP is in NP, we conclude that MISP is NP-complete

4. REDUCING MISP TO THE MAXIMUM INDEPENDENT SET

Now that we know that MISP is NP-complete, we can use one of the the vast number of algorithms already developed for solving various problems in NP, once we reduce MISP to that problem. The most natural choice is the maximum independent set problem.

Based on the MISP graph $G = (V, E)$, we construct a new graph $G' = (V', E')$, where $V' = E$, and two vertices are connected (in G') if and only if their corresponding edges in G share a common vertex or have two of their vertices connected by an edge. It is easy to see that finding an independent set in G' will give us an independent set of pairs, as per the definition in section 2. Moreover, due to our construction, an independent set of pairs in G also gives us a unique independent set in G' .

Thus, we have obtained a bijection between the independent sets of G' and the independent sets of pairs of G .

5. RESULTS

We use results from the previous section to solve the MISP of the input graph G which is constructed from the input set of peptides $P = p_1, p_2, \dots, p_n$ in several steps.

1. Based on previous work by [3], we calculate the interaction scores s_{ij} for each pair of peptides $p_i p_j$ (including homodimers $p_i p_i$), and store that matrix for the following steps
2. Choose thresholds t and T based on which we decide whether peptides p_i and p_j with interaction score s_{ij} will interact. If $s_{ij} < t$, we declare that p_i and p_j are not interacting (or, more precisely, interacting in a negligibly small proportion), and likewise, if $s_{ij} > T$, p_i and p_j interact. The greater the difference $T - t$, we are more certain that in the obtained orthogonal set only the designated pairs will interact.
3. Construct the graph G on the set of peptides by connecting the interacting ones, as in section 2.
4. Reduce G to G' , suitable for calculating the independent set, as in 4.
5. Find the maximum independent set in G' , as shown before, it corresponds to the MISP (or, orthogonal set) in G . We use the (exact) maximum clique solving algorithm presented in [1], which is based on greedy graph colorings – i.e. if we can color a particular subgraph with k colors, we know that that the maximum clique in that subgraph has size at most k .

In order to test our algorithm, we generated synthetic initial sets of peptides, based on two observations: Firstly, the interaction scoring function is designed to consider only 4 positions in each heptad. Secondly, using electrostatic arguments about individual amino acids and their positions in the coiled-coil, we reduced the variation even further, by allowing only 2 different amino acids on 3 of those 4 positions, and completely fixing the remaining amino acid. Thus, we

obtain 8 essentially different heptads, which we use to build up larger peptides. Our main result is the calculation of a 29-peptide orthogonal subset of the 5-heptad initial set (2^{15} peptides generated as described above), as well as a 26-peptide purely heterodimeric orthogonal subset of the same initial set. The interaction score heatmap can be seen on Figures 1 and 2

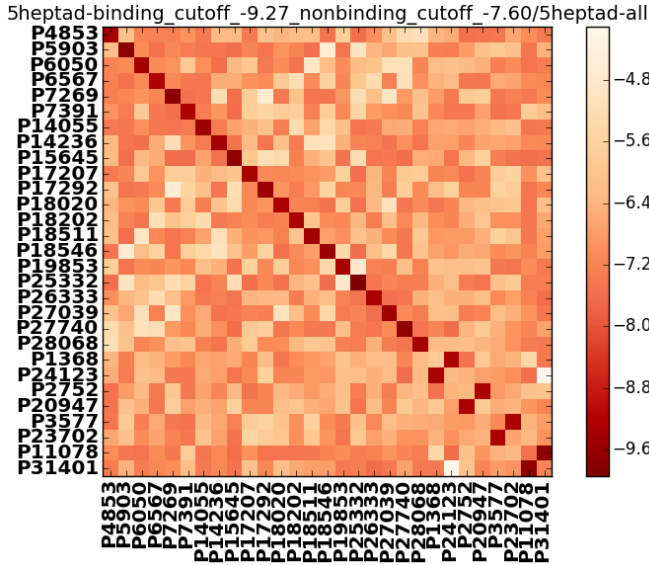


Figure 1: 5-heptad orthogonal set, no restriction

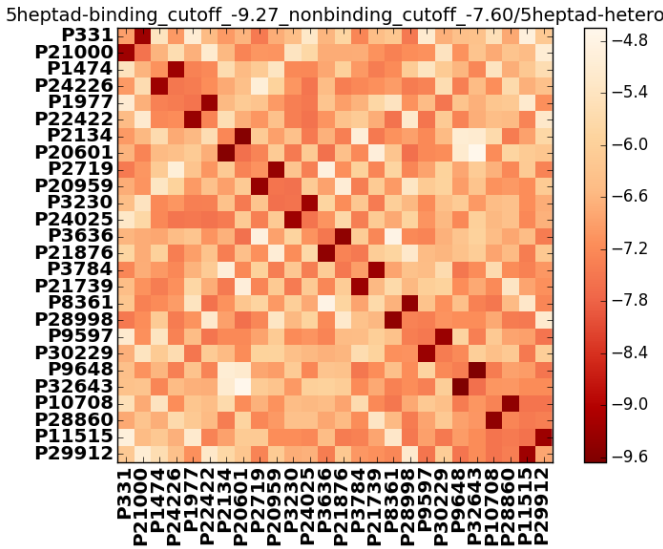


Figure 2: 5-heptad orthogonal set, heterodimers only

The peptides which belong to orthogonal set are in both figures colored in dark red.

6. FUTURE WORK

Up to now, we have only considered orthogonal sets derived from synthetically generated peptides, as described in the previous section. To actually use such an orthogonal set, we have to manually synthesize all of those peptides.

Alternative is to construct a maximal orthogonal set from the set of all natural tetraheptads (coiled-coils where each of the 4 heptads occurs naturally). Since there are 1171 known natural heptads, we can combine them to get $1171^4 = 1880301880081$ possible tetraheptads. Finding a maximal orthogonal subset of this set would require finding the maximum independent set of a graph with more than 10^{12} vertices – a task clearly impossible to do in a reasonable amount of time.

Our idea is to use heuristic to reduce the initial set to a more manageable size: Since it is possible to calculate the interaction matrix for single natural heptads, we can approximate scores for tetraheptads as shown at Figure 3. More specifically, we will add up the precalculated scores between (adjacent) heptads which are connected as on figure 3. Of course, some interactions will be left unaccounted for in the final score, for example the last amino acid in heptad 1 on 3 may interact with first amino acid of heptad 7 which is not added to the final score.

This observation enables us to construct more meaningful initial peptide sets consisting of longer peptides, based on the already-calculated orthogonal sets of shorter peptides.

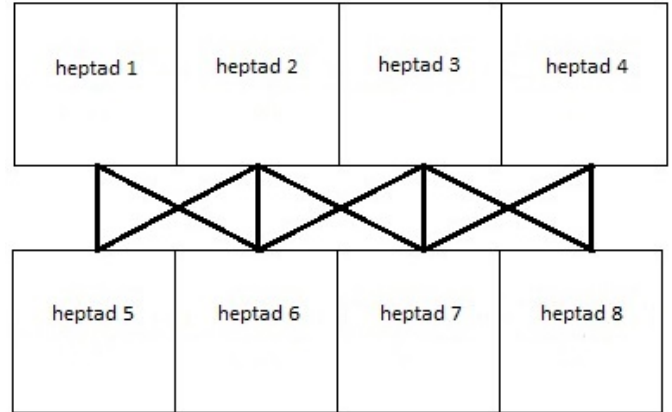


Figure 3: Proposed way of scoring

7. CONCLUDING REMARKS

In this paper, we presented an exact method for determining an orthogonal set of coiled-coil polypeptides, if we are given a numeric measure of their interaction strength. Our approach has been demonstrated to be successful for moderately large initial peptide sets (tens of thousands), and has given us optimal orthogonal sets that could not have been calculated by hand.

Unfortunately, for even larger initial sets, we are maximum-clique solver becomes an apparent bottleneck, as it has to operate on graphs of size $O(n^4)$, where n is the size of the initial set. In that case, we suggest investigating a bottom-up method described in the section 6.

8. REFERENCES

- [1] M. Depolli, J. Konc, K. Rozman, R. Trobec, and D. Janezic. Exact parallel maximum clique algorithm

- for general and protein graphs. *Journal of chemical information and modeling*, 53(9):2217–2228, 2013.
- [2] H. Gradišar, S. Božič, T. Doles, D. Vengust, I. Hafner-Bratkovič, A. Mertelj, B. Webb, A. Šali, S. Klavžar, and R. Jerala. Design of a single-chain polypeptide tetrahedron assembled from coiled-coil segments. *Nature chemical biology*, 9(6):362–366, 2013.
- [3] V. Potapov, J. B. Kaplan, and A. E. Keating. Data-driven prediction and design of bzip coiled-coil interactions. *PLoS Comput Biol*, 11(2):1–28, 02 2015.

Usage of hereditary colorings of product graphs in clique search programs

Matjaž Depolli
Department of Communication
Systems
Jožef Stefan Institute
matjaz.depolli@ijs.si

Janez Konc
Laboratory for Molecular
Modeling
National Institute of Chemistry
konc@cmm.ki.si

Sandor Szabo
Institute of Mathematics and
Informatics
University of Pecs
sszabo7@hotmail.com

Bogdan Zavalnij
Institute of Mathematics and
Informatics
University of Pecs
bogdan@ttk.pte.hu

ABSTRACT

There are computationally demanding problems that can be solved by k -clique search algorithms in auxiliary product graphs. The best clique search programs heavily rely upon good colorings. But obtaining a good coloring is a demanding task itself. We present some coloring schemes that exploit the property of the product graph itself and can be constructed with ease. We call these colorings hereditary. There are indications that using these colorings some hard problems would become feasible.

Keywords

clique, maximum clique, product graph, graph isomorphism

1. INTRODUCTION

Let $G = (V, E)$ be a finite simple graph. Let D be a subset of V and let Δ be the subgraph of G induced by D . The subgraph Δ is called a clique in G if any two distinct elements of D are adjacent in G . If the set D has k elements, then we call Δ a k -clique in G .

Finding cliques in a given graph is an important problem in discrete applied mathematics with many applications inside and outside of mathematics. For further details see [1], [2], [4], [7], [13], [14].

We formally state the following clique search problem.

PROBLEM 1. *Given a finite simple graph G and given a positive integer k . Decide if G contains a k -clique.*

Many practical clique search algorithms employ coloring to

speed up the computation by reducing the search space. Finding optimal or nearly optimal colorings is itself a computationally demanding problem. From this reason in the above computations computationally more feasible greedy algorithms are used to construct suboptimal colorings. It is customary to color the nodes of a graph G satisfying the following conditions.

1. Each node of G receives exactly one color.
2. Adjacent nodes in G cannot receive the same color.

This is the most commonly encountered coloring of the nodes of a graph and it is referred as legal coloring of the nodes. It is well known that coloring can be used for estimating clique size.

For each finite simple graph G there is a well defined non-negative integer k such that the nodes of G admit a legal coloring with k colors but the nodes of G cannot be colored legally using $k - 1$ colors. This number k is called the chromatic number of the graph G and it is denoted by $\chi(G)$.

Let us suppose that Δ is an l -clique in G and let us suppose that the nodes of G have a legal coloring with k colors. Then $l \leq k$ holds.

PROBLEM 2. *Given a finite simple graph G and given a positive integer k . Decide if the nodes of G have a legal coloring using k colors.*

Both Problems 2 and 1 are decision problems. From the complexity theory of computations we know that these problems belong to the NP-complete complexity class.

Let $G = (V, E)$ be a finite simple graph and let s be a positive integer such that $s \geq 2$. A subset U of V is called an s -free set if the graph spanned by U in G does not contain any s -clique. A partition U_1, \dots, U_r of V is called an s -clique free partition of V if U_i is an s -clique free subset of V for

each i , $1 \leq i \leq r$. We can look at this partitioning as an alternative coloring method and we will call it s -clique free coloring [12].

2. PRODUCT GRAPHS AND THEIR COLORINGS

In our paper we are interested in some special problems. We assume that these problems can be reduced to k -clique search in a given product graph. The problems of graph isomorphism and spanned subgraph isomorphism are representatives of this type of problems so we will use the spanned subgraph isomorphism problem to illustrate the method. Other problems can be dealt with by similar means. Spanned subgraph isomorphism has important applications for example in drug design, chemical database problems, artificial intelligence or pattern recognition. Let us state the spanned subgraph isomorphism problem more formally:

PROBLEM 3. *Let $G = (V, E), H = (V', E')$ be finite simple graphs. Is there a spanned subgraph G_0 in G such that G_0 is isomorphic to H . In other words is there a $G_0 = (V_0, E_0) : V_0 \subseteq V$ where $v_1, v_2 \in V_0$ and $\{v_1, v_2\} \in E$ then and only then $\{v_1, v_2\} \in E_0$ such that $G_0 \cong H$?*

A possible method of solving this problem is to construct an auxiliary graph $\Gamma = (W, F)$ where $|W| = |V||V'|$. The nodes of the graph Γ are labeled by ordered pairs of nodes from G and H . That is if $a_1 \in V, b_1 \in V'$ then $(a_1, b_1) \in W$. The edges of the graph Γ are constructed as follows. Let us consider $(a_1, b_1), (a_2, b_2)$ as two distinct nodes of Γ . We put an edge between them if $\{a_1, a_2\} \in E$ and $\{b_1, b_2\} \in E'$. We also put an edge between them if $\{a_1, a_2\} \notin E$ and $\{b_1, b_2\} \notin E'$ for $a_1 \neq a_2, b_1 \neq b_2$. This means that $\{a_1, a_2\}$ and $\{b_1, b_2\}$ both should be connected or both should not be connected. A k -clique where $k = |V'|$ in the graph Γ represents the function $f: V_0 \rightarrow V'$ such that $b_1 = f(a_1), b_2 = f(a_2), \{a_1, b_2\} \in E_0 \Leftrightarrow (f(a_1), f(a_2)) \in E'$.

It is well known, that the k -clique search algorithm can be sped up by using a good coloring of the given graph. We will describe several coloring schemes in the following subsections. We called these colorings “hereditary” because of the fact that they derive solely from the two input graphs and the constructing method of the auxiliary product graph.

2.1 First hereditary coloring scheme

Note that the nodes $(a_1, b_1), (a_1, b_2), (a_1, b_3), \dots, (a_1, b_{|V'|})$ of the graph Γ form an independent set. Intuitively this means that the node a_1 can be paired only with only one of the nodes in V' at the same time. Thus we can define $|V|$ number of color classes in Γ where the nodes labeled by the same node from G fall into one color class. That is the first color class will consist of nodes $(a_1, b_1), (a_1, b_2), (a_1, b_3), \dots$, the second will consist of nodes $(a_2, b_1), (a_2, b_2), (a_2, b_3), \dots$, the third of nodes $(a_3, b_1), (a_3, b_2), (a_3, b_3), \dots$, and so on.

Similarly, the nodes $(a_1, b_1), (a_2, b_1), (a_3, b_1), \dots, (a_{|V|}, b_1)$ of the graph Γ form an independent set. Thus we can define $|V'|$ number of color classes in Γ where the nodes labeled by the same node from H fall into one color class.

Note that if there is a solution to the spanned subgraph isomorphism problem then the second coloring defines also a “best” coloring as it uses equal to the chromatic number of colors, because $k = |V'| = \chi(\Gamma)$.

We also should point out an interesting phenomenon. In real life if one constructs a product graph for a given problem the nodes of this product graph will be listed in such order that they will be also listed by color classes by one of the above scheme. It is because one lists the nodes of the product graph by a double nested for loop listing the nodes of one graph and the nodes of the other graph. From this it follows that programs using sequential greedy coloring may result in the best possible coloring and will run extremely fast comparing to other programs which use other coloring methods. By our knowledge this phenomenon was not detected previously.

2.2 Second hereditary coloring scheme

We take an independent set $I \subseteq V$ from G , and a clique $K \subseteq V'$ from H . Note that nodes of $\Gamma (a, b) \in W, a \in I, b \in K$ form an independent set. This follows from that all the nodes of I are independent and all the nodes of K are connected thus no (a_1, b_1) and (a_2, b_2) pair can be connected as $\{a_1, a_2\} \in V$ and $\{b_1, b_2\} \notin V'$. Thus if we partition the nodes from G into i independent sets and partition the nodes from H into k cliques then we can define $i \times k$ color classes in Γ where the color classes are formed by pairs of an independent set from G and a clique from H .

Obviously we can partition G into cliques and H into independent sets as well.

The described method can be used with many different partitioning, thus resulting with several different colorings.

2.3 Third hereditary coloring scheme

Similarly to the previous method we partition the set of nodes of G and H graphs. But instead of independent sets in G we shall use s -clique free set I_s , and instead of cliques in H – which is equivalent to an independent set in the \bar{H} complement graph – we shall use an r -clique free set K_r in \bar{H} . Using nodes from these two sets, $a \in I_s, b \in K_r$ the nodes (a, b) in Γ form an $(s + r - 1)$ -clique free set. For further details of s -clique free colorings in clique search see [12].

3. PROPOSED PROGRAM AND PRELIMINARY RESULTS

The proposed k -clique search program is working as follows. In the first part several different colorings are prepared and saved. In the second part we use the standard Carraghan-Pardalos clique search method. In this procedure we always check the remaining nodes against the saved different colorings and use the best possible one.

As the presented work is still in progress the proposed software is yet to be completed. For the sole purpose of demonstration we present here just two problem instances. The take a basic graph B , which is a 25 node graph named `s3myc-3x3.c1q`. From this graph two EVIL graphs were build using 3 and 10 instances as described in our paper

“Benchmark problems for exhaustive exact maximum clique search algorithms” presented in this same conference. The first graph, G_1 has 75 nodes, the second one, G_2 has 250 nodes. The 25 node B graph and 250 node G_2 graph can be downloaded from the site <http://clique.ttk.pte.hu/evil> as the generator program which were used producing the 75 node G_1 graph. We pictured these three graphs on Figure 2 and Figure 1.

We state the problem of spanned subgraph isomorphism that if the 25 node B graph is isomorphic to a subgraph of the 75 node G_1 and the 250 node G_2 graph. For this purpose we produced the two auxiliary product graphs $\Gamma_1 = (W_1, F_1)$ and $\Gamma_2 = (W_2, F_2)$ having $|W_1| = 1875$ and $|W_2| = 6250$ nodes accordingly as described in the Introduction. We pictured the 1875 node Γ_1 product graph on Figure 3. Obviously the base graph B of 25 node is part of the 75 node G_1 and the 250 node G_2 graph, so both auxiliary graphs have maximum cliques of size 25 representing a mapping between the base graph B and the 75 node G_1 and the 250 node G_2 graph.

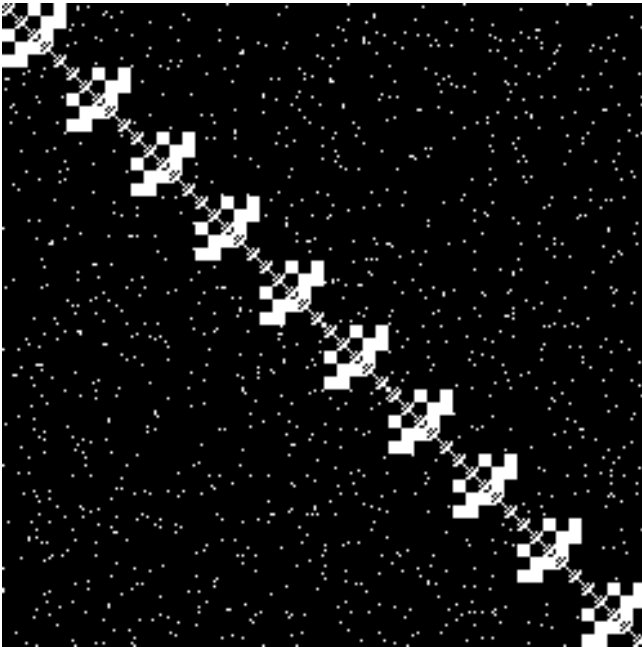


Figure 1: The 250 node G_2 graph.

We used the following 12 clique search algorithms on these product graphs from the following researchers. To eliminate the effect of pre-ordered nodes by color classes noted in Subsection 2.1 we shuffled randomly the nodes of the auxiliary graphs. Note though, that most programs did not perform much better even with the unshuffled variants.

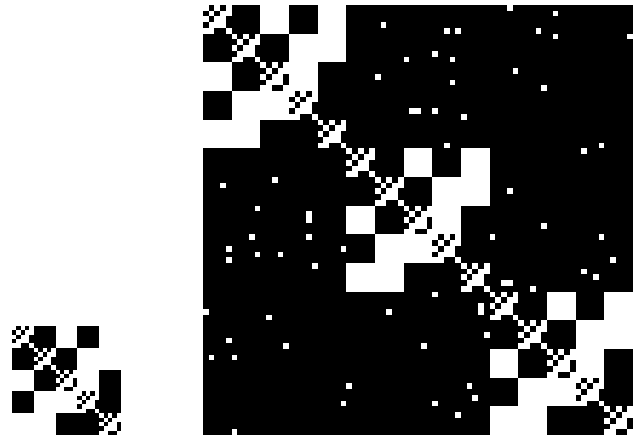


Figure 2: The 25 node B base graph and the 75 node G_1 graph.

1. San Segundo¹ [8], [9], [10], [11] (BBMC, BBMC-R, BBMC-L and BBMC-X).
2. Li² [5], [6] (MaxCLQ 10, MaxCLQ 13-1 and MaxCLQ 13-2).
3. Prosser³ (who implemented Tomita’s algorithm [13]) (MCR)
4. Östergård⁴ [7] (Cliquer),
5. Konc⁵ [3] (mcqd and mcqd-dyn)

There are three ways to use the 2013 version of C.-M. Li program. A switch can be set to either “1” or “2” to select between two built in orderings of the nodes of the graph. In case no value of the switch is specified the program chooses between the “1” and “2” possibilities. During our test we explicitly used the switch “1” and “2” (M-cql 13-1 and M-cql 13-2).

We compared these programs to our own program using the first hereditary coloring scheme. The program can be found on the same site as the EVIL instances and named **antiB**. The brief description of the program is the following.

1. Use the given coloring to color the nodes and save these colors.
2. Set k to be the number of colors of the legal coloring we have been given.
3. Carry out a k -clique search.
4. If a k -clique is found, then it is a maximum clique of the graph. Otherwise reduce the value of k and go to step 3.

¹https://www.biicode.com/pablodev/examples_clique

²<http://home.mis.u-picardie.fr/~cli/EnglishPage.html>

³<http://www.dcs.gla.ac.uk/~pat/maxClique/distribution/>

⁴<http://users.aalto.fi/~pat/cliquer.html>

⁵<http://insilab.org/maxclique/>

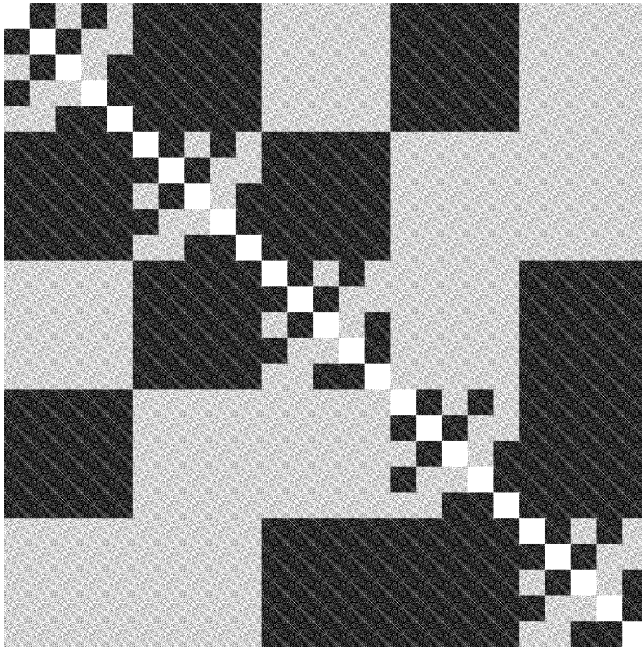


Figure 3: The 1875 node Γ_1 auxiliary graph for testing if B graph is isomorphic to a subgraph of the 75 node G_1 graph.

The k -clique search is based on the Carraghan-Pardalos algorithm, where we utilized original coloring of the nodes. The ordering of the nodes was done by the size of the color classes and the node degrees.

The results of our small test can be seen in Table 1. As one can easily see our simple program is two magnitude faster than the best performing programs on these examples. This clearly indicates the possible potential of the proposed coloring schemes.

Acknowledgment

This research was supported by National Research, Development and Innovation Office – NKFIH Fund No. SNN-117879.

4. REFERENCES

- [1] E. Balas, J. Xue, Weighted and unweighted maximum clique algorithms with upper bounds from fractional coloring, *Algorithmica* **15** (1996), 397–412.
- [2] R. Carraghan, P. M. Pardalos, An exact algorithm for the maximum clique problem, *Operation Research Letters* **9** (1990), 375–382.
- [3] J. Konc and D. Janežič, An improved branch and bound algorithm for the maximum clique problem, *MATCH Communications in Mathematical and Computer Chemistry* **58** (2007), 569–590.
- [4] D. Kumlander, *Some Practical Algorithms to Solve the Maximal Clique problem* PhD. Thesis, Tallin University of Technology, 2005.
- [5] C.-M. Li, Z. Quan, An efficient branch-and-bound algorithm based on MaxSAT for the maximum clique problem, *Proceedings of the Twenty-Fourth AAI*

	$\Gamma_1 = (W_1, F_1)$ $ W_1 = 1875$	$\Gamma_2 = (W_2, F_2)$ $ W_2 = 6250$
Zavalnij: antiB	0.2s	2.5s
San Segundo: BBMC	64s	1768s
BBMC-R	20s	365s
BBMC-L	14s	291s
BBMC-X	21s	387s
Li: MaxCLQ 10	9s	error
MaxCLQ 13-1	136s	>1h
MaxCLQ 13-2	2044s	>1h
Prosser: MCR	35s	>1h
Östergård: Cliquer	134s	1767s
Konc: mcqd	25s	>1h
mcqd-dyn	14s	3065s

Table 1: Running time results for two test problems

Conference on Artificial Intelligence. (AAAI-10), pp. 128–133.

- [6] C.-M. Li, Z. Fang, K. Xu, Combining MaxSAT reasoning and incremental upper bound for the maximum clique problem, *Proceedings of the 2013 IEEE 25th International Conference on Tools with Artificial Intelligence.* (ICTAI2013), pp. 939–946.
- [7] P. R. J. Östergård, A fast algorithm for the maximum clique problem, *Discrete Applied Mathematics* **120** (2002), 197–207.
- [8] P. San Segundo, D. Rodriguez-Losada, A. Jimenez, An exact bit-parallel algorithm for the maximum clique problem, *Computers & Operations Research.* **38** (2011), 571–581.
- [9] P. San Segundo, F. Matia, D. Rodriguez-Losada, M. Hernando, An improved bit parallel exact maximum clique algorithm, *Optimization Letters.* **7** (2013), 467–479.
- [10] P. San Segundo, C. Tapia, Relaxed approximate coloring in exact maximum clique search, *Computers & Operations Research.* **44** (2014), 185–192.
- [11] P. San Segundo, A. Nikolaev, M. Batsyn, Infra-chromatic bound for exact maximum clique search, *Computers & Operations Research.* **64** (2015), 293–303.
- [12] Szabo S. and Zavalnij B. Greedy algorithms for triangle free coloring. *AKCE International Journal of Graphs and Combinatorics.* 9:(2) pp. 169–186. (2012)
- [13] E. Tomita and T. Seki, An efficient branch-and-bound algorithm for finding a maximum clique, *Lecture Notes in Computer Science* **2631** (2003), 278–289.
- [14] D. R. Wood, An algorithm for finding a maximum clique in a graph, *Oper. Res. Lett.* **21** (1997), 211–217.

Testing the Markowitz Portfolio Optimization Method with Filtered Correlation Matrices*

Imre Gera
Institute of Informatics
P.O. Box 652
H-6701 Szeged, Hungary

Balázs Bánhelyi
Institute of Informatics
P.O. Box 652
H-6701 Szeged, Hungary

András London[†]
Institute of Informatics
P.O. Box 652
H-6701 Szeged, Hungary
london@inf.u-szeged.hu

ABSTRACT

In this work we analyze the performance of the Markowitz portfolio optimization method on the Budapest Stock Exchange data set using two different filtering techniques defined for correlation matrices. The results show that the estimated risk is much closer to the realized risk using filtering methods. Bootstrap analysis shows that ratio between the realized return and the estimated risk (Sharpe ratio) is also improved by filtering.

Categories and Subject Descriptors

I.6 [Simulation and Modelling]: Applications
; G.1.6 [Optimization]: *Constrained optimization, Nonlinear programming*

Keywords

Portfolio optimization, Markowitz model, Correlation matrices, Random matrix theory, Hierarchical clustering

1. INTRODUCTION

The portfolio optimization is one of the most important problem in asset management aims at reducing the risk of an investment by diversifying it into independently fluctuating assets [5]. In his seminal work [14], Markowitz formulated the problem through the criteria that given the expected return, the risk - measured by the variability of the return - has to be minimized. The classical model measures the risk as the variance of the asset returns resulting in a quadratic programming problem. Recently, the analysis of the correlation coefficient matrix, that appears through the covariance matrix in the objective function of the model, has become the focus of interest [2, 4, 9, 10, 17, 19]. Many attempts have been made in order to quantify the degree of statistical uncertainty present in the correlation matrix and filter

*This work was partially supported by the National Research, Development and Innovation Office - NKFIH, SNN-117879.

[†]Corresponding author

the part of information which is robust against this uncertainty [2, 7, 9, 10, 11]. The filtered correlation matrices have been successfully used in portfolio optimization in terms of risk reduction [10, 17, 19]. In these studies, it was often assumed that the investor has perfect knowledge on the future returns.

In this work we investigate the portfolio selection problem using different filtering procedures applied to the correlation matrix. We measure the performance of the procedures in terms of both the predicted and realized risk and return, respectively. The future returns are not known at the time of the investment. In Section 2 we briefly describe the Markowitz portfolio optimization problem and two approaches for the correlation matrix filtering (Random Matrix Theory, Clustering). In Section 3 we present our results using standard performance measures on the return and risk, and finally, in Section 4 we draw some conclusions and indicate future work.

2. PORTFOLIO OPTIMIZATION

In Markowitz' formulation, the portfolio problem is a single period model of investment. At the beginning of the period (t_0), an investor allocates the capital among different assets. During the investment period ($[t_0, T]$), the portfolio produces a random rate of return and results a new value of the capital. In the original model of Markowitz, the risk of a single asset is measured by the variance of its returns, while the risk of the portfolio is measured via the covariance matrix of the returns of the assets in the portfolio. In this section we briefly introduce the Markowitz portfolio optimization problem and describe two filtering procedures of the covariance matrix in order to obtain less noisy matrix to decrease the statistical uncertainty it contains.

2.1 Markowitz's model

Given n risky assets, a portfolio composition is determined by the weights p_i ($i = 1, \dots, n$), such that $\sum_i^n p_i = 1$, indicating the fraction of wealth invested in asset i . The expected return and the variance of the portfolio $\mathbf{p} = (p_1, \dots, p_n)$ are

$$r_p = \sum_{i=1}^n p_i r_i = \mathbf{p} \mathbf{r}^T \quad (1)$$

and

$$\sigma_p^2 = \sum_{i=1}^n \sum_{j=1}^n p_i p_j \sigma_{ij} = \mathbf{p} \Sigma \mathbf{p}^T, \quad (2)$$

where r_i is the expected return of asset i , σ_{ij} is the covariance between asset i and j and Σ is the covariance matrix. Vectors are considered as row vectors in this paper.

In the classical Markowitz model [14] the risk is measured by the variance providing a quadratic optimization problem which consists of finding a vector \mathbf{p} , assuming $\sum_{i=1}^n p_i = 1$, that minimizes σ_p^2 for a given "minimal expected return" value of r_p . Now, we assume that short selling is allowed and therefore p_i can be negative. The solution of this problem, found by Markowitz, is

$$\mathbf{p}^* = \lambda \Sigma^{-1} \mathbf{1}^T + \gamma \Sigma^{-1} \mathbf{r}^T, \quad (3)$$

where $\mathbf{1} = (1, \dots, 1)$, while the other parameters are

$$\lambda = (C - r_p B)/D \text{ and } \gamma = (r_p A - B)/D,$$

where

$$A = \mathbf{1} \Sigma^{-1} \mathbf{1}^T, B = \mathbf{1} \Sigma^{-1} \mathbf{r}^T, C = \mathbf{r} \Sigma^{-1} \mathbf{r}^T, D = AC - B^2.$$

Considering the daily price time series of n assets and denoting the closure price of asset i at time t ($t = 1, \dots, T$) by $P_i(t)$, the daily logarithmic return of i is defined as

$$r_{it} = \log \frac{P_i(t)}{P_i(t-1)} = \log P_i(t) - \log P_i(t-1). \quad (4)$$

In case of stationary independent normal returns, which is usually assumed for asset prices, the maximum likelihood estimator is the sample mean of the past observations of r_i , is defined as

$$\hat{r}_i = \frac{1}{T} \sum_{t=1}^T r_{it}. \quad (5)$$

Hence, for the portfolio we define $\hat{\mathbf{r}} = (\hat{r}_1, \dots, \hat{r}_n)$. The covariance σ_{ij} between assets i and j is estimated by

$$\hat{\sigma}_{ij} = \frac{1}{T-1} \sum_{t=1}^T (r_{it} - \hat{r}_i)(r_{jt} - \hat{r}_j) \quad (6)$$

and for the portfolio $\hat{\Sigma} = (\hat{\sigma}_{ij})_{i,j}$. The correlation coefficient between asset i and j is defined as

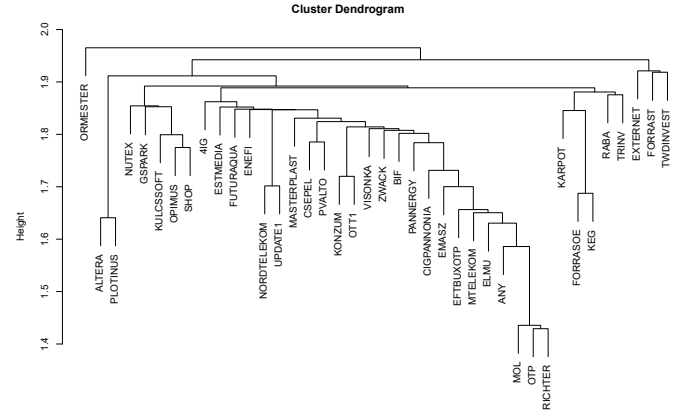
$$\rho_{ij} = \sigma_{ij} / \sqrt{\sigma_{ii} \sigma_{jj}}, \quad (7)$$

where σ_{ii} is often called the volatility of asset i .

2.2 Random matrix theory and correlation matrices

A simple random matrix is a matrix whose elements are random numbers from a given distribution [15]. In context of asset portfolios random matrix theory (RMT) can be useful to investigate the effect of statistical uncertainty in the estimation of the correlation matrix [19]. Given the time series of length T of the returns of n assets and assuming that the returns are independent Gaussian random variables with zero mean and variance σ^2 , then in the limit $n \rightarrow \infty$, $T \rightarrow \infty$ such that $Q = T/n$ is fixed, the distribution $\mathcal{P}_{rm}(\lambda)$ of the eigenvalues of the random correlation matrix (C_{rm}) is given by

$$\mathcal{P}_{rm}(\lambda) = \frac{Q}{2\pi\sigma^2} \frac{\sqrt{(\lambda_{\min} - \lambda)(\lambda_{\max} - \lambda)}}{\lambda}, \quad (8)$$



Minimal Spanning Tree of 40 BUX Assets

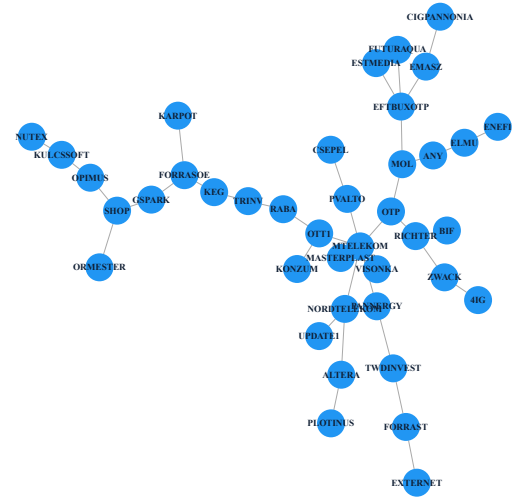


Figure 1: Indexed hierarchical tree - obtained by the single linkage procedure - and the associated MST of the correlation matrix of 40 assets of the Budapest Stock Exchange

where λ_{\min} and λ_{\max} are the minimum and maximum eigenvalues, respectively [18], given in the form

$$\lambda_{\max, \min} = \sigma^2 \left(1 + \frac{1}{Q} \pm 2\sqrt{\frac{1}{Q}} \right). \quad (9)$$

Previous studies have pointed out that the largest eigenvalue of correlation matrices from returns of financial assets is completely inconsistent with Eq. 8 and refers to the common behavior of the stocks in the portfolio [9, 16]. Since Eq. 8 is strictly valid only for $n \rightarrow \infty$, $T \rightarrow \infty$, we constructed random matrices for certain n and T values of the data sets that are used and compare the largest eigenvalues and the spectrum with \mathbf{C} . We found high consistency with Eq. 8. Since $\text{Trace}(\mathbf{C}) = n$ the variance of the part not explained by the largest eigenvalue can be quantified as $\sigma^2 = 1 - \lambda_{\text{largest}}/n$. Using this, we can recalculate λ_{\min} and λ_{\max} in Eq. 9 and construct a filtered diagonal matrix \mathbf{C}_{RMT} , that we get by setting all eigenvalues of \mathbf{C} smaller than λ_{\max} to zero and transform it to the basis of \mathbf{C} with set-

ting the diagonal elements to one (and using singular value decomposition). A possible RMT approach for portfolio optimization, following [17], is to use Σ_{RMT} (that can be easily calculated from C_{RMT}) instead of Σ in the Markowitz model.

2.3 Clustering

The correlation matrix C has $n(n-1)/2 \sim n^2$ distinct elements therefore it contains a huge amount of information even for a small number of assets considered in the portfolio selection problem. As shown by Mantegna and later many others [3, 8, 12, 19, 20], the single linkage clustering approach [6] (closely related to minimal spanning trees (MST), Fig. 1) provides economically meaningful information using only $n-1$ distinct elements of the correlation matrix. To construct the filtered matrix, the correlation matrix C is converted into a distance matrix D , for instance following [12, 13], using $d_{ij} = \sqrt{2(1-\rho_{ij})}$ ultrametric distance¹. The distance matrix D can be seen as a fully connected graph of the assets with edge weights d_{ij} representing similarity between time series of them. Then the filtered correlation matrix C_{MST} is constructed with just $n-1$ distinct correlation coefficients by converting the filtered ultrametric distance matrix back. It was proven that the ultrametric correlation matrix obtained by the single linkage clustering method is always positive definite if all the elements of the obtained ultrametric correlation matrix are positive [1]. This condition has been observed for all correlation matrices we used. Then, for portfolio optimization, we can use the obtained Σ_{MST} instead of Σ in the Markowitz model.

3. RESULTS

3.1 Data set

To compare the performance of the methods we analyze the data set of $n = 40$ stocks traded in the Budapest Stock Exchange (BSE) in the period 1995-2016, using 5145 records of daily returns per stock.

We consider $t = t_0$ as the time when the optimization is performed. Since the covariance matrix has $\sim n^2$ elements while the number of records used in the estimation is nT , the length of the time series need to be $T \gg n$ in order to get small errors on the covariance. On the other hand, for large T the non-stationarity of the time series likely appears. This problem is known as the curse of dimensionality. Because of this, we compute the covariance matrix and expected returns using the $[-T, 0]$ interval, i.e. using $T = 50 \approx n$, $T = 100 > n$ and $T = 500 \gg n$ days preceding $t = 0$. Furthermore, filtering techniques are able to filter the part of the covariance matrix which is less affected by statistical uncertainty. To quantify and compare the different methods are considered, we use the measures described below.

3.2 Performance evaluation

To measure the performance of the portfolios determined by the different models, we use the following quantities for the estimated return and risk at the time of investment and the realized risk and returns after the investment period. For

¹Ultrametric distances are such distances that satisfy the inequality $d_{ij} \leq \max\{d_{ik}, d_{kj}\}$, which is a stronger assumption than the standard triangular inequality.

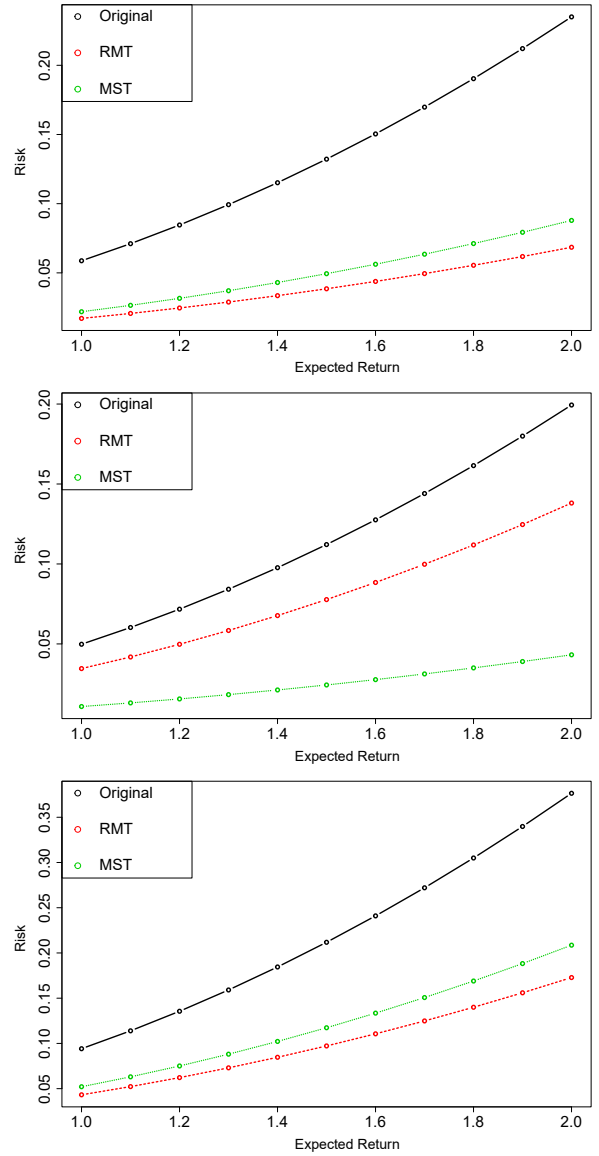


Figure 2: The ratio of the realized risk σ_r^2 and the predicted risk $\hat{\sigma}_p^2$ as the function of expected portfolio return r_p for the different procedures as $T = 50, 100, 500$ (top-down). The data set contains 40 BSE stocks in the period

portfolio p , the ex-ante Sharpe ratio measures the excess return per unit of risk:

$$S_p = \frac{\hat{r}_p - r_f}{\sigma_p}, \quad (10)$$

while the ex-post Sharpe ratio uses the same equation but with the realized return r_p . Here, r_f is the risk-free rate of return. The portfolio risk, due to the estimation of the correlation matrix is calculated as

$$R_p = \frac{|\sigma_r^2 - \hat{\sigma}_p^2|}{\hat{\sigma}_p^2} \quad (11)$$

where $\hat{\sigma}_p^2$ is the predicted risk, while σ_r^2 is the realized risk of the portfolio.

Table 1: Bootstrap experiments using 50 random samples for each value of T in case of 120% expected return

$r_p = 1.2$		Original	RMT	MST
T=50	Return	0.145 (0.330)	0.180 (0.425)	0.186 (0.348)
	S_p	0.009	0.180	0.186
	R_p	16.66	0.99	0.99
T=100	Return	0.319 (0.332)	0.315 (0.541)	0.362 (0.418)
	S_p	0.036	0.315	0.364
	R_p	8.954	0.99	0.99
T=500	Return	-0.185 (0.928)	-0.313 (1.234)	0.264 (0.724)
	S_p	-0.077	-0.313	0.264
	R_p	2.415	0.99	0.99

3.3 Experiments

Fig. 2 shows the ratio of the realized risk σ_r^2 and the predicted risk $\hat{\sigma}_p^2$ as the function of the estimated return r_p obtained by the different procedures. For each T , the investment time t_0 and the set of stocks were the same. The ratio is significantly smaller in case of the portfolios that obtained by using filtering. Interestingly, for $T = 100$ the MST method gave better results than the RMT.

To check the robustness of the methods, we performed a bootstrap experiment as follows. We considered 50 random initial times to solve the optimization problem using the time series on the intervals $[-T, t_0]$ ($T = 50, 100, 500$). For each portfolio, we computed the predicted risk using Eq. 2 for expected returns $r_p = 1, 1.1, \dots, 2$ (0 – 100% gain). We further constrained p_i to the interval $[-1, 1]$ and used the Lagrange multiplier method for the optimization. In all cases, the portfolios with realized returns in the top and bottom 10% were neglected. We computed the realized risk using the calculated stock weights at t_0 and the realized covariance matrix on $[t_0, T]$. We also computed the realized returns by comparing the value of the portfolio at t_0 and T . The average S_p , R_p values and returns with standard deviations for $r_p = 1.2$ are shown in Tab. 1. It can be seen, the R_p values are significantly smaller in case of the RMT and MST than in case of the original method for each T confirming the reliability of the filtering methods. The post-ante Sharpe ratio, however it is much smaller than 1 in every case, also shows that the RMT and MST methods outperforms the original method. We note, interestingly, that the highest average return was obtained for $T = 100$ (and not for $T = 500$) using the BSE data set.

4. CONCLUSIONS

In this study, we performed portfolio optimization using filtered correlation matrices obtained by two different procedures, namely a random matrix theory approach and the single linkage clustering. A large set of experiments have shown that using filtered covariance matrices the original Markowitz solution is outperformed in terms of standard portfolio performance measures.

In the future, it would be interesting to analyze portfolio optimization using various estimators of expected returns together with different filtering procedures and check the methods using various stock exchange data sets and also varying the number of stocks considered.

5. REFERENCES

- [1] M. R. Anderberg. Cluster analysis for applications. monographs and textbooks on probability and mathematical statistics, 1973.
- [2] T. Conlon, H. J. Ruskin, and M. Crane. Random matrix theory and fund of funds portfolio optimisation. *Physica A: Statistical Mechanics and its Applications*, 382(2):565–576, 2007.
- [3] T. Di Matteo, T. Aste, and R. N. Mantegna. An interest rates cluster analysis. *Physica A: Statistical Mechanics and its Applications*, 339(1):181–188, 2004.
- [4] M. El Alaoui. Random matrix theory and portfolio optimization in moroccan stock exchange. *Physica A: Statistical Mechanics and its Applications*, 433:92–99, 2015.
- [5] E. J. Elton, M. J. Gruber, S. J. Brown, and W. N. Goetzmann. *Modern portfolio theory and investment analysis*. John Wiley & Sons, 2009.
- [6] J. C. Gower and G. Ross. Minimum spanning trees and single linkage cluster analysis. *Applied statistics*, pages 54–64, 1969.
- [7] T. Guhr and B. Kälber. A new method to estimate the noise in financial correlation matrices. *Journal of Physics A: Mathematical and General*, 36(12):3009, 2003.
- [8] L. Kullmann, J. Kertész, and K. Kaski. Time-dependent cross-correlations between different stock returns: A directed network of influence. *Physical Review E*, 66(2):026125, 2002.
- [9] L. Laloux, P. Cizeau, J.-P. Bouchaud, and M. Potters. Noise dressing of financial correlation matrices. *Physical review letters*, 83(7):1467, 1999.
- [10] L. Laloux, P. Cizeau, M. Potters, and J.-P. Bouchaud. Random matrix theory and financial correlations. *International Journal of Theoretical and Applied Finance*, 3(03):391–397, 2000.
- [11] Y. Malevergne and D. Sornette. Collective origin of the coexistence of apparent random matrix theory noise and factors in large sample correlation matrices. *Physica A: Statistical Mechanics and its Applications*, 331(3):660–668, 2004.
- [12] R. N. Mantegna. Hierarchical structure in financial markets. *The European Physical Journal B-Condensed Matter and Complex Systems*, 11(1):193–197, 1999.
- [13] R. N. Mantegna and H. E. Stanley. *Introduction to econophysics: correlations and complexity in finance*. Cambridge university press, 1999.
- [14] H. Markowitz. Portfolio selection: Efficient diversification of investments. cowles foundation monograph no. 16, 1959.
- [15] M. L. Mehta. *Random matrices*, volume 142. Academic press, 2004.
- [16] V. Plerou, P. Gopikrishnan, B. Rosenow, L. A. N. Amaral, and H. E. Stanley. Universal and nonuniversal properties of cross correlations in financial time series. *Physical Review Letters*, 83(7):1471, 1999.
- [17] B. Rosenow, V. Plerou, P. Gopikrishnan, and H. E. Stanley. Portfolio optimization and the random magnet problem. *EPL (Europhysics Letters)*, 59(4):500, 2002.
- [18] A. M. Sengupta and P. P. Mitra. Distributions of singular values for some random matrices. *Physical Review E*, 60(3):3389, 1999.
- [19] V. Tola, F. Lillo, M. Gallegati, and R. N. Mantegna. Cluster analysis for portfolio optimization. *Journal of Economic Dynamics and Control*, 32(1):235–258, 2008.
- [20] M. Tumminello, T. Di Matteo, T. Aste, and R. Mantegna. Correlation based networks of equity returns sampled at different time horizons. *The European Physical Journal B*, 55(2):209–217, 2007.

Tight Online Bin Packing Algorithm with Buffer and Parametric Item Sizes *

József Békési

Department of Applied Informatics
Gyula Juhász Faculty of Education
University of Szeged
H-6701 Szeged, POB 396, Hungary
bekesi@jgypk.szte.hu

Gábor Galambos

Department of Applied Informatics
Gyula Juhász Faculty of Education
University of Szeged
H-6701 Szeged, POB 396, Hungary
galambos@jgypk.szte.hu

ABSTRACT

In this paper we investigate the online bin packing problem with constant buffer size, where the item sizes are in the interval $(0, \frac{1}{r}]$, where $r \geq 2$ is an integer. The problem was originally given by Zheng et al [13]. They gave a lower bound and an algorithm, which were later improved by Zhang et al [12]. We close the gap on the competitive ratio and give a First Fit based optimal solution for the parametric version for arbitrary r .

General Terms

Theory

Keywords

asymptotic competitive ratio, next fit, first fit, lower bound

1. INTRODUCTION

One-dimensional bin packing is a well-known problem of combinatorial optimization. It can be defined as follows: we are given a list $L = \{x_1, x_2, \dots, x_n\}$ of real numbers (called items) from the interval $(0, 1]$, and we want to pack each item into a unique capacity bin. The aim is to use the minimal number of bins. It is known that finding the optimal assignment is NP-hard [6]. Consequently, it is interesting to find polynomial time approximation algorithms with good approximate behaviour (see surveys [3] [4]).

In practical situations it often happens, that the input is not known completely by the algorithm. This is the reason that researchers focused on studying *online problems*. In this case the items come one by one and the algorithm should assign the next item to a bin immediately after its arrival. Later the items can no be repacked. The algorithms defined for online problems are called *on-line algorithms*. Of course

*This research was supported by the Austrian-Hungarian Action Foundation (Project number: 91öu2).

the approximation of the optimal algorithm by an online algorithm is harder than the one by an *offline algorithm*, which knows the whole input in advance.

To measure the efficiency of an online algorithm, we have several possibilities. In case of bin packing one of the classical methods is the worst case analysis. Traditionally the so called *asymptotic competitive ratio* R_∞ is used to measure the efficiency of an online algorithm in the bin packing literature. Its definition for algorithm A is the following:

$$R_\infty(A) := \limsup_{l \rightarrow \infty} \left\{ \max_L \left\{ \frac{A(L)}{l} \mid OPT(L) = l \right\} \right\}. \quad (1)$$

The online algorithm with the best known asymptotic competitive ratio 1.5815 is due to Heidrich and van Stee [9], while the best lower bound is 1.54037... given by Balogh et al. in [2]. One of the simplest online algorithms is *Next Fit*. It uses only one open bin and puts the next element into it, if it is possible. Otherwise it closes the bin and opens a new one. It is well-known that $R_{NF}^\infty = 2$. A very famous online algorithm is First Fit (FF). It keeps open all bins used during the algorithm, and packs the next item into the first bin where it fits. Ullmann [11] proved first that the asymptotic competitive ratio of FF is 1.7.

It is clear that the online restriction results in a bad competitive ratio. To avoid this several relaxations of the online property and space limitations appear in the literature. Using lookahead buffers, repacking or preordering of the input are the most common methods. In general the algorithms that use such techniques are called *semi-online algorithms*. For example arriving of the input in decreasing order improves the asymptotic competitive ratio of NF to 1.69103... [1]. A similar improvement can be achieved by repacking, which was proved by Galambos and Woeginger in [5]. Garey et al. [7] and Johnson et al. [8] proved that FF works much better if the elements of the input are sorted in decreasing order. In this case $R_\infty(FFD) = \frac{11}{9}$. These techniques can be used in many practical applications.

Based on such an application Zheng et al [13] defined a variant of the original bin packing problem. In this version a list of items with sizes bounded by a small interval arrive and they can be temporarily stored in a capacitated buffer before packing them into bins. Zheng et al gave a lower bound of $\frac{4}{3}$ and defined an algorithm with competitive ratio of $\frac{13}{9}$

[13]. Later Zhang et al improved this to 1.423 and 1.4243 respectively [12]. So a small gap of 0.0013 remained and they analysed only the case when the upper bound of the sizes of the item is $\frac{1}{2}$. In this paper we investigate the so-called *r-parametric case* where the item sizes are in the interval $(0, \frac{1}{r}]$. First, we give an improved lower bound for any online algorithm with constant buffer size S . We also give an algorithm, which based on the method given by Galambos and Woeginger in [5]. We prove that the competitive ratio of our algorithm is equal to the value of the new lower bound. So, we close the gap for arbitrary values of r , where $r = 2, 3, \dots$

2. PRELIMINARIES

We will use a sequence which was first introduced by Sylvester in [10] (1880) for the case $r = 1$, therefore, we refer to this sequence as *generalized Sylvester sequence*.

For integers $k > 1$ and $r \geq 1$, the generalized Sylvester sequence m_1^r, \dots, m_k^r can be given by the following recursion.

$$m_1^r = r + 1, \quad m_2^r = r + 2, \\ m_j^r = m_{j-1}^r(m_{j-1}^r - 1) + 1, \text{ for } j = 3, \dots, k.$$

m_j^r	$r = 1$	$r = 2$	$r = 3$	$r = 4$	$r = 5$
$j = 1$	2	3	4	5	6
$j = 2$	3	4	5	6	7
$j = 3$	7	13	21	31	43
$j = 4$	43	157	421	931	1807
$j = 5$	1807	24493	176821	865831	3263443

Table 1.1. The first few elements of the generalized Sylvester sequences if $k \leq 5$.

These sequences have the following properties.

$$\sum_{i=j}^k \frac{1}{m_i^r} = \frac{1}{m_j^r - 1} - \frac{1}{m_{k+1}^r - 1}, \quad \text{if } j \geq 2,$$

and

$$\frac{r}{m_1^r} + \sum_{i=2}^k \frac{1}{m_i^r} = 1 - \frac{1}{m_{k+1}^r - 1} \quad \text{if } r \geq 2.$$

In the above equations the sizes of the lists were derived from the generalized Sylvester sequences. For example, if $r = 1$, then the sizes are $\frac{1}{2} + \varepsilon, \frac{1}{3} + \varepsilon, \frac{1}{7} + \varepsilon, \frac{1}{43} + \varepsilon, \dots$. We will use these sequences to give lower bounds. Similarly, we will allude to the following constants.

$$h_\infty(r) = 1 + \sum_{i=2}^{\infty} \frac{1}{m_i^r - 1}.$$

The first few values of $h_\infty(r)$: $h_\infty(1) \approx 1.69103$, $h_\infty(2) \approx 1.42312$, $h_\infty(3) \approx 1.30238$.

Generally, to avoid the pilling of indexes we will denote m_j^r by m_j .

3. LOWER BOUND CONSTRUCTION

We will construct the following instance. Let $n > 0$ be a large integer and let $k > 4$ be an integer. Then we will consider the concatenated list $L = (L_1, L_2, \dots, L_k)$, where

- L_1 contains $n(m_k - 1)(m_1 - 1)$ items with size $\frac{1}{m_1} + \varepsilon$.
- L_i contains $n(m_k - 1)$ items with size $\frac{1}{m_i} + \varepsilon$, for $2 \leq i \leq k - 1$,
- L_k contains $n(m_k - 1)$ items with size $\frac{1}{m_k - 1} - k\varepsilon$,

where ε is arbitrary small, ie. $\frac{1}{m_k - 1} - (m_1 + k - 3)\varepsilon > \frac{1}{m_k}$.

Using the above construction we can prove the following theorem for the given problem.

THEOREM 3.1. *Let us consider the r-parametric case. If there is a buffer with buffer size $|B| = S$, then for any online algorithm $R_\infty(A) \geq h_\infty(r)$.*

Since for $r = 2$ $h_\infty(r) = 1.423117\dots$, for the problem considered in [13] and [12] this lower bound gives an improvement.

4. THE WEIGHTING FUNCTION

Investigating online bounded-spaced algorithms in [5] a weighting function was defined to use during the analysis of an algorithm. Generalizing the idea we define the following weighting function.

$$W(x) = \begin{cases} x + \frac{1}{m_i(m_i - 1)}, & \text{if } \frac{1}{m_i} < x \leq \frac{1}{m_i - 1} \\ \frac{m_i + 1}{m_i}x, & \text{if } \frac{1}{m_{i+1} - 1} < x \leq \frac{1}{m_i}. \end{cases}$$

The weight of a bin is defined as the weight of all elements in it, and generally, the weight of a set is the weight of all items in the set. It is easy to see that the following statements are true.

FACT 4.1.

- $W(x)$ is nondecreasing in $(0, 1]$.
- For $i \geq 1$, $\frac{W(x)}{x} \leq \frac{m_i + 1}{m_i}$ if $x \leq \frac{1}{m_i}$,
- For $i \geq 1$, $\frac{W(x)}{x} \geq \frac{m_i + 1}{m_i}$ if $x \geq \frac{1}{m_{i+1} - 1}$.

First we prove the following theorem.

THEOREM 4.2. *Let us consider the r-parametric problem. Then any packing of a list L the weight of any bin is at most $h_\infty(r)$.*

As a consequence of the above theorem, the following corollary is true.

COROLLARY 4.3. *For any list L , $W(L) \leq h_\infty(r)OPT(L)$.*

5. THE ALGORITHM

Our algorithm – called *First Fit Decreasing with Buffer-length 3*, FFD3B – is as follows.

- (1) Fill up the buffer with the subsequent elements of the list until the next item fits into the buffer.
- (2) Order the items in the buffer in nonincreasing order, and put the items in three virtual bins each of them with capacity 1 using the FFD rule.
- (3) Check the sum of the weights in the virtual bins. Find a set of items in the virtual bins with weight greater or equal than one, open a new empty bin, put the items from the virtual bins into this new-opened bin, and close the bin.
- (4) If there is an unplaced item then goto (1),
- (5) Empty the contents of the virtual bins into new-opened bins. Close the bins, and quit.

THEOREM 5.1. *If we pack the items of any list by the algorithm FFD3B then either in the step (3) we have a good subset of items (a subset of weight greater or equal than one) or we have enough place in the buffer to accept a new item from the list.*

Because of the above theorem, as a consequence we get the following theorem.

THEOREM 5.2. *For any list L , $W(L) \geq \text{FFD3B}(L) - 3$.*

Therefore we get the following corollary.

COROLLARY 5.3. *For the r -parametric case $R_\infty(\text{FFD3B}) = h_\infty(r)$.*

6. CONCLUSION

In this paper we gave an online bin packing algorithm for the special problem when the sizes of the items are in the interval $(0, \frac{1}{r}]$ for arbitrary values of $r, r = 2, 3, \dots$ and when we can use a capacitated lookahead buffer to temporarily store some elements. We proved that the asymptotic competitive ratio of our algorithm is tight.

7. REFERENCES

- [1] B.S. Baker, E.G. Coffman, A tight asymptotic bound for next-fit-decreasing bin-packing, *SIAM J. Algebraic Discrete Methods*,2,147–152,(1981).
- [2] J. Balogh, J. Békési, G. Galambos, New lower bounds for certain classes of bin packing algorithms, *Theoretical Computer Science*,440–441, 1–13,(2012).
- [3] E.G. Coffman, M. Garey, D. Johnson, Approximation algorithms for bin packing: A survey, *In: Approximation algorithms for NP-hard problems, Edited by Dorit S. Hochbaum, PWS Publishing Co.,46–93,(1996).*
- [4] E.G. Coffman, G. Galambos, S. Martello, D. Vigo, Bin packing approximation algorithms: combinatorial analysis. *In: DZ. Du, P. Pardalos (eds) Handbook of Combinatorial Optimization.* Kluwer,Dordrecht,151–208,(1999).
- [5] G. Galambos, G. J. Woeginger, Repacking helps in bounded space online bin packing, *Computing*, 49,329–338,(1993).
- [6] M. R. Garey, D. S. Johnson, Computer and Intractability: A Guide to the theory of NP-Completeness, New York, Freeman, (1979).
- [7] M. R. Garey, R. L. Graham, J. D. Ullman, Worst-case analysis of memory allocation algorithms. *Proc. 4th Symp. Theory of Computing (STOC)*, ACM,143–150,(1973).
- [8] D. S. Johnson, A. Demers, J. D. Ullman, M. R. Garey, R. L. Graham, Worst-case performance bounds for simple one-dimensional packing algorithms. *SIAM J. Comput.*,3,256–278,(1974).
- [9] S. Heydrich, R. van Stee, Beating the Harmonic lower bound for online bin packing, *in Proceedings of the 43rd International Colloquium on Automata, Languages and Programming (ICALP)*,(2016).
- [10] J. Sylvester, On a Point in the Theory of Vulgar Fractions, *American Journal of Mathematics*,3,332–335,(1880).
- [11] J. D. Ullman, The performance of a memory allocation algorithm, Technical Report 100, Princeton Univ., Princeton, NJ,(1971).
- [12] M. Zhang, X.Han, Y. Lan, H-F. Ting, Online bin packing problem with buffer and bounded size revisited. *Journal of Combinatorial Optimization*,DOI 10.1007/s10878-015-9976-5,(2015).
- [13] F. Zheng, L. Huo, E. Zhang, NF-based algorithms for online bin packing with buffer and bounded item size. *Journal of Combinatorial Optimization*, 30(2),360–369,(2015).

A Branch-and-Cut Algorithm for the Multi-Depot Rural Postman Problem

Elena Fernández
Universitat Politècnica de
Catalunya-BcnTech
Barcelona, Spain
e.fernandez@upc.edu

Gilbert Laporte
HEC Montréal
Montréal, Canada
gilbert.laporte@cirrelt.ca

Jessica
Rodríguez-Pereira
Universitat Politècnica de
Catalunya-BcnTech
Barcelona, Spain
jessica.rodriguez@upc.edu

ABSTRACT

This paper studies the Multi-Depot Rural Postman Problem on an undirected graph. This problem is the extension of the well-known Undirected Rural Postman Problem to the case where there are several depots instead of just one. A linear integer programming formulation that only uses binary variables is proposed, which includes three families of constraints of exponential size. An exact branch-and-cut algorithm is presented, where violated constraints of both types are separated in polynomial time. Despite the difficulty of the problem, the numerical results from a series of computational experiments with various types of instances illustrate a quite good behavior of the algorithm.

Categories and Subject Descriptors

[Theory of computation]: Mathematical optimization Branch-and-bound Algorithm design techniques

General Terms

Theory

Keywords

Arc routing; multi-depot rural postman problem; worst-case analysis; polyhedral analysis; branch-and-cut

1. INTRODUCTION

In this work we develop a branch-and-cut algorithm for the Multi-Depot Rural Postman Problem (MDRPP), which extends the classical Rural Postman Problem (RPP) [13] where there is only one depot. Similarly to the RPP, routes must be designed to serve a given set of required edges. In contrast, in the MDRPP the depot from which each required edge is served is not known in advance. The MDRPP combines two types of decisions: the allocation of required edges to depots and the planning of routes. The objective is to determine a minimum cost set of routes, each starting and ending at the

same depot, and such that each required edge is traversed at least once.

The motivation for studying the MDRPP comes not only from its theoretical interest but also from its real-life applications. Similarly to other arc routing problems, such applications arise in a wide variety of practical cases, namely garbage collection, road maintenance, mail delivery, snow plowing or pipelines inspection, to name just a few. In large-scale instances, there is usually more than one depot from which service demand can be satisfied. Such depots may be vehicle stations, dump sites, replenishment points or relay boxes. A way of handling such problems is to first define a smaller operating area for each depot, by using a districting procedure in which each district contains a single depot, and then solving the RPP associated with each district. This solution strategy is of course suboptimal.

The literature on Multi-Depot Arc Routing Problems (MDARP) is scarce. To the best of our knowledge, [4, 3] are the only existing exact algorithms for the MDRPP. Both use *natural* decision variables which explicitly indicate the depot with which each traversed edge or arc is associated. Other than this, previous work on MDARPs focused on multi-depot capacitated arc routing problems (MDCARPs). Although there are some theoretical works and exact algorithms [14, 9] most of the existing literature on MDCARPs focuses on heuristic solution algorithms (see, [1, 12, 11, 8, 7, 6]).

Multi-depot routing problems are also related to districting-arc routing problems where a set of clusters or districts that suitably partition the required edge set is sought. The design of good districts at a strategic level, where demand points or edges are allocated to depots, allows finding efficient routes at each district at an operational in a later phase. There exists a rich districting literature in relation to arc routing (see, for instance, [12, 11, 10]). Two recent works on districting for arc routing are [2, 5].

A natural option when dealing with routing problems with multiple depots is to associate routes with depots and then to define the routes for each depot. From a modeling point of view, this can be easily done by using decision variables that explicitly indicate the arcs/edges traversed by the routes of each depot. Such an alternative offers two main advantages. On the one hand, in absence of capacity or other type of con-

straints, the feasibility of a route corresponding to a fixed depot is guaranteed by connectivity plus parity constraints. On the other hand, routes can be easily reproduced once the values of the decision variables are known. The obvious disadvantage of this option is that the number of variables increases with the number of depots, so the success of exact solution methods for large size instances becomes a challenge. The two previous MDRPP works referenced above [4, 3] use this type of decision variables. In [4] which deals with exactly the same undirected MDRPP that we study in this paper, instances with up to 100 vertices and 4 depots were solved to optimality. In [3], which addresses a directed MDARP dealing with carriers collaboration, instances with up to 50 vertices and 2 depots were optimally solved.

2. THE MULTI-DEPOT RURAL POSTMAN PROBLEM

In this work we carry out a worst-case analysis to study the potential savings that can be obtained, with respect to the RPP and some variations, when multiple depots are considered. We denote by $z^*(MDRPP)$ the optimal value of a MDRPP instance and by $z^*(RPP)$ the optimal value of the same instance with only one depot. It is possible to prove that savings can be obtained in both directions, since the best solutions are not necessarily obtained using more than one depot. A summary of the results that we prove are:

Theorem 2.1. *There exists no finite upper bound for the ratio $z^*(RPP)/z^*(MDRPP)$.*

Theorem 2.2. *$z^*(RPP)/z^*(MDRPP) \geq 1/2$, and the bound is asymptotically tight.*

Furthermore, we present a new integer linear formulation for the MDRPP with binary decision variables, which are solely associated with edges, but not with depots. In particular, two sets of binary variables are used, associated with the first and second traversals of edges, respectively. For each $e \in E$, let x_e be a binary variable indicating whether or not edge e is traversed by some route. We denote by $E^y \subset E$ the set of edges that can be traversed twice in an optimal solution. For each $e \in E^y$, let y_e be a binary variable that equal to one if and only if edge e is traversed twice. Indeed, the reduction on the number of decision variables used in our formulation comes at the expenses of additional difficulties. Now, connectivity and parity constraints are not enough to guarantee well-defined routes. To overcome this difficulty we propose a new set of constraints that guarantee that each route terminates at the same depot where it has started, which can be separated in polynomial time.

Likewise, we study the polyhedral properties of the formulation. In this sense, we prove that the convex hull of the polyhedron associated with feasible solutions is full-dimensional under a certain conditions. Furthermore, under mild conditions the trivial inequalities and some families of constraints are facet defining.

We finally propose an exact branch-and-cut algorithm with exact separation for all the families of inequalities of expo-

ponential size. The algorithm has been implemented and its computational effectiveness tested on a series of computational experiments with a set of benchmark instances. The numerical results assess the good performance of the solution algorithm, as it is capable of solving to optimality, in reasonable computing times, instances with up to 700 vertices and four depots.

3. CONCLUSIONS

We have studied the Multi-Depot Rural Postman Problem (MDRPP), which is the extension of the RPP to the case of several depots. A worst-case analysis of the MDRPP with respect to the RPP indicates that the potential savings can be arbitrarily large, but also that the RPP may produce better solutions. Worst case analysis has been carried out and binary linear formulation for the MDRPP has been presented. The formulation includes a new family of inequalities that ensure that routes start and end at the same depot. The properties of the polyhedron associated with the formulation have studied. Furthermore, we have developed a branch-and-cut algorithm for the MDRPP based on the proposed formulation. The algorithm is capable of solving to optimality within reasonable computing times instances with up to 700 vertices and four depots.

4. ACKNOWLEDGMENTS

This research has been partially supported by the Spanish Ministry of Economy and Competitiveness and EDRF funds through grants EEBB-I-16-10670, BES-2013-063633, and MTM2012-36163-C06-05 and MTM2015-63779-R (MINECO/FEDER), and by the Canadian Natural Sciences and Engineering Research Council under grant 2015-06189. This support is gratefully acknowledged.

5. REFERENCES

- [1] A. Amberg, W. Domschke, and S. Voß. Multiple center capacitated arc routing problems: A tabu search algorithm using capacitated trees. *European Journal of Operational Research*, 124(2):360–376, 2000.
- [2] A. Butsch, J. Kalcsics, and G. Laporte. Districting for arc routing. *INFORMS Journal on Computing*, 26(4):809–824, 2014.
- [3] E. Fernández, D. Fontana, and M. Speranza. On the collaboration uncapacitated arc routing problem. *Computers & Operations Research*, 67:120–131, 2016.
- [4] E. Fernández and J. Rodríguez-Pereira. The multi-depot rural postman problem. *Submitted for publication*, 2016.
- [5] G. García Ayala, J. González-Velarde, R. Ríos-Mercado, and E. Fernández. A novel model for arc territory design: Promoting Eulerian districts. *International Transactions in Operation al Research*, 23:433–458, 2015.
- [6] B. Golden and R. Wong. Capacitated arc routing problems. *Networks*, 11(3):305–315, 1981.
- [7] H. Hu, T. Liu, N. Zhao, Y. Zhou, and D. Min. A hybrid genetic algorithm with perturbation for the multi-depot capacitated arc routing problem. *Journal of Applied Sciences*, 13(16):32–39, 2013.
- [8] A. Kansou and A. Yassine. A two ant colony approaches for the multi-depot capacitated arc routing problem. In *International Conference on Computers &*

Industrial Engineering, 2009. CIE 2009., pages 1040–1045. IEEE, 2009.

- [9] D. Krushinsky and T. Van Woensel. Lower and upper bounds for location-arc routing problems with vehicle capacity constraints. *European Journal of Operational Research*, 244(1):100–109, 2015.
- [10] L. Muyldermans. Routing, districting and location for arc traversal problems. Technical report, PhD dissertation, Catholic University of Leuven, Leuven, Belgium, 2003.
- [11] L. Muyldermans, D. Cattrysse, and D. Van Oudheusden. Districting for arc-routing applications. *Journal of the Operational Research Society*, 54:1209–1221, 2003.
- [12] L. Muyldermans, D. Cattrysse, D. Van Oudheusden, and T. Lotan. Districting for salt spreading operations. *European Journal of Operational Research*, 139(3):521–532, 2002.
- [13] C. S. Orloff. A fundamental problem in vehicle routing. *Networks*, 4(1):35–64, 1974.
- [14] S. Wøhlk. Contributions to arc routing. Technical report, PhD dissertation, University of Southern Denmark, Odense, Denmark, 2004.

Allocation and Pricing on a Network in Presence of Negative Externalities

[Extended Abstract]

Saša Pekeč
Duke University
pekec@duke.edu

1. INTRODUCTION

We discuss network optimization problems that arise naturally in the context of optimally allocating and pricing indivisible homogeneous items to unit-demand agents in a network, with the caveat that the agents face negative allocative externalities. Specifically, agent's value for (not) getting an item depends on whether any of its rivals did (not) get an item. The rivalry is represented by a network with nodes representing agents and arcs representing whether an agent considers another agent its rival.

An agent i could have four different values depending on the allocation structure: w_i is the value if agent i gets an item but no rival gets it; v_i is the value if agent i gets an item and at least one of its rivals also gets it; without loss of generality, we normalize to zero agent i 's value for no item allocated to i nor any of its rivals; finally, agent i experiences a loss $-\alpha_i$ if i does not get an item but one of the rival's does get it. (Note that this valuation structure could be generalized beyond binary, so that agent i 's value depends on the number of rivals who also got an item.) With normalization for one value, agents' valuation function is three-dimensional. Such valuation structure generalizes those studied in [10] and [2] where results analogous to those presented below were initially established.

The settings that can be represented with such allocation structure are prevalent in business. For example, representation $w_i \geq v_i \geq 0 = -\alpha_i$ describes a setting in which agents put a premium $w_i - v_i$ on an exclusive allocation, and lose nothing ($0 = -\alpha_i$) if a rival gets the item. Exclusivity is considered valuable in a variety of settings. For example, the right to sell a product or offer a service exclusively is more valuable than having to compete for a market share with rivals who might secure the same right. The scope of exclusivity rights might be limited to a geographic area or a market segment, as is common with franchising and with exclusive sales, service and distribution agreements. For another example, representation $w_i = v_i \geq 0 \geq -\alpha_i$ describes

a fierce competitive setting in which rival obtaining an item imposes a loss for agent i who did not get an item. Industries that involve patent and intellectual property protection or a regulatory approval are examples in which an agent who obtains an item imposes negative externalities on all of its rivals. For example, when pharmaceutical companies race to develop a drug, the company who is first to obtain the patent and/or regulatory approval essentially eliminates competition from the market for that drug, thereby turning into losses all of the rival's R&D investment for the same purpose drug development.

We focus on the setting in which agents' values are private and the monopolist seller's objective is to maximize expected revenue. In other words, we are looking for the optimal deterministic mechanism in presence of multi-dimensional independent private values. The mechanism design problem with externalities has been studied in computer science literature, mostly motivated by mechanisms for allocation and pricing of digital ads. (The prevailing approach to design an auction procedure is to design a polynomial time algorithm for solving given mechanism design problem and then interpret it as an auction.) Allocative externalities in digital ad context arise naturally: an advertiser could value an exclusive ad placement; similarly an advertiser who lost out on ad placement might prefer competitor ad not being shown either. Some notable works that focus on related problems include [6],[11],[8],[3],[12],[9],[7],[4],[5].

2. EXCLUSIVITY MODEL

A monopolist seller has K identical items that can be allocated among $N = \{1, 2, \dots, n\}$ unit-demand agents. Relationships among agents are defined by a network (N, E) where E is the 0-1 adjacency matrix: $e_{ij} = 1$ if and only if agent i considers agent j , $j \neq i$, to be its rival (e.g., i considers j a competitor or i and j are geographical neighbors or directly connected in a social network). Let $S(i) \subseteq N \setminus \{i\}$ denote the set of agent i 's neighbors, i.e., the set of all other agents that i considers to be related to her: $S(i) = \{j \in N : e_{ij} = 1\}$.

Agent i 's type is represented by a vector $\mathbf{v}_i = (w_i, v_i)$, where w_i is agent i 's (exclusivity) valuation for the item if none of her neighbors $j \in S(i)$ gets an item, and where v_i is agent i 's (non-exclusivity) valuation for the item if there is a neighbor $j \in S(i)$ who also obtains the item. We assume

$$w_i \geq v_i \geq 0,$$

where, without the loss of generality, we can normalize by setting agent i 's value for not getting an item to zero. Note that $w_i - v_i$ can be thought of as the exclusivity premium.

We consider the setting in which \mathbf{v}_i is privately known, while network (N, E) and the number of available items K are publicly known. Agent i 's private information $\mathbf{v}_i = (w_i, v_i)$ is drawn independently (across agents) from a joint cumulative distribution function F_i with support $V = [\underline{w}, \bar{w}] \times [\underline{v}, \bar{v}]$. The corresponding density function is denoted by f_i . Seller's valuation vector is $(0, 0)$.

By the Revelation Principle [14], we consider direct mechanisms that allocate items based on agents' reports. Reports from all agents are denoted by $\widehat{\mathbf{v}} = (\widehat{\mathbf{v}}_i, \widehat{\mathbf{v}}_{-i}) \in V^2$, with commonly (ab)used convention that subscript $-i$ denotes information corresponding to all agents except agent i (e.g., $\widehat{\mathbf{v}}_{-i}$ is a shorthand notation for $\{\widehat{\mathbf{v}}_j : j \neq i\}$). A direct mechanism specifies the allocation: $p_i : V^2 \rightarrow \{0, 1\}$ is agent i 's probability to get an item, and payments: $m_i : V^2 \rightarrow \mathbf{R}$ is the payment from agent i to the seller, for each $\widehat{\mathbf{v}} \in V^2$. If agent i does not participate, she does not get any item.

Agent i 's ex post utility when she reports her type as $\widehat{\mathbf{v}}_i$, while her true type is \mathbf{v}_i , and when other agents report \mathbf{v}_{-i} , is

$$\begin{aligned} U_i(\widehat{\mathbf{v}}_i, \mathbf{v}_i, \mathbf{v}_{-i}) &= w_i p_i(\widehat{\mathbf{v}}_i, \mathbf{v}_{-i}) \prod_{j \in S(i)} (1 - p_j(\widehat{\mathbf{v}}_i, \mathbf{v}_{-i})) \\ &+ v_i p_i(\widehat{\mathbf{v}}_i, \mathbf{v}_{-i}) \left(1 - \prod_{j \in S(i)} (1 - p_j(\widehat{\mathbf{v}}_i, \mathbf{v}_{-i})) \right) \\ &- m_i(\widehat{\mathbf{v}}_i, \mathbf{v}_{-i}). \end{aligned} \quad (1)$$

Therefore, the LP relaxation of the seller's Revenue Maximization Problem (General-RMP) is

$$\max_{\{p_i, m_i\}_{i=1}^N} \sum_{i=1}^N \int m_i(\mathbf{v}_i, \mathbf{v}_{-i}) dF(\mathbf{v})$$

subject to

$$\text{(EPIC)} \quad U_i(\mathbf{v}_i, \mathbf{v}_i, \mathbf{v}_{-i}) \geq U_i(\widehat{\mathbf{v}}_i, \mathbf{v}_i, \mathbf{v}_{-i}) \\ \text{for all } i \text{ and all } \mathbf{v}_i, \widehat{\mathbf{v}}_i, \mathbf{v}_{-i},$$

$$\text{(EPIR)} \quad U_i(\mathbf{v}_i, \mathbf{v}_i, \mathbf{v}_{-i}) \geq 0 \\ \text{for all } i \text{ and all } \mathbf{v}_i, \mathbf{v}_{-i},$$

$$\text{(Feasibility)} \quad \sum_{i=1}^n p_i(\mathbf{v}) \leq K \text{ and } 0 \leq p_i(\mathbf{v}) \leq 1 \\ \text{for all } i,$$

where (EPIC) is the ex-post incentive compatibility constraint to ensure truth-telling and (EPIR) is the ex-post individual rationality constraint to ensure participation. (In the rest of this paper, we simplify the notation by denoting $U_i(\mathbf{v}_i, \mathbf{v}_i, \mathbf{v}_{-i})$ by $U_i(\mathbf{v}_i, \mathbf{v}_{-i})$.)

Problem (General-RMP), as well as the corresponding social surplus maximization problem, is a multi-dimensional mechanism design problem which is extremely difficult to solve analytically. It is well known that solutions to multi-

dimensional mechanism design problems are sensitive to various details of the environment, e.g., the seller's belief about the agents' types. Hence, there is little hope for finding closed-form solutions. This unappealing feature of multi-dimensional mechanism design problems has been demonstrated by [15], [1], and [13]. Furthermore, a numerical approach to solving this problem also has a limited potential given that even simplistic instances exhibit computational complexity obstacles: for example, even if $\mathbf{v}_i = (1, 0)$ for all i (i.e., agents only value exclusivity and this valuation is the same for all agents and is publicly known, so there is no private information at all in this setting), the Problem (General-RMP) reduces to determining whether there exists a K -independent set on (N, E) .

To avoid this inherent analytical obstacle stemming from mechanism design, we focus on valuation structures that have one-dimensional representation. The mechanism design problem can be solved for a large class of such valuations, including additive premium valuations $w_i = v_i + b_i$ or multiplicative premium valuations $w_i = b_i v_i$, where b_i is publicly known. Here, we illustrate our findings with modular valuations which we name *local linear exclusivity* (LLE):

$$w_i = v_i + \sum_{j \in S(i)} \alpha_{ij} v_j \quad (2)$$

with publicly known non-negative matrix $A = [\alpha_{ij}]$. Also of interest is a special case of LLE, where for every j ,

$$\sum_{\{i: j \in S(i)\}} \alpha_{ij} \leq 1. \quad (3)$$

We say that such valuations satisfy *bounded local linear exclusivity* (BLLE).

PROPOSITION 1. *Suppose that valuations are BLLE and publicly known and that the seller's supply is unlimited, i.e., $K \geq n$. Then, allocating an item to every buyer i who pays v_i is an optimal solution to the (FB-RMP) problem.*

Hence, the problem is trivial with publicly known BLLE valuations. However, introducing private values drastically changes the complexity of the problem.

PROPOSITION 2. *Suppose that valuations are BLLE, and that the seller's supply is unlimited, i.e., $K \geq n$. Then allocating exclusively to some buyers could be optimal. Furthermore, finding a deterministic optimal solution to the (SB-RMP) problem is at least as hard as finding the maximum independent set in (N, E) , even if virtual valuations $\psi_i \geq 0$ for all i .*

3. GENERALIZED EXTERNALITIES

We establish the following results within the exclusivity model and hence it extends to a generalized negative externalities setting.

THEOREM 1. *The expected revenues from the optimal mechanism are not monotone with respect any dimension of agents' valuations.*

This result is in stark contrast with standard mechanism literature that assumes no externalities. In fact, even with positive externalities, the monotonicity of the expected revenues in the optimal mechanism is established in a straightforward manner.

A direct consequence of this result involves rivalry networks.

COROLLARY 1. *The expected revenues of the optimal mechanism for a given rivalry network are not monotone with respect to arc addition/deletion.*

Finally, consider the setting in which the seller could also design/impose a rivalry network and answer the question of the (non)-existence of an optimal (extremal) rivalry network. In general, such an extremal network could depend on the distributional assumptions on the bidder valuations. However, we show that in some restricted settings such an extremal network not only exists, but is also independent of distributional assumptions.

THEOREM 2. *Suppose there exists $v > 0$ such that*

$$v_i = v > 0 > -\alpha_i$$

for all agents i and that all values are publicly known. Suppose that the seller has $k = 1$ item to allocate. Then there exists extremal networks that maximize seller's revenue across all possible rivalry networks.

4. REFERENCES

- [1] M. Armstrong. Multiproduct nonlinear pricing. *Econometrica*, 64(1):51–75, Jan. 1996.
- [2] A. Belloni, C. Deng, and S. Pekeč. Mechanism and network design with private negative externalities. Technical report, Duke University, 2015.
- [3] S. Bhattacharya, J. Kulkarni, K. Munagala, and X. Xu. On allocations with negative externalities. In *WINE*, pages 25–36, 2011.
- [4] Z. Cao, X. Chen, X. Hu, and C. Wang. Pricing in social networks with negative externalities. In *International Conference on Computational Social Networks*, pages 14–25. Springer, 2015.
- [5] J. Chen and S. Micali. Auction revenue in the general spiteful-utility model. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science*, pages 201–211. ACM, 2016.
- [6] P.-A. Chen and D. Kempe. Bayesian auctions with friends and foes. In M. Mavronicolas and V. Papadopoulou, editors, *Lecture Notes in Computer Science*, volume 5814, pages 335–346. Springer Berlin / Heidelberg, 2009.
- [7] V. Conitzer and T. Sandholm. Computing optimal outcomes under an expressive representation of settings with externalities. *Journal of Computer and System Sciences*, 78(1):2 – 14, 2012.
- [8] F. Constantin, M. Rao, C.-C. Huang, and D. C. Parkes. On expressing value externalities in position auctions. In *6th Workshop on Ad Auctions (at EC-10)*, 2010.
- [9] C. Deng and S. Pekeč. Money for nothing: exploiting negative externalities. In *Proceedings of the 12th ACM conference on Electronic commerce*, pages 361–370, San Jose, California, USA, 2011. ACM.
- [10] C. Deng and S. Pekeč. Optimal allocation of exclusivity contracts. Technical report, Duke University, 2013.
- [11] A. Ghosh and A. Sayedi. Expressive auctions for externalities in online advertising. In *Proceedings of the 19th international conference on World Wide Web, WWW '10*, pages 371–380, New York, NY, USA, 2010. ACM.
- [12] N. Haghpanah, N. Immorlica, V. Mirrokni, and K. Munagala. Optimal auctions with positive network externalities. In *Proceedings of the 12th ACM conference on Electronic commerce, EC '11*, pages 11–20, New York, NY, USA, 2011. ACM.
- [13] A. M. Manelli and D. R. Vincent. Multidimensional mechanism design: Revenue maximization and the multiple-good monopoly. *Journal of Economic Theory*, 137(1):153–185, Nov. 2007.
- [14] R. B. Myerson. Optimal auction design. *Mathematics of Operations Research*, 6(1):58–73, 1981.
- [15] J.-C. Rochet and P. Choné, P. Ironing, sweeping, and multidimensional screening. *Econometrica*, 66(4):783–826, July 1998.

The vertex sign balance of (hyper)graphs*

Dezső Miklós

Alfréd Rényi Institute of Mathematics,
Hungarian Academy of Sciences

Budapest P.O.B. 127 H-1364 Hungary

e-mail: dezso@renyi.hu

Joint work with J. Ahmann, E. Collins-Wildman,

J. Wallace, S. Yang and Yicong Guo

Budapest Semesters in Mathematics and

Gy. Y. Katona

Budapest University of Technology and Economics

Department of Computer Science and Information Theory

Budapest P.O.B. 91 H-1521 Hungary

e-mail: kiskat@cs.bme.hu

August 29, 2016

We define the vertex sign balance of a (hyper)graph G as the minimum number of non-negative edges over all $\omega : V(G) \rightarrow \mathbb{R}$ satisfying $\sum_{x \in V(G)} \omega(x) \geq 0$ (i.e., the minimum number of edges with non-negative sum of weights of vertices in it). Clearly, the vertex sign balance of a (hyper)graph is always less than or equal to the minimum degree, as it is shown by assigning a large positive number to a minimum degree vertex and close to 0 negative numbers to all other vertices. We will denote the vertex sign balance of a (hyper)graph by $\mu(G)$, as it will be shown closely related to the matching number of the (hyper)graph, $\nu(G)$. Huang and Sudakov [1], and Pokrovsky[3] (probably independently) introduced the so-called Manickam-Miklós-Singhi (MMS-)property of a hypergraph: a hypergraph H has the MMS-property if $\mu(H) = \delta(H)$, the minimum degree of the hypergraph. (The definition is based on the 30 years old conjecture of Manickam, Miklós[2], and Singhi, which says, in this language, that the complete r -uniform hypergraph on n vertices has the MMS-property if $n \geq 4r$.) Both of the papers (of Huang and Sudakov and Pokrovsky) used the notion of MMS-property to prove a better than earlier known bound on n to ensure the MMS-property of this complete r -uniform hypergraph on n vertices. Huang and Sudakov showed that every r -uniform n -vertex hypergraph with

*This research was partially supported by National Research, Development and Innovation Office – NKFIH Fund No. SNN-117879.

equal codegrees and $n > 10r^3$ has the MMS-property while Pokrovskiy proved that if a (any) d -regular r -uniform n -vertex hypergraph has the MMS-property, then the complete r -uniform hypergraph on n vertices has the MMS-property as well.

Here we will explore this newly introduced graph parameter, the vertex sign balance of graphs and — to some extent — hypergraphs.

Definition. The vertex sign balance of a (hyper)graph G (or H), denoted by $\mu(G)$ ($\mu(H)$, resp.) is defined to be the minimum number of non-negative edges for all weighting of vertices $\omega : V(G) \rightarrow \mathbb{R}$ satisfying $\sum_{x \in V(G)} \omega(x) \geq 0$, i.e. the minimum number of edges with non-negative sum of weights of vertices in it. For a given weight assignment to the vertices, an edge will be called positive (non-negative, negative) if the sum of the weights of the vertices in the edge are positive (non-negative, negative, resp.)

Remark. The vertex sign balance of a (hyper)graph is always less than or equal to the minimum degree.

Theorem 1. *For any graph G the following statements are equivalent:*

1. $\mu(G) \geq 1$
2. the fractional matching number, $\nu^*(G) = n/2$.
3. There exists no independent subset of vertices $S \subset V(G)$ such that $|N(S)| = |S| - 1$.
4. G has a perfect 2-matching, that is, there exists a collection of edges (multiple choice of an edges is allowed) which covers every vertex exactly twice.

We mention here that a perfect 2-matching can always be pictured as the union of disjoint odd cycles (whose edges are counted once) and a matching, i.e., collection of disjoint edges (which are counted twice).

A similar theorem can be stated for hypergraphs as well:

Theorem 2. *For r -uniform hypergraph $H = (V, E)$ with n vertices, $\mu(H) = 0$ if and only if the fractional matching number of H , $\nu^*(G)$, is less than $\frac{n}{r}$.*

The existence and structure of a perfect 2-matching of a graph can be used not only for determining whether $\mu(G) > 0$ but also to find the value of $\mu(G)$, as the following theorem gives:

Theorem 3. 1. $\mu(G) =$ the min # of edges one can remove from G to get G^* such that there exists $S^* \subset V(G)$ and $|S^*| > |N(S^*)|$.

2. $\mu(G) =$ the min # of edges one can remove from G so that the remaining graph does not have a perfect 2-matching.

This characterization of $\mu(G)$, together with some recent result on the computational complexity of the existence of a bounded size set of edges in a bipartite graph covering all maximal size matchings by Zenklusen, Ries, Picouleau, Werra, and Bentz[4] we managed to show that finding the vertex sign balance of a graph is NP-complete:

Theorem 4. *The following two problems about the vertex sign balance of a (hyper) graph are NP – complete:*

*VS*B(G, k)

Instance: An undirected (hyper) graph $G = (V, E)$ and a positive integer $0 \leq k \leq |E|$.

Question: Is $\varepsilon(G) \leq k$?

*VS*B(G)

Instance: An undirected (hyper) graph $G = (V, E)$

Question: Is $\varepsilon(G) < \delta(G)$, that is, is it true that G does not have the MMS-property?

Further, we succeeded in characterizing some classes of graphs which have MMS property and gave lower bounds on $\mu(G)$ in terms of the minimum degree (which is constant in case of regular graphs) of the graphs:

Theorem 5. *Suppose G is a graph with n many vertices. If $\delta(G) \geq \frac{n^2-1}{2n-6}$, then G has MMS-property.*

The following theorem will give the exact bounds on $\delta(G)$, the minimum degree of a graph to ensure the MMS-property.

Theorem 6. *(Sharp bounds for minimum degree) For any graph G with n vertices where $n \geq 6$, if n is odd and the minimum degree $\delta(G) \geq \frac{n+5}{2}$, or n is even and $\delta(G) \geq \frac{n+2}{2}$, then G has MMS-property. This bound is sharp.*

In case the minimal degree is smaller than the required lower bound, it still implies high vertex sign balance value (though not as high as δ).

Theorem 7. *For any graph G with n many vertices, if $\delta(G) \geq \frac{n^2+8t-1}{2n+2}$, then $\mu(G) \geq t$ for all $t \leq \delta(G)$.*

The following theorems will give lower bound on the vertex sign balance for regular graphs:

Theorem 8. *For any k -regular graph G , $\mu(G) \geq \frac{k}{2}$*

Theorem 9. *For a k -regular graph, G , with n vertices, $\mu(G) = \frac{k}{2}$ if and only if G has an independent subset S such that $|S| = \frac{n-1}{2}$ (or G is disconnected and has some component with this property).*

Corollary 1. *For a k -regular graph G with n vertices, the lower bound $\mu(G) = \frac{k}{2}$ can only be achieved if G has an odd number of vertices and $k \leq \frac{n+1}{2}$. (These are necessary but not sufficient conditions)*

Corollary 2. *For a connected k -regular graph G on an even number of vertices, $\mu(G) \geq \lfloor \frac{k}{2} + 1 \rfloor$ is a sharp lower bound.*

References

- [1] HUANG, H., AND SUDAKOV, B., The Minimum Number of Nonnegative Edges in Hypergraphs, *The Electronic Journal of Combinatorics*, **21**(3), 2014, pp. 3-7.
- [2] MANICKAM, N., AND MIKLÓS, D., On the number of non-negative partial sums of a non-negative sum, in *Colloq. Math. Soc. Janos Bolyai*, Vol. **52**, pp. 385-392.
- [3] POKROVSKIY, A., A linear bound on the Manickam–Miklós–Singhi conjecture, *Journal of Combinatorial Theory, Series*, **133** 2015, pp. 280-306.
- [4] R. ZENKLUSEN, B. RIES, C. PICOULEAU, D. DE WERRA, M. -C. COSTA, AND C. BENTZ, Blockers and transversals, *Discrete Mathematics*, **309** 2009, pp. 4306-4314.

Packing tree degree sequences

[Extended Abstract]

Kristóf Bérczi
Department of Computer
Science Eötvös Loránd
University
Pázmány Péter sétány 1/c
1117 Budapest, Hungary
berkri@cs.elte.hu

Changshuo Liu
Budapest Semesters in
Mathematics
Bethlen Gábor tér 2
1071 Budapest, Hungary
cl20@princeton.edu

Zoltán Király
Department of Computer
Science Eötvös Loránd
University
Pázmány Péter sétány 1/c
1117 Budapest, Hungary
kiraly@cs.elte.hu

István Miklós^{*}
Rényi Institute
Reáltanoda u. 13-15
1053 Budapest, Hungary
miklos.istvan@renyi.mta.hu

ABSTRACT

A degree sequence $D = d_1, d_2, \dots, d_n$ is a series on non-negative integers. A degree sequence is graphical if there is a vertex labeled graph G in which the degrees of the vertices are exactly D . Such graph G is called a realization of D . The color degree matrix problem also known as edge disjoint realization, edge packing or graph factorization problem is the following: given a $c \times n$ degree matrix $D = \{\{d_{1,1}, d_{1,2}, \dots, d_{1,n}\}, \{d_{2,1}, d_{2,2}, d_{2,n}\}, \dots, \{d_{c,1}, d_{c,2}, d_{c,n}\}\}$, in which each row of the matrix is a degree sequence, decide if there is an ensemble of edge disjoint realizations of the degree sequences. Such set of edge disjoint graphs is called a realization of the degree matrix. A realization can also be presented as an edge colored simple graph, in which the edges with a given color form a realization of the degree sequence in a given row of the color degree matrix.

It is known that the color degree sequence problem is NP-complete even if the number of colors is 3. Here we consider a special case when two of the degree sequences are degree sequences of trees. We show that this special case is easy. We also show that the problem is still NP-complete if only one of the degree sequences is a degree sequence of a tree.

We also consider counting the number of solutions. We show that efficient approximations for the number of solutions exists as well as an almost uniform sampler exists if two degree

^{*}Secondary affiliation: SZTAKI, 1111 Budapest, Lágymányosi u. 11, Hungary

sequences are degree sequences of trees and they do not share common leaves.

1. INTRODUCTION

Packing degree sequences is related to discrete tomography. The central problem of tomography is to reconstruct spatial objects from lower dimensional projections. The discrete 2D version is to reconstruct a coloured grid from vertical and horizontal projections. In the simplest version, this problem is to reconstruct the colouring of an $n \times m$ grid with the requirement that each row and column has a specific number of entries for each colour. Such coloured matrix can be considered as a factorization of the complete bipartite graph $K_{n,m}$. Indeed, for each colour c_i , the 0-1 matrix obtained by replacing c_i to 1 and all other colours to 0 is an adjacency matrix of a simple bipartite graph such that the disjoint union of these simple graphs is $K_{n,m}$. The prescribed number of entries for each colour are the degrees of the simple bipartite graphs. Therefore, an equivalent problem is to give a factorization of the complete bipartite graph given prescribed degree sequences.

It is also possible to consider the simple (not bipartite) version of the graph factorization problem. Obviously, the sum of the degrees for each vertex must be $n - 1$ when the complete graph K_n is factorized. Therefore, if there are k degree sequences, the last degree sequence is unequivocally determined by the first $k - 1$ degree sequences.

When $k = 2$, the problem is reduced to the degree sequence problem, and can be solved in polynomial time [2, 3]. When $k = 3$, the problem already becomes NP-complete [1]. However, special cases are polynomial solvable even when $k = 3$. Such a special case is when one of the degree sequences is almost regular, that is, any two degrees differ at most by 1 [5].

In this paper we consider the case when $k = 3$ and two of the degree sequences are tree degree sequences. We show that

this special case is polynomial solvable. Some results on the solution space is also presented. We also provide a negative result: when only one of the degree sequences is tree degree sequence, the problem is still NP-complete.

2. PRELIMINARIES

In this section we give the definitions needed to state the theorems. The central problem in this paper is the colour degree sequence problem.

DEFINITION 1. A degree sequence $D = d_1, d_2, \dots, d_n$ is a series on non-negative integers. A degree sequence is graphical if there is a vertex labeled simple graph G in which the degrees of the vertices are exactly D . Such graph G is called a realization of D . The colour degree matrix problem is the following: given a $c \times n$ degree matrix $D = \{\{d_{1,1}, d_{1,2}, \dots, d_{1,n}\}, \{d_{2,1}, d_{2,2}, d_{2,n}\}, \dots, \{d_{c,1}, d_{c,2}, d_{c,n}\}\}$, in which each row of the matrix is a degree sequence, decide if there is an ensemble of edge disjoint realizations of the degree sequences. Such set of edge disjoint graphs is called a realization of the degree matrix.

Although it is well known, we also define the tree degree sequences and caterpillars.

DEFINITION 2. A degree sequence $D = d_1, d_2, \dots, d_n$ is called a tree sequence if $\sum_{i=1}^n d_i = 2n - 2$ and each degree is positive.

DEFINITION 3. A tree is a caterpillar if the non-leaf vertices form a path in it.

In this paper, we are using complexity classes which might be unfamiliar for non-expert readers, therefore we give the definition of them here.

DEFINITION 4. A decision problem is in NP if a non-deterministic Turing Machine can solve it in polynomial time. An equivalent definition is that a witness proving the “yes” answer to the question can be verified in polynomial time. A counting problem is in #P if it asks for the number of witnesses of a problem in NP.

DEFINITION 5. A counting problem in #P is in FP if there is a polynomial running time algorithm which gives the solution. It is #P – complete if any problem in #P can be reduced to it by a polynomial-time counting reduction.

DEFINITION 6. A counting problem in #P is in FPRAS (Fully Polynomial Randomized Approximation Scheme) if there exists a randomized algorithm such that for any instance x , and $\epsilon, \delta > 0$, it generates an approximation \hat{f} for the solution f , satisfying

$$P\left(\frac{f}{1+\epsilon} \leq \hat{f} \leq f(1+\epsilon)\right) \geq 1 - \delta \quad (1)$$

and the algorithm has a time complexity bounded by a polynomial of $|x|$, $1/\epsilon$ and $-\log(\delta)$.

The total variational distance $d_{TV}(p, \pi)$ between two discrete distributions p and π over the set X is defined as

$$d_{TV}(p, \pi) := \frac{1}{2} \sum_{x \in X} |p(x) - \pi(x)| \quad (2)$$

DEFINITION 7. A counting problem in #P is in FPAUS if there exists a randomized algorithm (a Fully Polynomial Almost Uniform Sampler that is also abbreviated as FPAUS) such that for any instance x , and $\epsilon > 0$, it generates a random element of the solution space following a distribution p satisfying

$$d_{TV}(p, U) \leq \epsilon \quad (3)$$

where U is the uniform distribution over the solution space, and the algorithm has a time complexity bounded by a polynomial of $|x|$, and $-\log(\epsilon)$.

3. PACKING TWO TREES

Our main result is about packing two tree sequences with no common leaves.

THEOREM 1. Let $D = d_1, d_2, \dots, d_n$ and $F = f_1, f_2, \dots, f_n$ be two tree degree sequences, such that $\min_i \{d_i + f_i\} \leq 3$. Then D and F have edge disjoint caterpillar realizations.

The theorem implicitly states that if two degree sequences do not share common leaves then their sum is graphical. The proof of this theorem is skipped here. If the two trees have common leaves, their sum is not necessarily graphical. However, when their sum is graphical, they do have edge disjoint realizations, as Kundu already proved it.

THEOREM 2. [6] Let $D = d_1, d_2, \dots, d_n$ and $F = f_1, f_2, \dots, f_n$ be two tree degree sequences. Then there exist edge disjoint tree realizations of D and F iff $D + F (= d_1 + f_1, d_2 + f_2, \dots, d_n + f_n)$ is graphical.

We also give another theorem that also provide edge disjoint realizations.

THEOREM 3. Let $D = d_1, d_2, \dots, d_n$ and $F = f_1, f_2, \dots, f_n$ be two tree degree sequences, such that $\min_i \{d_i + f_i\} \leq 3$. Let T_1 and T_2 be random realizations of D and F uniformly distributed. Then the expected number of common edges of T_1 and T_2 is strictly less than 1 if there exists a vertex which is not a leaf in both trees and at most 1 if each vertex is a leaf in exactly one of the trees.

If there is a vertex which is not a leaf in both trees then there must exist edge disjoint realizations T_1 and T_2 , otherwise the number of common edges cannot be less than 1 in expectation. If each vertex is a leaf in exactly one of the trees then there must be vertices v_1 and v_2 which have degree 1 in D and v_3 and v_4 which have degree 1 in F (recall that any tree contains at least 2 leaves). Then there exist a pair of trees T_1 and T_2 such that both trees contain edges

(v_1, v_3) and (v_2, v_4) . Indeed, the degree 1 vertices can be connected to any of the non-leaf vertices. This means that there are trees having at least 2 common edges, which is above the average. However, then there must be a pair of trees with less than average number of common edges. That is, they are edge disjoint realizations.

Now we turn to the case when there are common leaves. The following lemma helps here.

LEMMA 4. *Let $D = d_1, d_2, \dots, d_n$ and $F = f_1, f_2, \dots, f_n$ be two tree degree sequences, such that $d_1 + f_1 \geq d_2 + f_2 \geq \dots \geq d_n + f_n$ is graphical. Define*

$$i := \min\{j | j > 1 \wedge [(d_1 > 1 \wedge f_j > 1) \vee (f_1 > 1 \wedge d_j > 1)]\} \quad (4)$$

Furthermore, assume that $d_n + f_n = 2$. Then the degree sequence is also graphical which is obtained from $D + F$ by removing 1 from $d_1 + f_1$ and $d_i + f_i$ and deleting $d_n + f_n$.

This lemma says that we can construct an edge disjoint realization of D and F by iteratively removing the common leaves and modifying the remaining degree sequences, and the lemma guarantees that the remaining degree sequence will be graphical. Once there is no common leaf, then we can apply Theorem 1. The so obtained caterpillar realizations can be extended to edge disjoint realizations of the original degree sequences by adding back the common leaves.

4. COUNTING AND SAMPLING REALIZATIONS

Since typically there are more than one realizations when a realization exists, and typically the number of realizations might grow exponentially, is also a computational challenge to estimate their number and/or sample almost uniformly a solution. Here we have the following theorem.

THEOREM 5. *Let $D = d_1, d_2, \dots, d_n$ and $F = f_1, f_2, \dots, f_n$ be two tree degree sequences, such that $\min_i \{d_i + f_i\} \leq 3$. Then there is an FPRAS for estimating the number of disjoint realizations and there is an FPAUS to almost uniformly sample realizations.*

This theorem is based on Theorem 3. If there is a vertex which is not a leaf then it is easy to show that the expected number of common edges is polynomially separated from 1, that is, the inverse of 1 minus the expectation is polynomially bounded. It means that in a polynomial number of trials of random couple of trees, at least one edge disjoint realization is expected. It follows from the central limit theorem that an FPRAS algorithm can be designed based on this property. It is also well known that an FPAUS algorithm can be designed in this case, see [4] for technical details.

When each vertex is a leaf in exactly one of the trees than it is easy to show that a non-negligible fraction of the random pair of trees contains at least two common edges. In fact, the inverse of the fraction of the couple of trees T_1 and T_2 that have the above mentioned common edges (v_1, v_3) and (v_2, v_4) is polynomially bounded. Then the inverse of

the fraction of edge disjoint realizations is also polynomially bounded. This means that FRPAS and FPAUS algorithms can be designed in exactly the same way than above.

It remains an open question whether or not similar theorems exist for the case when the tree degree sequences have common leaves. Also it is open if exact counting of the edge disjoint solutions is possible in polynomial time, although the natural conjecture is that this counting problem is #P-complete.

5. AN NP-COMPLETE THEOREM

What can we say when only one of the degree sequences is a tree degree sequence and the other is arbitrary? Unfortunately, we have a negative result here.

THEOREM 6. *It is NP-complete to decide if there is an edge disjoint realization of a tree degree sequence and an arbitrary degree sequence. (It is not required that the tree degree sequence have a tree realization).*

PROOF. We use the theorem by [1] that it is NP-complete to decide if two bipartite degree sequences has an edge disjoint realizations. We have the following observations.

- A bipartite degree sequence pair

$$D = (d_{1,1}, d_{1,2}, \dots, d_{1,n_1}), (d_{2,1}, d_{2,2}, \dots, d_{2,n_2})$$

and

$$F = (f_{1,1}, f_{1,2}, \dots, f_{1,n_1}), (f_{2,1}, f_{2,2}, \dots, f_{2,n_2})$$

has an edge disjoint realization iff the simple degree sequence pair

$$D' = (d_{1,1} + n_1 - 1, \dots, d_{1,n_1} + n_1 - 1, d_{2,1}, \dots, d_{2,n_2})$$

and

$$F' = (f_{1,1}, \dots, f_{1,n_1}, f_{2,1} + n_2 - 1, \dots, f_{2,n_2} + n_2 - 1)$$

has an edge disjoint realization. Indeed, if an edge disjoint bipartite realization of D and F is given, then the complete graph on the first vertex class can be added to the first realization and the complete graph on the second vertex class can be added to the second realization to get a (now non-bipartite) realization of D' and F' . On the other hand, it is easy to see that any realization of D' contains K_{n_1} on the first n_1 vertices, and any realization of F' contains K_{n_2} on the last n_2 vertices. Given an edge disjoint realization of D' and F' , deleting K_{n_1} from D' and K_{n_2} from F' yields an edge disjoint realization of D and F .

- The degree sequence pair $D = d_1, d_2, \dots, d_n$ and $F = f_1, f_2, \dots, f_n$ has an edge disjoint realization iff the degree sequence pair $D' = d_1 + 1, d_2 + 1, \dots, d_n + 1, n$ and $F' = f_1, f_2, \dots, f_n, 0$ has an edge disjoint realization. Indeed, let G_1 and G_2 be an edge disjoint realization of D and F . Then add a vertex v_{n+1} to G_1 , and connect it to all the other vertices to get a realization of D' . Add an isolated vertex v_{n+1} to G_2 to get a realization of F' . These realizations of D' and F' are edge disjoint. On the other hand, in any realization of D' ,

v_{n+1} is connected to all the other vertices. If edge disjoint realizations of D' and F' are given, delete v_{n+1} from both realizations to get edge disjoint realizations of D and F .

- The degree sequence pair $D = d_1, d_2, \dots, d_n$ and $F = f_1, f_2, \dots, f_n$ has an edge disjoint realization iff the degree sequence pair $D' = d_1, d_2, \dots, d_n, 1, 1$ and $F' = f_1 + 1, f_2 + 1, \dots, f_n + 1, n, 0$ has an edge disjoint realization. Indeed, any edge disjoint realization G_1 and G_2 of D and F can be extended to an edge disjoint realization of D' and F' by adding two vertices v_{n+1} and v_{n+2} , and then connecting v_{n+1} to all v_1, \dots, v_n in G_2 and connecting v_{n+1} and v_{n+2} in G_1 . On the other hand, in any edge disjoint realizations G'_1 and G'_2 of D' and F' , v_{n+1} is connected to all v_1, \dots, v_n in G'_2 , therefore, v_{n+1} must be connected to v_{n+2} in G'_1 . Therefore deleting v_{n+1} and v_{n+2} yields an edge disjoint realization of D and F .

We can use the first observation to prove that it is also NP-complete to decide that two simple degree sequences have edge disjoint realizations. The second observation provides that it is NP-complete to decide if two degree sequences have edge disjoint realizations such that one of the degree sequences does not have 0 degrees. Finally, we can use the third observation to iteratively transform any D degree sequence (that already does not have a 0 degree) to a tree degree sequence. Indeed, in each step, we add two vertices to D and extend the sum of the degrees only by 2. Therefore in a polynomial number of steps, we get a degree sequence D' in which the sum of the degrees is exactly twice the number of vertices minus 2. Therefore it follows that given any bipartite degree sequences D and F , we can construct in polynomial time two simple degree sequences D' and F' such that D and F have edge disjoint realizations iff D' and F' have edge disjoint realizations, furthermore, D' is a tree degree sequence.

□

6. DISCUSSION AND CONCLUSIONS

In this paper, we considered packing tree degree sequences. Our main theorem is that two tree degree sequences have edge disjoint tree realizations iff their sum is graphical. This is similar to the Kundu's theorem [5] stating that a degree sequence and an almost regular degree sequence have an edge disjoint realization iff their sum is graphical. This raises the natural question if a degree sequence and a tree sequence have edge disjoint realizations iff their sum is graphical. We showed that the answer is no to this question, and actually, it is NP-complete to decide if an arbitrary degree sequence and a tree degree sequence have edge disjoint realizations.

We also considered to approximately count and sample edge disjoint tree realizations with prescribed degrees. We showed that it is possible if there are no common leaves. It remains an open question when the two degree sequences have common leaves.

7. REFERENCES

- [1] Dürr, C., Guinez, F., Matamala, M.: Reconstructing 3-colored grids from horizontal and vertical projections is NP-hard. *European Symposium on Algorithms*, 776–787 (2009)
- [2] S.L. Hakimi: On the degrees of the vertices of a directed graph. *J. Franklin Institute*, 279(4):290–308. (1965)
- [3] V. Havel: A remark on the existence of finite graphs. (Czech), *Časopis Pěst. Mat.* 80:477–480. (1955)
- [4] Jerrum, M.R., Valiant, L.G., Vazirani, V.V.: Random generation of combinatorial structures from a uniform distribution. *Theoretical Computer Science*, 43(2-3):169–188 (1986)
- [5] Kundu, S.: The k-factor conjecture is true. *Discrete Mathematics*, 6(4):367–376. (1973)
- [6] Kundu, S.: Disjoint Representation of Tree Realizable Sequences. *SIAM Journal on Applied Mathematics*, 26(1):103–107. (1974)

Benchmark problems for exhaustive exact maximum clique search algorithms

Sandor Szabo
Institute of Mathematics and Informatics
University of Pecs
sszabo7@hotmail.com

Bogdan Zavalnij
Institute of Mathematics and Informatics
University of Pecs
bogdan@ttk.pte.hu

ABSTRACT

There are well established widely used benchmark tests to assess the performance of practical exact clique search algorithms. In this paper a family of further benchmark problems is proposed mainly to test exhaustive clique search procedures.

Keywords

clique, maximum clique, random graph

Let $G = (V, E)$ be a finite simple graph. Here V is the set of vertices of G and E is the set of edges of G . Let C be a subset of V . If two distinct nodes in C are always adjacent in G , then C is called a clique in G . When C has k elements, then we talk about a k -clique. A k -clique is a maximum clique in G if G does not contain any $(k + 1)$ -clique. We call this well defined number the clique number of G and we denote it by $\omega(G)$.

A number of problems is referred as clique search problems.

PROBLEM 1. *Given a finite simple graph G and given a positive integer k . Decide if G has a k -clique.*

PROBLEM 2. *Given a finite simple graph G . Determine $\omega(G)$.*

The complexity theory of the algorithm tells us that Problem 1 is in the NP-complete complexity class. (See for instance [2].) Consequently, Problem 2 must be NP-hard. Loosely speaking it can be interpreted such that the maximum clique problem is computationally demanding.

As at this moment there are no readily available mathematical tools to evaluate the performance of practical clique search algorithms, the standard procedure is to carry out numerical experiments on a battery of well selected benchmark tests.

The most widely used test instances are the Erdős–Rényi random graphs, graphs from the second DIMACS challenge¹, combinatorial problems of monotonic matrices [5], and hard coding problems of Deletion-Correcting Codes².

Evaluating the performances of various clique search algorithms is a delicate matter. On one hand one would like to reach some practically relevant conclusion about the competing algorithms. On the other hand this conclusion is based on a finite list of instances.

One has to be ever cautious not to draw overly sweeping conclusions from these inherently limited nature experiments. (We intended to contrast this approach to the asymptotic techniques which are intimately tied to infinity.) The situation is of course not completely pessimistic. After all, these benchmarks were successful at shedding light on the practicality of many of the latest clique search procedures. However, we should strive for enhancing the test procedures. The main purpose of this paper is to propose new benchmark instances.

There are occasions when we are trying to locate a large clique in a given graph such that the clique is not necessarily optimal. This approach is referred as non-exact method to contrast it to the exhaustive search. For instance constructing a large time table in this way can be practically important and useful even without a certificate of optimality.

The benchmark tests are of course relevant in connection with non-exact procedures too. In order to avoid any unnecessary confusion we would like emphasize that in this paper we are focusing solely on the exact clique search methods.

Let n be a positive integer and let p be a real number such that $0 \leq p \leq 1$. An Erdős–Rényi random graph with parameters n, p is a graph G with vertices $1, 2, \dots, n$. The probability that the unordered pair $\{x, y\}$ is an edge of G is equal to p for each $x, y, 1 \leq x < y \leq n$. The events that the distinct pairs

$$\{x_{i(1)}, y_{i(1)}\}, \dots, \{x_{i(s)}, y_{i(s)}\}$$

are edges of G are independent of each other for each subset $\{i(1), \dots, i(s)\}$ of $\{1, 2, \dots, n\}$, where $s \geq 2$.

¹<ftp://dimacs.rutgers.edu/pub/challenge/>

²<http://neilsloane.com/doc/graphs.html>

In a more formal way the Erdős-Rényi random graph of parameters n, p is a random variable whose values are all the simple graphs with n vertices. The probability distribution over these graphs is specified in the manner we have described above. In this paper we can work safely in a more intuitive level. We start with a complete graph on n vertices and we decide the fate of each edge by flipping a biased coin.

In the case $p = 0$ we end up with a graph consisting of n isolated nodes. In the case $p = 1$ we end up with a complete graph on n nodes. (Paper [1] is the basic reference on Erdős-Rényi random graph.)

Let l, n be positive integers. Let $H_i = (V_i, E_i)$ be a graph consisting of l isolated nodes. This means that $|V_i| = l$ and $E_i = \emptyset$ for each $i, 1 \leq i \leq n$. Let $V_i = \{v_{i,1}, \dots, v_{i,l}\}$. We construct a new graph $G = (V, E)$. We set $V = V_1 \cup \dots \cup V_n$. The nodes $v_{i,r}, v_{j,s}$ are connected by an edge in G whenever $i \neq j$. We may say that the graph G is isomorphic to the lexicographic product of the graphs H and K , where H consists of l isolated nodes and K is the complete graph on n nodes. (For further details of graph products see [3].)

Clearly, V_i is an independent set in G for each $i, 1 \leq i \leq n$. The subgraph induced by $V_i \cup V_j$ in G is a complete bipartite graph for each $i, j, 1 \leq i < j \leq n$. Obviously, $\chi(G) = n$ and $\omega(G) = n$ hold. In fact G contains l^n distinct n -cliques.

At this stage we choose a real number p such that $0 \leq p \leq 1$. At each edge of G we flip a biased coin. The edge stays with probability p . We call this step randomizing G . The resulting random graph G' belongs to the parameters l, n, p . The $l = 1$ particular case corresponds to the Erdős-Rényi random graph of parameters n, p .

It is clear that $\chi(G') \leq n$ and $\omega(G') \leq n$. In order to guarantee that $\omega(G') = n$ holds we will plant an n -clique into G' . One can achieve this by picking $x_i \in V_i$ for each $i, 1 \leq i \leq n$ and connect each distinct pairs among x_1, \dots, x_n by an edge in G' .

Benchmark tests based on these random graphs are collected in the BHOSLIB library.³ (The acronym BHOSLIB stands for Benchmarks with Hidden Optimum Solutions Library.)

After all these preparations we are ready to describe the graphs we would like to propose for testing clique search algorithms. Let k, m be positive integers. Let $M_i^{(k)} = (V_i, E_i)$ be the Mycielski graph of parameter k . (For the definition of Mycielski graphs see [4].) Let $V_i = \{v_{i,1}, \dots, v_{i,n}\}$ for each $i, 1 \leq i \leq m$. We construct a new graph $G = (V, E)$. We set $V = V_1 \cup \dots \cup V_m$. Let $v_{i,r}, v_{i,s} \in V_i$. If the unordered pair $\{v_{i,r}, v_{i,s}\}$ is an edge of $M_i^{(k)}$, then we add this pair as an edge to G . These edges will be the blue edges of G . In other words the subgraph induced by V_i in G is isomorphic to $M_i^{(k)}$ for each $i, 1 \leq i \leq m$.

Pick $v_{i,r} \in V_i, v_{j,s} \in V_j$. We connect the nodes $v_{i,r}, v_{j,s}$ by an edge in G whenever $i \neq j$. These edges will be the red edges of G .

³<http://www.nlsde.buaa.edu.cn/~kexu/benchmarks/graph-benchmarks.htm>

Note that the graph G is isomorphic to the lexicographic product of the graphs $M^{(k)}$ and K , where $M^{(k)}$ is the Mycielski graph of parameter k and K is the complete graph on m nodes. One can verify that $\chi(G) = (k)(m)$ and $\omega(G) = (2)(m)$.

We choose a real number p such that $0 \leq p \leq 1$. We randomize the red edges of G . We flip a biased coin and keep each red edge with probability p . The resulted random graph is denoted by G' . It is obvious that $\chi(G') \leq (k)(m)$ and $\omega(G') \leq (2)(m)$. By planting a $(2m)$ -clique into G' we can guarantee that $\omega(G') = (2)(m)$. We pick $x_i, y_i \in V_i$ such that the unordered pair $\{x_i, y_i\}$ is an edge in G' for each $i, 1 \leq i \leq m$. Finally, we construct a $(2m)$ -clique whose nodes are $x_1, y_1, \dots, x_m, y_m$.

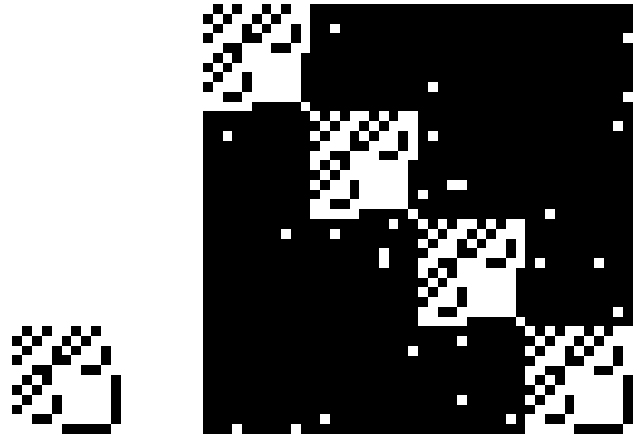


Figure 1: The adjacency matrices of the Mycielski graph $M^{(4)}$ and the random graph G' .

Note that other graphs can be used instead of the Mycielski graphs. Presumably the kind of graphs where the clique number is far from the chromatic number. Using this method we constructed several test problems. The proposed new collection of test graphs can be found on the site clique.ttk.pte.hu/evil. The source code of the program that generates the adjacency matrices of these graphs are also available on this site.

We carried out a large scale numerical experiment to check the proposed EVIL benchmark problems. We used 55 test graphs. We took 35 BHOSLIB graphs and 20 EVIL graphs. The experiment involved 7 programs implementing 12 different algorithms and so we are able to compare the running times of 660 clique searches. We shall present the results in the extended version of our paper. One particular result was that there is a test graph with 220 nodes – 20 copies of the $M^{(4)}$ graph, $p = 98\%$ edge probability – whose clique number could be determined by only one program in slightly less than 12 hours. We suppose that this problem is the hardest one of such small size.

We would like to close the paper with a few remarks why the reader should appreciate the proposed benchmark problems. Although it seems that there is a large number of

benchmark problems for maximum clique search the plain fact is that there are not enough of them. Many of these test problems are too easy for the modern solvers as the sizes of these problems are small. On the other hand there are test instances that are overly hard for the contemporary clique solvers. The proposed EVIL test graphs are forming parameterized families. The parameters can be tuned to produce benchmark problems in various degrees of difficulty.

Acknowledgments

This research was supported by National Research, Development and Innovation Office – NKFIH Fund No. SNN-117879 and the Pécsi Tudományegyetem Alapítvány.

1. REFERENCES

- [1] P. Erdős, A. Rényi, On the evolution of random graphs, *Magyar Tud. Akad. Mat. Kutató Int. Közl.* **5** (1960), 17–61.
- [2] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness*, Freeman, New York, 2003.
- [3] R. Hammack, W. Imrich, S. Klavžar, *Handbook of Product Graphs*, CRC Press, Boca Raton, FL, 2011.
- [4] J. Mycielski, Sur le coloriage des graphes, *Colloq. Math.* **3** (1955), 161–162.
- [5] E. W. Weisstein, Monotonic Matrix, In: *MathWorld—A Wolfram Web Resource*. <http://mathworld.wolfram.com/MonotonicMatrix.html>

On embedding degree sequences

[Extended Abstract]

Béla Csaba^{*}

Bolyai Institute

University of Szeged

6720 Szeged, Hungary, Aradi vértanúk tere 1.

bcsaba@math.u-szeged.hu

Bálint Vásárhelyi[†]

Bolyai Institute

University of Szeged

6720 Szeged, Hungary, Aradi vértanúk tere 1.

mesti90@gmail.com

ABSTRACT

Assume that we are given two graphic sequences, π_1 and π_2 . We consider conditions for π_1 and π_2 which guarantee that there exists a simple graph G_2 realizing π_2 such that G_2 is the subgraph of any simple graph G_1 that realizes π_1 .

Categories and Subject Descriptors

G.2.2 [Graph theory]: Extremal graph theory; Matchings and factors; Graph coloring

General Terms

Graph theory

Keywords

degree sequence, embedding, extremal graph theory

1. INTRODUCTION

All graphs considered in this paper are simple. We use standard graph theory notation, see for example [4]. Let us provide a short list of a few perhaps not so common notions, notations. Given a bipartite graph $G(A, B)$ we call it *balanced* if $|A| = |B|$. This notion naturally generalizes for r -partite graphs with $r \in \mathbb{N}$, $r \geq 2$.

If $S \subset V$ for some graph $G = (V, E)$, then the subgraph spanned by S is denoted by $G[S]$. Moreover, let $Q \subset V$ so that $S \cap Q = \emptyset$, then $G[S, Q]$ denotes the bipartite subgraph of G on vertex classes S and Q , having every edge of G that connects a vertex of S with a vertex of Q . The number of edges of a graph is denoted by $e(G)$. The chromatic number of a graph G is $\chi(G)$. The complete graph on n vertices is denoted by K_n , the complete bipartite graph with vertex class sizes n and m is denoted by $K_{n,m}$.

^{*}Partially supported by ERC-AdG. 321400 and by the National Research, Development and Innovation Office - NK-FIH Fund No. SNN-117879.

[†]Supported by TÁMOP-4.2.2.B-15/1/KONV-2015-0006.

A finite sequence of natural numbers $\pi = (d_1, \dots, d_n)$ is a *graphic sequence* or *degree sequence* if there exists a graph G such that π is the (not necessarily) monotone degree sequence of G . Such a graph G *realizes* π . The largest value of π is denoted by $\Delta(\pi)$. We sometimes refer to the value of π at vertex v as $\pi(v)$.

Let G and H be two graphs on n vertices. They *pack* if there exist edge-disjoint copies of G and H in K_n . Two degree sequences π_1 and π_2 *pack*, if there are graphs G_1 and G_2 realizing π_1 and π_2 , respectively, such that G_1 and G_2 pack. Equivalently, G_1 and G_2 pack if and only if $G_1 \subset \overline{G_2}$, that is, G_1 can be embedded into $\overline{G_2}$, where \overline{G} denotes the *complement* of G .

It is an old and well-understood problem in graph theory to tell whether a given sequence of natural numbers is a degree sequence or not. We consider a generalization of it, which is remotely related to the so-called discrete tomography [3] (or degree sequence packing) problem as well[‡]. The question whether a sequence π of n numbers is a degree sequence can be formulated as follows: Does K_n have a subgraph H such that the degree sequence of H is π ? The question becomes more general if K_n is replaced by some (simple) graph G on n vertices. If the answer is yes, we say that π *can be embedded into* G , or equivalently, π *packs with* \overline{G} . In order to state our main result let $\delta(G)$ and $\Delta(G)$ denote the minimum and maximum degree of G , respectively. We prove the following.

THEOREM 1. *For every $\varepsilon > 0$ and $D \in \mathbb{N}$ there exists an $n_0 = n_0(\varepsilon, D)$ such that for all $n > n_0$ if G is a graph on n vertices with $\delta(G) \geq \frac{n}{2} + \varepsilon n$ and π is a degree sequence of length n with $\Delta(\pi) \leq D$, then π is embeddable into G .*

We also state Theorem 1 in an equivalent complementary form, as a packing problem.

THEOREM 2. *For every $\varepsilon > 0$ and $D \in \mathbb{N}$ there exists an $n_0 = n_0(\varepsilon, D)$ such that for all $n > n_0$ if π_1 and π_2 are graphic sequences of length n satisfying $\Delta(\pi_1) < (\frac{1}{2} - \varepsilon)n$ and $\Delta(\pi_2) \leq D$ then there exists a graph G_2 that realizes π_2 and packs with any G_1 realizing π_1 .*

It is easy to see that Theorem 1 is sharp up to the εn additive term. For that let n be an even number, and sup-

[‡]This relation is discussed in the full version of the paper

pose that every element of π is 1. Then the only graph that realizes π is the union of $n/2$ vertex disjoint edges. Let $G = K_{n/2-1, n/2+1}$ be the complete bipartite graph with vertex class sizes $n/2 - 1$ and $n/2 + 1$. Clearly G does not have $n/2$ vertex disjoint edges.

2. PROOF OF THEOREM 1

We are going to construct a 3-colorable graph H that realizes π and has the following properties. There exists $A \subset V = V(H)$ such that

- (1) $|A| \leq 5\Delta^3(\pi)$,
- (2) the components of $H[V - A]$ are balanced complete bipartite graphs, each having size at most $2\Delta(\pi)$,
- (3) $\chi(H[A]) = 3$ if A is non-empty, and
- (4) $e(H[A, V - A]) = 0$.

In order to construct H we will use two types of "gadgets". Type 1 gadgets are balanced complete bipartite graphs on $2k$ vertices, where $k \in \{1, \dots, \Delta(\pi)\}$, these are the components of $H[V - A]$. Type 2 gadgets are composed of at least two type 1 gadgets and at most two other vertices, these are the components of $H[A]$.

We find type 1 gadgets with the following algorithm.

ALGORITHM 3. Assign the elements of π arbitrarily to V . Set every vertex *active*. Let $k = 1$.

- Step 1** If there are at least $2k$ active vertices with degree k , then take any $2k$ such vertices, create a balanced complete bipartite graph on these $2k$ vertices, and then *unactivate* them.
- Step 2** If the number of active vertices with degree k drops below $2k$, set $k = k + 1$.
- Step 3** If $k \leq \Delta(\pi)$, then go to Step 1. Else stop the algorithm.

This way we obtain several components, each being a balanced complete bipartite graph. These are type 1 gadgets. It is easy to see that for every $k \in \{1, \dots, \Delta(\pi)\}$ at most $2k - 1$ vertices are left out from the union of type 1 gadgets, a total of at most $\Delta^2(\pi) - 2\Delta(\pi)$ vertices. Furthermore, if a vertex v belongs to some type 1 gadget, then its degree is exactly $\pi(v)$.

Let R denote the set of vertices that are *uncovered* by the above set of type 1 gadgets. As we noted earlier $|R| \leq \Delta(\pi)^2 - 2\Delta(\pi)$. In order to get the right degrees for the vertices of R we construct type 2 gadgets, using some type 1 gadgets as well.

Notice first that the sum of the degrees of the vertices of R must be an even number, hence, R_o , the subset of R containing the odd degree vertices, has an even number of elements. Find $|R_o|/2$ disjoint pairs in R_o , and join vertices by a new edge that belong to the same pair. With this we get that every vertex of R misses an even number of edges.

We construct the type 2 gadgets using the following algorithm.

ALGORITHM 4. Set every type 1 gadget *unmarked* and every vertex in $R - R_o$ uncolored.

- Step 1** Choose an uncolored vertex v from $R - R_o$ and color it.
- Step 2** Choose a type 1 unmarked gadget K and *mark* it.
- Step 3** Choose an arbitrary perfect matching M_K in K (M_K exists since K is a balanced complete bipartite graph).
- Step 4** Choose an arbitrary xy edge in M_K .
- Step 5** Replace the edge xy with the new edges vx and vy .
- Step 6** If v is still missing edges, then if M_K is not empty, go to Step 4, else go to Step 2.
- Step 7** If v reaches its desired degree and there are still uncolored vertices in $R - R_o$, then go to Step 1, else stop the algorithm

It is easy to see that in $\pi(v)/2$ steps v reaches its desired degree, while the degrees of vertices in the marked type 1 gadgets have not changed. It is straightforward to use this algorithm for vertices in R_o , since each of these miss an even number of edges.

Figure 1 shows examples of type 2 gadgets. Let $F \subset H$ denote the subgraph containing the union of all type 2 gadgets, thus $F = H[A]$. Observe that type 2 gadgets of F are 3-chromatic, and all have less than $5\Delta^2(\pi)$ vertices. This easily implies the following claim.

CLAIM 5. *We have that $|V(F)| \leq 5\Delta^3(\pi)$.*

We are going to show that $H \subset G$. For that we first embed the 3-chromatic part F using the following strengthening of the Erdős–Stone theorem proved by Chvátal and Szemerédi [1].

THEOREM 6. *Let $\varphi > 0$ and assume that G is a graph on n vertices where n is sufficiently large. Let $r \in \mathbb{N}$, $r \geq 2$. If*

$$|E(G)| \geq \left(\frac{r-2}{2(r-1)} + \varphi \right) n^2,$$

then G contains a $K_r(t)$, i.e. a complete r -partite graph with t vertices in each class, such that

$$t > \frac{\log n}{500 \log \frac{1}{\varphi}}. \quad (1)$$

Since $\delta(G) \geq n/2 + \varepsilon n$, the conditions of Theorem 6 are satisfied with $r = 3$ and $\varphi = \varepsilon/2$, hence, G contains a balanced complete tripartite subgraph T on $\Omega(\log n)$ vertices. Using Claim 5 and the 3-colorability of F this implies that $F \subset T$.

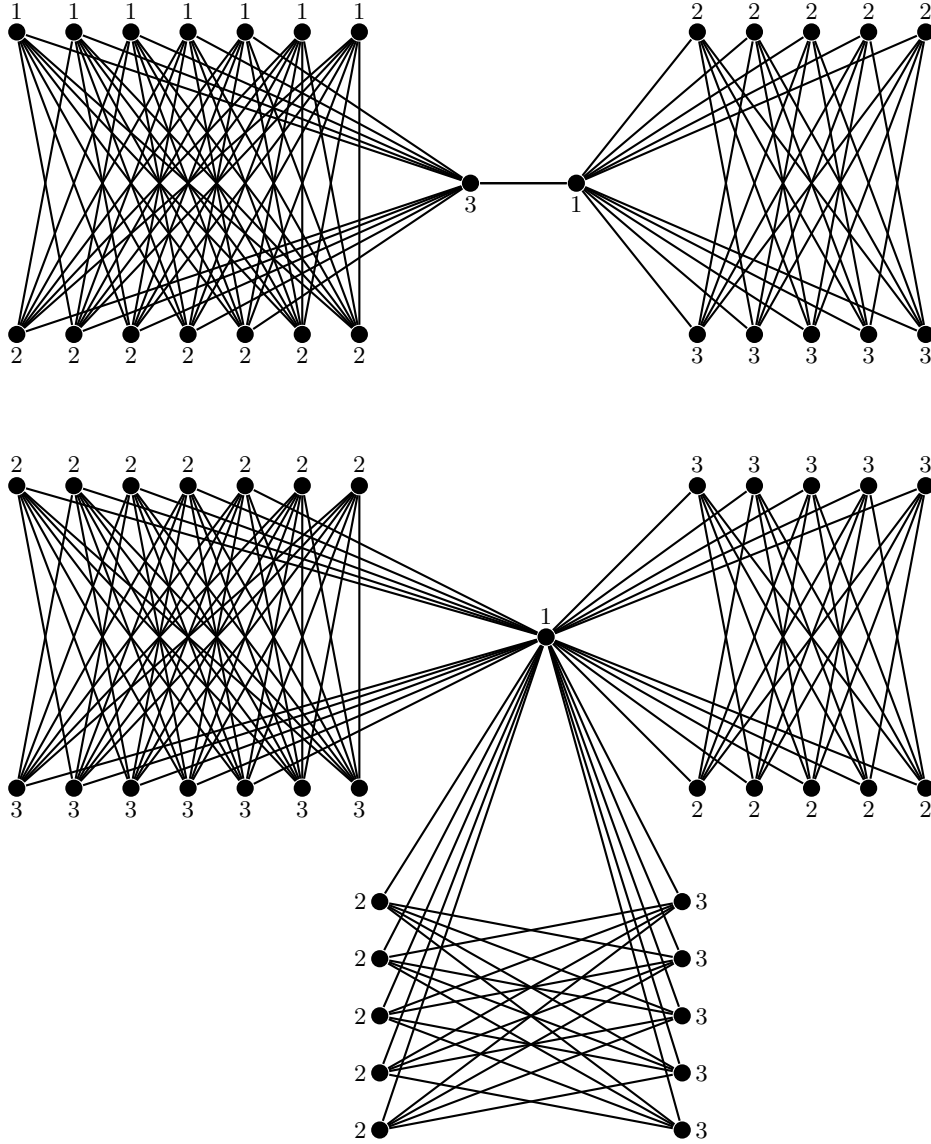


Figure 1: Type 2 gadgets of H with a 3-coloring

Observe that after embedding F into G every uncovered vertex still has at least $\delta(G) - v(F) > n/2 + \varepsilon n/2$ uncovered neighbors. Denoting the uncovered subgraph of G by G' we obtain that $\delta(G') > n/2 + \varepsilon n/2$.

We need a definition from [2].

DEFINITION 7. [2] *A graph H on n vertices is well-separable, if it has a subset $S \subset V(H)$ of size $o(n)$ such that all components of $H - S$ are of size $o(n)$.*

In order to prove that $H - F \subset G'$ we will apply a special case of the main theorem of [2], which is as follows:

THEOREM 8. [2] *For every $\gamma > 0$ and positive integer D there exists an n_0 such that for all $n > n_0$ if J is a bipartite well-separable graph on n vertices, $\Delta(J) \leq D$ and $\delta(G) \geq (\frac{1}{2} + \gamma)n$ for a graph G of order n , then $J \subset G$.*

Since $H - F$ has bounded size components, we can apply Theorem 8 for $H - F$ and G' , with parameter $\gamma = \varepsilon/2$. With this we finished proving what was desired.

3. A GENERALIZATION

While Theorem 1 is best possible up to the εn additive term, if π has a special property, one can claim much more as Theorem 9 shows below. Let us call a bipartite graph $H(A, B)$ *u-unbalanced* if $|A| = u|B|$ for some $u \in \mathbb{N}$. A bipartite degree sequence π is *u-unbalanced* if π can be realized by a *u-unbalanced* bipartite graph. We need the notion of *edit distance* of graphs: the edit distance between two graphs on the same labeled vertex set is defined to be the size of the symmetric difference of the edge sets.

A generalization of Theorem 1 is the following:

THEOREM 9. *For every $\varepsilon > 0$ and $D, u \in \mathbb{N}$ there exist an*

$n_0 = n_0(\varepsilon, u)$ and a $K = K(\varepsilon, D, u)$ such that if $n \geq n_0$, π is a u -unbalanced degree sequence of length n with $\Delta(\pi) \leq D$, G is a graph on n vertices with $\delta(G) \geq \frac{n}{u+1} + \varepsilon n$, then there exists a graph G' on n vertices so that the edit distance of G and G' is at most K , and π is embeddable into G' .

Hence, if π is unbalanced, the minimum degree requirement of Theorem 1 can be substantially decreased, what we pay for this is the "almost embedding" of π . For example, if π is a 10-unbalanced bounded degree sequence of length n and G is a graph on n vertices having $\delta(G) \geq n/11 + \varepsilon n$ for some $\varepsilon > 0$, then after deleting/adding a constant number (i.e. a function of ε^{\S}) of edges, we obtain a graph G' from G into which π can be embedded.

In another direction, one can also show that if π has little less elements than the number of vertices in G , then π can be embedded into G under very similar conditions.

THEOREM 10. *For every $\varepsilon > 0$ and $D, u \in \mathbb{N}$ there exist an $n_0 = n_0(\varepsilon, u)$ and an $M = M(\varepsilon, D, u)$ such that if $n \geq n_0$, π is a u -unbalanced degree sequence of length n with $\Delta(\pi) \leq D$, G is a graph on $n + M$ vertices with $\delta(G) \geq \frac{n+M}{u+1} + \varepsilon(n + M)$, then π is embeddable into G .*

The proofs of Theorem 9 and Theorem 10 are much more involved than that of Theorem 1, they are given in the full version of the paper. Let us note that the conditions for $\delta(G)$ are best possible in the above theorems up to the εn additive term.

4. REFERENCES

- [1] V. Chvátal and E. Szemerédi, *On the Erdős–Stone Theorem*, Journal of the London Mathematical Society **s2-23** (1981), no. 2, 207–214.
- [2] B. Csaba, *On embedding well-separable graphs*, Discrete Mathematics **308** (2008), 4322–4331.
- [3] J. Diemunsch, M.J. Ferrara, S. Jahanbekam, and J. M. Shook, *Extremal theorems for degree sequence packing and the 2-color discrete tomography problem*, SIAM Journal of Discrete Mathematics **29** (2015), no. 4, 2088–2099.
- [4] Douglas B. West, *Introduction to graph theory*, second ed., Prentice Hall, 2001.

[§]Unfortunately, it can be a tower function of $1/\varepsilon$

Computational complexity of the winner determination problem for geometrical combinatorial auctions

Dries Goossens
Faculty of Economics and
Business Administration,
Ghent University
Tweeerkenstraat 2, 9000
Gent
Belgium
dries.goossens@ugent.be

Bart Vangerven
Operations Research and
Statistics, Faculty of
Economics and Business, KU
Leuven
Naamsestraat 69, 3000
Leuven
Belgium

Frits C.R. Spieksma
Operations Research and
Statistics, Faculty of
Economics and Business, KU
Leuven
Naamsestraat 69, 3000
Leuven
Belgium

ABSTRACT

We consider auctions of items that can be arranged in rows, for instance pieces of land for real estate development. The objective is, given bids on subsets of items, to find a subset of bids that maximizes auction revenue (often referred to as the *winner determination problem*). We show that for a k -row problem with *connected* and *gap-free* bids, the winner determination problem can be solved in polynomial time, using a dynamic programming algorithm. We study the complexity for bids in a grid, complementing known results in literature. Additionally, we study variants of the geometrical winner determination setting. We provide a NP-hardness proof for the 2-row setting with gap-free bids. Finally, we extend this dynamic programming algorithm to solve the case where bidders submit connected, but not necessarily gap-free bids in a 2-row and a 3-row problem.

Keywords

Auctions, winner determination problem, computational complexity, rows, dynamic programming

1. INTRODUCTION

In combinatorial auctions, bidders can place bids on combinations of items, called packages or bundles. Clearly, combinatorial auctions allow bidders to better express their preferences compared to the traditional auction formats, where bidders place bids on individual items. In particular, it makes sense to use a combinatorial auction when complementarities or substitution effects exist between different items. For an introduction to combinatorial auctions, we refer to [5]; for a survey of the literature, we refer to [1] and [6].

One important challenge within this domain is, given the bids, to decide which items should be allocated to which

bidder, i.e., which bids to accept. In general, this *winner determination problem* is NP-hard [11], and does not allow good approximation results [10].

We discuss a combinatorial auction in a restricted topology. In this setting, an item corresponds to a rectangle, and all items are arranged in (a limited number of) rows, see Figure 1 for an example. Notice that the individual items (or

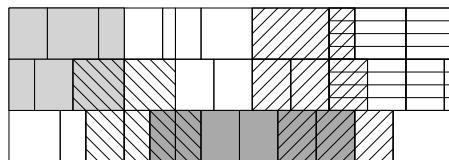


Figure 1: An example of an instance with 3 rows and 5 bids.

rectangles) need not have the same size. A bid consists of a set of items satisfying some restrictions (see Section 2 for a precise problem definition), together with a value. The objective is to select a set of bids that maximizes the sum of the expressed values, while making sure that each item is present at most once in a selected bid.

There are several situations in practice that motivate this specific geometric setting. We mention the following:

- Real estate. Goossens et al. [7] describe how space in a newly erected building, to be used for housing and commercial purposes, is allocated using a combinatorial auction. The geometric structure of each of the levels of the building features the properties described here. Quan [8] reports on empirical studies in real estate auctions. Several of these studies have focused on verifying and quantifying the *afternoon effect*. This afternoon effect describes similar items consistently selling for significantly less in later rounds in multi-object sequential auctions. A combinatorial auction, by selling all items simultaneously, can mitigate this effect.
- Mineral rights. Imagine a region that is partitioned into lots, with the lots organized in rows. For sale is the right to extract minerals, oil or gas found on or below the surface of the lot. Clearly, having adjacent

lots allows for exploration and production efficiencies, a complementarity. For more about this particular setting, we refer to [4]. Figure 2 shows an example of oil and gas leases neatly arranged in rows.

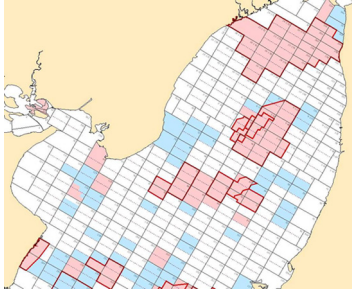


Figure 2: Oil and Gas Leases managed by the Texas General Land Office. Taken from: <http://www.glo.texas.gov/GLO/agency-administration/gis/gis-data.html>.

- Seats in a grandstand, theater or stadium. In some of these cases, one can even assume that a grid, consisting of rows and columns, is given where each cell represents a seat. Typically, demand exists for sets of adjacent seats - think of a family of four going to a ball game, or a group of friends visiting a concert. The complementarities that people perceive from adjacent seats offer possibilities for combinatorial auctions. Although tickets are usually sold at a fixed price, there are occasions where sports teams have auctioned off (part of) their seat licenses.

In all these cases, it is clear that complementarities between adjacent items exist; a combinatorial auction is best-placed to take these effects into account.

Goossens et al. [7] show that when a constraint is imposed stating that a bidder can have at most one winning bid, the winner determination problem is NP-hard even if all items are arranged on a single row. Hence, to have any prospect of coming up with a positive result, we allow bidders to win multiple bids.

Our problem is a special case of finding a maximum-weight independent set in a geometric intersection graph. In such a graph, there is a node for each bid (in our case: a (connected) set of rectangles), and two nodes are connected if and only if the corresponding bids overlap. Finding a maximum-weight independent set in a geometric intersection graph is a well-studied problem for several types of intersection graphs. For instance, in the work of [9], it is shown that if all items are arranged in a single row, and bids are only allowed for subsets of consecutive items, the resulting winner determination problem is polynomially solvable. These results follow from the equivalence of this problem to finding a maximum-weight independent set in an interval graph. For an overview on results for more general intersection graphs we refer to [3].

In this paper, we study the computational complexity of the winner determination problem for the specific geometric

setting described above. We show that it can be used to efficiently solve the winner determination problem (which is hard in general), using dynamic programming procedures. Additionally, we settle the complexity of the winner determination problem for bidding in a grid.

2. PROBLEM DESCRIPTION

The geometric setting that we consider can be described as follows. Given are k rows. Each row contains an (ordered) set of items (or rectangles). If, on some row, an item u lies to the left of item v , then we write $u \prec v$. We use $X_j = \{0, 1, \dots, m_j\}$ to denote the set of items in row j , $j = 1, \dots, k$. The set of items that can be bid on is $\bigcup_{j=1}^k X_j \setminus \{0\}$; item 0 cannot be part of any bid, and is only present for notational convenience. We assume that item ℓ lies directly to the left of item $\ell + 1$, for each $\ell \in X_j \setminus \{m_j\}$, $j = 1, \dots, k$.

Definition 1. We say that a pair of items are adjacent if and only if they share a border with non-zero length.

Clearly, items ℓ and $\ell + 1$ are adjacent. However, items on different (but consecutive) rows can be adjacent as well. We use m to denote the number of items in the instance, i.e., $m = \sum_{j=1}^k m_j$. Figure 3 visualizes this.

Row 1	1	2				m_1
Row 2	1	2				m_2
Row 3	1	2				m_3

Figure 3: An example of an instance with $k = 3$ (i.e. 3 rows) and $m_1 = 6$, $m_2 = 8$, $m_3 = 7$.

We investigate the following problem, called the winner determination problem (WDP). Given is a set of bids \mathcal{B} on subsets of items, with $v(b)$ denoting the value of bid b , for each $b \in \mathcal{B}$. We set $n = |\mathcal{B}|$, i.e. there are n bids; specifying a bid implies specifying a set of items, as well as a value $v(b) > 0$. The problem is to find an allocation that maximizes the sum of the values of the accepted bids, ensuring that each item is allocated at most once.

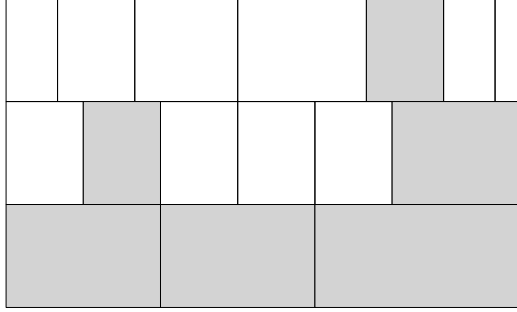
Given a bid b , consider the *item graph*, $H(b)$, which has a node for each item in bid b , and there is an edge between a pair of nodes in $H(b)$ if and only if the corresponding items are adjacent. There are two main restrictions on the bids that we consider. We define the concept of a *connected* bid.

Definition 2. We say that bid b is *connected* if the subgraph $H(b)$ induced by the items of bid b is connected. If bid b is not connected, we say that it is *disconnected*.

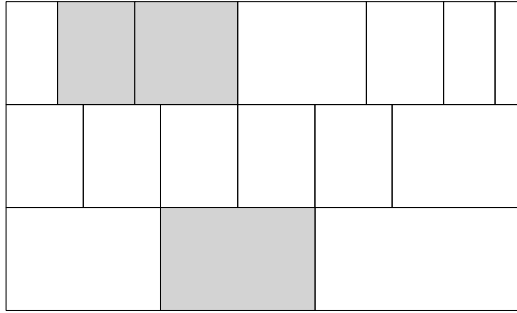
Further, let us define the concept of a bid that is *gap-free*. A formal definition of a bid having no gaps (i.e. being gap-free) is formulated as follows.

Definition 3. We say that bid b is *gap-free* if no three items $u \prec v \prec w$ on a single row exist for which $u \in b, v \notin b, w \in b$.

A bid that is not gap-free has at least one gap. Notice that it is easy to exhibit examples of connected bids that are not gap-free (see Figure 4a), and gap-free bids that are not connected (see Figure 4b). It is also easy to see that in the case of a single row, i.e. $k = 1$, connectedness of a bid is equivalent to a bid being gap-free.



(a) A bid that is connected and not gap-free.



(b) A bid that is disconnected and gap-free.

Figure 4: Examples illustrating the concepts of a connectedness and gap-freeness.

Finally, it is important to see that bids on identical sets of items but with different values need not all be considered. Indeed, one need only consider the bid with the highest value. If more than one bid has the highest value, one could use the bid entry time as a tie-breaker. Thus, all but the highest value bid on a specific set of items can be eliminated and bids will be unique in the sense that they are all for different sets of items.

3. RESULTS

For the setting where items are arranged in rows, we show the following:

- For connected and gap-free bids, the winner determination problem is easy when the number of rows is fixed. We solve this problem using a polynomial time dynamic programming algorithm.
- For the setting where the bid space is a grid and both the number of rows and columns are a part of the input, we show that even when bids are constrained to

be row bids or column bids, the resulting winner determination problem is NP-hard.

- For gap-free bids, the winner determination problem is NP-hard, even on two rows.
- For connected bids, the winner determination problem is easy on three rows or fewer. We show this by adapting and expanding upon the general dynamic programming algorithm developed for connected and gap-free bids.

We point out that the complexity of the winner determination problem with connected bids on a fixed number of rows k , with $k \geq 4$, is still an open problem. If the number of rows is part of the input, a result in [9] implies the problem is NP-hard.

Due to the page limitation imposed on this manuscript, the following section only describes our dynamic program for winner determination for the case of k rows, with connected and gap-free bids. For the proofs of our other claims, we refer the reader to our working paper [12].

4. A DYNAMIC PROGRAM FOR WINNER DETERMINATION FOR CONNECTED AND GAP-FREE BIDS

In this section we assume that bids are connected and gap-free. We show how the winner determination problem for a setting with k rows can be solved as a shortest path problem on a graph $G = (V, A)$, which is constructed as follows. There is a node in V for each element in the Cartesian product of the sets X_1, X_2, \dots, X_k . We write $V = \prod_{i=1}^k X_i$. Nodes in V are k -tuples. We consider the k -tuple $\mathbf{x} = \langle x_1, x_2, \dots, x_k \rangle$, where $x_1 \in X_1, x_2 \in X_2, \dots$ and $x_k \in X_k$. This k -tuple represents a state, i.e. a collection of assigned items. More specifically, the k -tuple \mathbf{x} represents a state where irrevocable decisions concerning the items $\{0, \dots, x_1\} \cup \{0, \dots, x_2\} \cup \dots \cup \{0, \dots, x_k\}$ have been made, i.e. for each row i all items from left to right up to and including x_i . As there is a node in V for every k -tuple, this leads to $O(m^k)$ nodes.

The arc set A includes two types of arcs: the zero arcs and bid arcs. The zero arcs have a weight of 0, and are used to handle items not included in the set of winning bids. Consider some node $\mathbf{x} = \langle x_1, x_2, \dots, x_i, \dots, x_k \rangle \in V$, with $1 \leq i \leq k$ and $x_i \neq m_i$. A zero arc goes from node \mathbf{x} to node $\langle x_1, \dots, x_i + 1, \dots, x_k \rangle \in V$, for each $1 \leq i \leq k$. Thus, up to k zero arcs emanate node $\mathbf{x} \in V$, giving rise to $O(m^k)$ zero arcs in the graph G .

The bid arcs correspond to actual bids and have a weight equal to the value of the bid $v(b)$. We represent a bid by listing k pairs of elements; each pair represents the first element, and the last element present in a bid on a particular row. For a bid b that contains elements from each of the k rows, we write: $b = \{(x_1^b, y_1^b), (x_2^b, y_2^b), \dots, (x_k^b, y_k^b)\}$, where the element $x_j^b \in X_j$ ($1 \leq j \leq k$) refers to the leftmost element of X_j present in bid b , and the element $y_j^b \in X_j$ ($1 \leq j \leq k$) refers to the rightmost element of X_j present in bid b . We use the symbol (\emptyset, \emptyset) to denote that a bid

does not include items from that row. Thus, as an example, when we write $b = \{(\emptyset, \emptyset), (x_2^b, y_2^b), (x_3^b, y_3^b), (\emptyset, \emptyset)\}$ this means that the bid b does not include any items on the first row, it includes items x_2 up to and including y_2 on the second row, it includes items x_3 up to and including y_3 on the third row, and it does not include any items on the fourth row.

The bid arcs can be described as follows. Let us, for convenience, first assume that bid b contains elements from each of the k rows. To represent bid b in the graph G , we draw an arc from node $\langle x_1^b - 1, x_2^b - 1, \dots, x_k^b - 1 \rangle$ to node $\langle y_1^b, y_2^b, \dots, y_k^b \rangle$ with weight $v(b)$. Consider now a bid b such that there are rows with no elements in b . Observe that, due to connectedness of b , these rows can only have indices $1, 2, \dots, s(b)$ and $f(b), f(b) + 1, \dots, k$ with $0 \leq s(b) < f(b) \leq k + 1$. Note that if a bid b is present on the row 1 then $s(b) = 0$. Similarly, if a bid b is present on row k then $f(b) = k + 1$. Now, to represent bid b , for each $x_1 \in X_1, x_2 \in X_2, \dots, x_{s(b)} \in X_{s(b)}, x_{f(b)} \in X_{f(b)}, x_{f(b)+1} \in X_{f(b)+1}, \dots, x_k \in X_k$ there is an arc from node $\langle x_1, x_2, \dots, x_{s(b)}, x_{s(b)+1}^b - 1, \dots, x_{f(b)-1}^b - 1, x_{f(b)}, \dots, x_k \rangle$ to node $\langle x_1, x_2, \dots, x_{s(b)}, y_{s(b)+1}^b, \dots, y_{f(b)-1}^b, x_{f(b)}, \dots, x_k \rangle$ with weight $v(b)$. Notice that there are $O(nm^{k-1})$ bid arcs (of course it is conceivable that the number of bid arcs will be far less).

We now compute a longest path from node $\mathbf{0} = \langle 0, \dots, 0 \rangle$ to node $\mathbf{m} = \langle m_1, \dots, m_k \rangle$. The length of this path corresponds to the optimal revenue of the auction, and the winning bids can be derived from the arcs in the path. Notice that $G = (V, A)$ is acyclic by construction and consists of $O(m^k)$ nodes and $O(m^{k-1}(n + m))$ arcs. Hence, a longest path can be found efficiently by solving a shortest path problem in $G = (V, A)$ with edge weights multiplied by -1 . Since Ahuja et al. [2] show that shortest path problems in directed acyclic graphs with p nodes and q arcs can be solved in $O(p + q)$ time, our dynamic program requires $O(m^k + nm^{k-1})$ time.

Once a longest path is found, it is easy to see which bids are accepted. Every arc that is not a zero arc in $G = (V, A)$ corresponds to exactly one bid. To find the set of winning bids, for every non-zero arc in the longest path simply accept the bid corresponding to that arc. For a numerical example and proof of correctness of our algorithm, we refer the reader to [12].

5. CONCLUSIONS

We study the winner determination problem for a combinatorial auction with a specific geometric structure. We argue that this structure is relevant, as it occurs in real estate, plots of land, and mineral rights. The complementarities present in these situations offer great potential for combinatorial auctions.

With our dynamic programming algorithm, we present auctioneers a tool that enables them, under some reasonable assumptions on the bids and with a fixed number of rows, to efficiently compute the winning bids. Next, we complement existing results by showing that bidding in a grid is difficult, even when only row and column bids are allowed,

if the number of rows is part of the input. Our results may also prove useful for experimental research on combinatorial auctions: our dynamic program will allow researchers to study bidder behavior in larger settings, involving more items and bidders than considered so far.

6. ACKNOWLEDGMENTS

This research is supported by the Interuniversity Attraction Poles Programme initiated by the Belgian Science Policy Office. The authors wish to thank James B. Orlin for an interesting and stimulating conversation.

7. REFERENCES

- [1] J. Abrache, T. G. Crainic, M. Gendreau, and M. Rekik. Combinatorial auctions. *Annals of Operations Research*, 153(1):131–164, 2007.
- [2] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993.
- [3] T. M. Chan and S. Har-Peled. Approximation Algorithms for Maximum Independent Set of Pseudo-Disks. *Discrete Computational Geometry*, 48:373–392, 2012.
- [4] P. Cramton. How Best to Auction Oil Rights. In M. Humphreys, J. Sachs, and J. E. Stiglitz, editors, *Escaping the resource curse*, chapter 5, pages 114–151. Cambridge Univ Press, 2007.
- [5] P. Cramton, Y. Shoham, and R. Steinberg. *Combinatorial auctions*. MIT Press, 2006.
- [6] S. de Vries and R. V. Vohra. Combinatorial Auctions: A Survey. *INFORMS Journal on Computing*, 15(3):284–309, 2003.
- [7] D. R. Goossens, S. Onderstal, J. Pijnacker, and F. C. R. Spieksma. Solids: A Combinatorial Auction for Real Estate. *Interfaces*, 44(4):351–363, 2014.
- [8] D. Quan. Real Estate Auctions: A Survey of Theory and Practice. *Journal of Real Estate Finance and Economics*, 9(1):23–49, 1994.
- [9] M. H. Rothkopf, A. Pekeč, and R. M. Harstad. Computationally Manageable Combinatorial Auctions. *Management Science*, 44(8):1131–1147, 1998.
- [10] T. Sandholm. Algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence*, 135(1-2):1–54, 2002.
- [11] S. Van Hoesel and R. Müller. Optimization in electronic markets: examples in combinatorial auctions. *Netnomics*, 3(1):23–33, 2001.
- [12] B. Vangerven, D. Goossens, and F. Spieksma. Winner determination in geometrical combinatorial auctions. Technical report, KBI 1614, Faculty of Economics and Business, KU Leuven, 2016.

Diploid Genome Rearrangement

[Extended Abstract]

István Miklós*
Rényi Institute
Reáltanoda u. 13-15
1053 Budapest, Hungary
miklos.istvan@renyi.mta.hu

Adrienn Szabó
SZTAKI
Lágymányosi u. 11
1111 Budapest, Hungary
aszabo@ilab.sztaki.hu

ABSTRACT

Next Generation Sequencing (NGS) techniques revolutionized the collection of genomic data. It allows massively parallel sequencing of short fragments reducing the time and cost of sequencing. When pairs of fragments are sequenced, it is possible to detect rearrangement events using NGS, but in case of diploid genomes, rearrangement events might happen on both chromosomes of homologous pairs, and the entire rearranged genome cannot be directly read out from NGS data.

We consider the problem of reconstructing the rearranged diploid genome from NGS data, and study the computational complexity of the problem. We prove that finding one solution can be done in polynomial running time. On the other hand, deciding if there is a solution without non-homologous recombination between homologous chromosomes is NP-complete.

1. INTRODUCTION

The Next Generation Sequencing technique breaks the genome into small, overlapping pieces (several copies of the genomic DNA are broken) and these small pieces are sequenced. From these small, overlapping copies, the whole genome is reconstructed. The diploid genomes contain two copies of each chromosome (except the sex chromosomes) and in case of healthy genomes, the two chromosomes are identical.

However, cancer genomes might undergo a huge amount of genome rearrangement events, see for example [11]. In these genomes, the homologous chromosomes might be rearranged in different ways. It is possible to read out from the NGS data where rearrangement events happened in terms of chromosome positions, however, this data does not reveal in which copy of the homologous chromosomes a particular rearrangement happened. However, if the rearranged intervals

*Secondary affiliation: SZTAKI, 1111 Budapest, Lágymányosi u. 11, Hungary

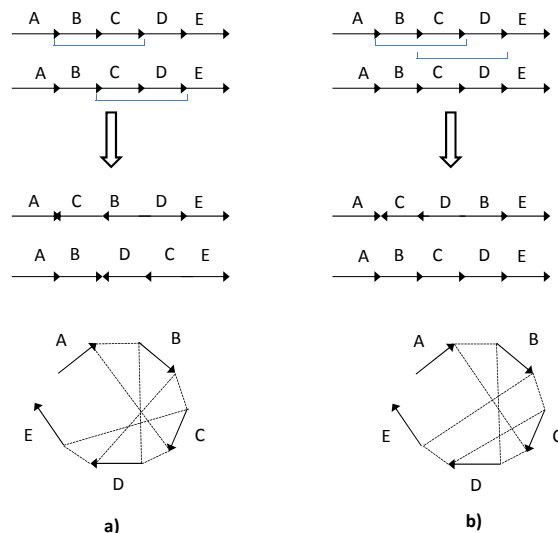


Figure 1: A pair of examples for diploid rearrangement and NGS graph. Both examples contain a pair of homologous chromosomes with 5 synteny blocks (unit segments), labelled by A, B, ... E. The grey edges of the NGS graphs are noted by dotted lines. The two copies of the black edges are replaced with one single edge due to sake of simplicity. See text for more details.

overlap, then it is decidable if the two rearrangement events happened on one or two copies of homologous chromosomes, see Fig. 1. On the left, segment BC is inverted in one of the chromosomes and the segment CD is inverted on the other chromosome. This latter inversion affects the adjacency of segments B and D and the adjacency of segments D and E. On the right, first the BC segment is inverted in one of the chromosomes, then another segment is inverted on the same chromosome also affecting the adjacency of segments B and C and the adjacency of segments D and E. The resulting rearranged genomes are different, and their NGS graphs are also different as shown at the bottom of the picture.

In this paper, we consider the problem of reconstructing the diploid genomes from NGS data. We show that without constraints, finding one solution is easy. On the other hand, the reconstruction problem is NP-complete if a biologically relevant restriction is introduced.

2. PRELIMINARIES

In this section, we transform the reconstruction problem into a graph theoretical problem.

DEFINITION 1. A diploid genome $\{G(V, E), L\}$ is an edge labelled directed graph in which each vertex has a total degree 1 or 2, each label in L is used exactly twice and the graph contains no cycles. The components of the diploid genome are called chromosomes. The edges are called synteny blocks. The beginning of an edge is called tail, and the end of the edge is called a head. The edges with the same labels are called homologous synteny blocks. The degree 1 vertices are called telomers, the degree 2 vertices are called adjacencies.

A synteny block is a DNA sequence that can be identified in a hereby not detailed biological way. Sometimes synteny blocks are called genes, however, a synteny block might be a large cluster of genes. A diploid genome contains two (almost) identical copies of each synteny block; this is why each label is used exactly twice in the graph representation. The differences in the two copies of the synteny blocks are point mutations that happen at less than one percentage of the nucleotides and causes the genetic variance of the individuals.

The NGS sequencing technique obtains short fragments from the genomes, typically at most one hundred of nucleotides. The typical length of a synteny block contains tenths of thousands or even more nucleotides. A run of few tens of nucleotides is typically unique in a genome, and thus, can identify a synteny block. Therefore the sequenced fragments are sufficiently long to identify which synteny blocks are neighbours (when a fragment covers the endings of two synteny blocks), however, it does not tell which copies of the two identical ones. Indeed, the rare point mutations do not provide sufficient information to distinguish the two copies of the synteny blocks. Furthermore, the sequenced fragments are not long enough to reveal the corresponding neighbours at the end of one copy of a synteny block. The information revealed from the NGS data can be summarized in the NGS graph, defined below.

DEFINITION 2. A NGS (Next Generation Sequencing) graph $\{G(V, E), L\}$ is an edge colored directed multigraph and labels with the following properties:

- The edges are coloured with black and gray. Black edges are directed and come in pairs, i.e. if there is a black edge from u to v , then there are exactly two black edges going from u to v .
- Each vertex has exactly 2 black edges and at most 2 gray edges. Loops are allowed only for gray edges, if a vertex has a gray loop, then it counts as 2 gray edges.
- Each couple of black vertices has a unique label coming from the label set L .

The couples of black vertices are called diploid synteny blocks. A diploid synteny block is a genomic segment that underwent

a rearrangement event in neither of the chromosomes. A vertex with no gray edge is the end of a pair of homologous synteny blocks that are telomers in two chromosomes, in an extreme case, it might be the two telomers of the same chromosome. A vertex with one gray edge is an end of a pair of synteny blocks whose one copy is a telomer, and whose another copy is in an adjacency with another synteny block. Finally, a vertex with two grey edges is the end of a pair of synteny blocks such that both copies are in adjacency with another synteny block ends.

Example genomes and NGS graphs can be seen on Fig. 1. For example, on Fig. 1 a), the two grey edges at the head of the synteny block A indicates the two synteny blocks adjacent to the head of synteny block A in the rearranged genome: head of C and tail of B.

DEFINITION 3. A diploid genome $\{G'(V', E'), L'\}$ is a realization of a NGS graph $\{G(V, E), L\}$ if $L = L'$ and there is a bijection between the grey edges in G and the degree 2 vertices in G' such that the grey edges connect the same endpoints of the diploid synteny blocks that are adjacent in G' .

Equivalently, a realization is a decomposition of the NGS graph into alternating walks, defined below. We also define the alternating circuits for technical reasons.

DEFINITION 4. An alternating walk on an NGS graph is a series of edges e_1, e_2, \dots, e_n such that all edges are different, for all $i = 1, 2, \dots, n - 1$, edges e_i and e_{i+1} have a common vertex, and the edges have alternating colourings in the series. Similarly, an alternating circuit is a series of edges e_1, e_2, \dots, e_{2n} such that all edges are different, for all $i = 1, 2, \dots, 2n - 1$, edges e_i and e_{i+1} have a common vertex, furthermore, e_{2n} and e_1 have a common vertex, and the edges have alternating colourings in the series.

3. FINDING ONE SOLUTION FOR THE DIPOLOID REARRANGEMENT PROBLEM

Here we consider two versions of the diploid rearrangement problem. The first version allows non-homologous recombinations between the same chromosomes. Such a rearrangement yields a chromosome that contains 2 copies of the same diploid synteny block. (In comparison, a homologous recombination swaps the almost identical synteny blocks between two chromosomes.) The second version does not allow such rearrangements, this happens for example, when the rearrangement events contain only reversals. We show that the first version is an easy problem while the other is an NP-complete one.

3.1 Diploid rearrangement allowing non-homologous recombinations between homologous chromosomes

The diploid rearrangement with non-homologous recombinations is the following problem: given a NGS graph, construct a diploid genome which is a realization of the NGS graph such that one chromosome might contain two copies of the same diploid synteny block.

THEOREM 1. Let $\{G(V, E), L\}$ be a NGS graph. It has at least one diploid genome realization iff each component of G contains at least one vertex with degree less than 4.

PROOF. If there is a component in G whose vertices all have degree 4 then it is impossible to map its grey edges onto linear components of a diploid genome. Indeed, since each vertex has 2 grey edges in the component, both copies of the diploid synteny blocks must be in adjacency with another synteny block, and thus, they cannot be telomers.

On the other hand, if there is a vertex with a degree less than 4 then there is a vertex with degree 2 or there are at least 2 vertices with degree 3, since the sum of degrees must be an even number. Starting with a vertex with less grey edges than black ones, take an alternating walk on the component, starting with a black edge, and ending with a vertex with no remaining edges with the alternating colour. Such walk ends in a vertex which has less grey edges than black ones, and thus, it ends with a black edge.

Once the walk is finished, remove this walk from the component. Either the walk covers the entire component, or there are remaining vertices. In this later case, removing the walk might create more than one components, take any of them. If there are remaining vertices in the component having less grey edges than black ones, start a new alternating walk in such a vertex, taking a black edge first, and finish it in another such vertex, remove this walk from the component, consider the remaining component, etc. After removing a few alternating walks, either the remaining component is empty or the remaining component contains only vertices having the same number of grey and black edges. There must exist a vertex with a degree less than 4, otherwise the removed alternating paths were vertex disjoint from the remaining component, thus disjoint from the component, a contradiction.

Choose any vertex with degree 2 from the remaining component, let it be denoted by v , and take an alternating circuit starting with v . Since this alternating circuit shares the vertex v with one of the removed walks, it can be merged with this alternating walk thus obtaining a larger walk. If the component is still not empty after removing the alternating circuit, keep processing it in the same way: take an alternating circuit having a vertex shared with one of the already removed walks, and merge the circuit and the path, thus obtaining a larger walk. Eventually, the component is decomposed into alternating walks.

Now, each alternating walk represents a chromosome by contracting the grey edges into a single vertex. Clearly, this set of chromosomes give a realization of the component. \square

It is also trivial that the decomposition of a component into alternating paths and thus into chromosomes can be done in polynomial time.

3.2 Diploid rearrangement excluding non-homologous recombinations between homologous chromosomes

The diploid rearrangement without non-homologous recombinations between homologous chromosomes ask if a real-

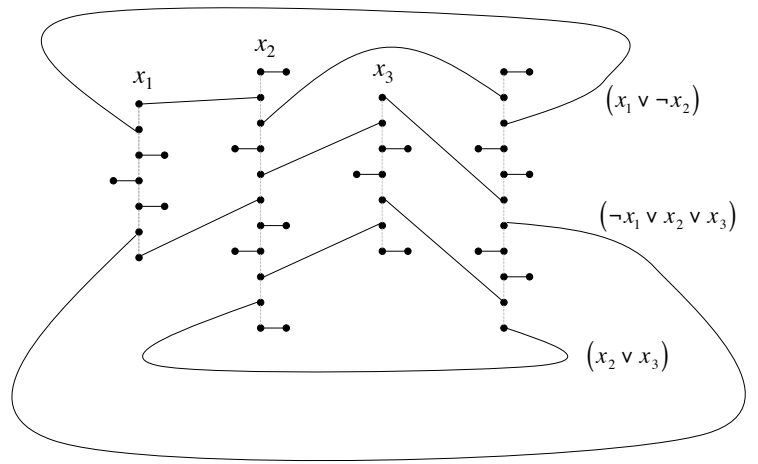


Figure 2: The NGS graph for the CNF $(x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (x_2 \vee x_3)$. Gray edges are indicated with dotted lines. Pairs of black edges are represented with a single black edge for sake of simplicity, and their directions are also omitted. The gray path for each boolean variable as well as the alternating grey-black cycle for each clause are labelled. See text for more details.

ization of a NGS graph $\{G(V, E), L\}$ exists in which the chromosomes can be split into two sets such that each set contains the entire label set L . Unfortunately, this question is hard to answer, as the following theorem states.

THEOREM 2. The diploid rearrangement problem without non-homologous recombinations between homologous chromosomes is NP-complete.

PROOF. It is trivial that any solution can be verified in polynomial time, thus the problem is clearly in NP. Below we prove that the problem is NP-hard by proving that SAT is polynomially reducible to it.

Clearly, a NGS graph has a realization without non-homologous recombination between homologous chromosomes if the gray edges can be coloured with two colours, say, blue and red such that the red edges and one copy from each pair of black vertices can be decomposed into a collection of alternating paths, furthermore, the blue edges and the other copies of black vertices can be decomposed into a collection of alternating paths. Each vertex of the NGS graph contains at most two gray edges, therefore the gray subgraph can be decomposed unequivocally into paths and cycles. It is obvious that in any solution to the problem, these paths and cycles are coloured alternating, hence there are two candidate colourings of each component. In the polynomial reduction, there will be a grey path for each boolean variable, and the two possible colourings will correspond to the logical true and false assignments of the boolean variables.

Consider a conjunctive normal form with n boolean variables and k clauses. Construct a NGS graph for the diploid rearrangement problem in the following way (see also Fig. 2): Make $n + 1$ chains of grey edges, for $j = 1, 2, \dots, n$, the j th

chain contains $4m_j - 1$ vertices, where m_j is the number of clauses in which the j th boolean variable participates (either negated or not negated). Number the vertices in each chain starting with 1, and in each of these chains, the vertices with indices $4i - 2$ accommodate the incoming pair of black edges for the i th clause having the boolean variable in it. The vertices with indexes $4i$ will have a "separator" pair of black edges, whose other vertex is a "dead end", namely, has a degree 2. The vertices with indexes $4i - 1$ and $4i - 3$ are connected to the outgoing pair of black edges of the i th clause. If the logical true value of the j th boolean variable satisfies the i th clause, then the black edges going out from the $4i - 1$ st vertex have a dead end, and the black edges going out from the $4i - 3$ rd vertex will be the incoming edge for the next grey chain. Otherwise, the edges going out from the $4i - 3$ rd vertex will have a dead end, and the other pair will be the incoming pair of the next gray chain.

The last gray chain contains $4k - 1$ vertices, the vertices with indexes $4i - 2$ will have the incoming pair of black edges for the i th clause. The vertices with indexes $4i$ will have a "separator" pair of black edges, whose other vertex is a "dead end", namely, has a degree 2. The vertices with indexes $4i - 1$ and $4i - 3$ are connected to the outgoing pair of black edges of the i th clause. The black edges going out from the $4i - 1$ st vertex are the incoming edges in the first gray chain for the i th clause, and the pair of vertices going out from the $4i - 3$ rd vertex have a dead end. In this way, for each clause, we create a cycle, containing alternately pairs of black edges and grey edges. The gray edges indicate the logical assignments of the boolean variables providing that the clause is not satisfied (and there is an additional grey edge in the last grey chain). If these grey edges have the same colour, then such colouring cannot provide a solution to the problem, since it contains a cycle.

We claim that the diploid rearrangement is solvable for the so-constructed graph if and only if the CNF is satisfiable. If the CNF is satisfiable, then there is a colouring of the gray edges such that the red edges and one copy of the black edges have at least one dead end for each clause, so the red-black subgraph contains only paths. The last chain can be coloured such that the blue edges and the other copies of black edges will have dead ends in this chain, so the blue-black subgraph contains only paths, and thus, we have a solution for the diploid rearrangement problem.

On the other hand, if the CNF is not satisfiable, then for any colouring of the first n gray chains, at least one of the clauses does not have a dead end for the blue-black subgraph in the first n gray chains and also at least one of them does not have a dead end for the red-black subgraph. Whatever is the colouring of the $n + 1$ st gray chain, one of the colourings will create a circular chromosome, hence there is no solution for the diploid rearrangement problem. \square

4. DISCUSSION AND CONSLUSIONS

In this paper, we considered the diploid rearrangement problem and showed that it is polynomial solvable when there is no restriction and NP-complete if the solution space is restricted.

There might be more than one solution to the problem, and

therefore it is also an interesting question what can we say about the computational complexity of counting or sampling these solutions. These types of questions have impact in bioinformatics, and were considered and partially answered for other genome rearrangement problems, see [1, 2, 3, 5, 6, 7, 8, 9, 10].

The diploid rearrangement problem is slightly similar to finding Eulerian circuits in an Eulerian graph, in fact, if a component of an NGS graph contains one vertex with degree 2 and all other vertices have degree 4 then each solution is a closed Eulerian walk that in fact is an Eulerian cycle. It is known that counting the Eulerian cycles in an undirected graph is #P-complete, and it is an open question if there are efficient algorithms for approximating the number of solutions and sampling almost uniformly the solutions [4]. Therefore it is natural to conjecture that counting and sampling the solutions for the diploid genome rearrangement problem is also #P-complete.

5. REFERENCES

- [1] Ajana, Y., Lefebvre, J.F., Tillier, E.R.M., El-Mabrouk, N. 2002. Exploring the set of all minimal sequences of reversals - an application to test the replication-directed reversal hypothesis. In: WABI '02: Proceedings of the Second International Workshop on Algorithms in Bioinformatics, London, UK, Springer-Verlag 300-315
- [2] Braga, MDV, Sagot, M-F, Scornavacca, C, Tannier, E (2008) Exploring the Solution Space of Sorting by Reversals with Experiments and an Application to Evolution, *IEEE-ACM Transactions on Computational Biology and Bioinformatics*, **5**, 348-356
- [3] Braga, M.D.V., Stoye, J. (2009) Counting All DCJ Sorting Scenarios. *LNCS*, **5817**, 36-47.
- [4] Brightwell, G.R., Winkler, P.: Note on Counting Eulerian Circuits, <http://arxiv.org/abs/cs/0405067> (2004)
- [5] Miklós, I., Darling, A. (2009) Efficient sampling of parsimonious inversion histories with application to genome rearrangement in *Yersinia* Genome Biology and Evolution, **1(1)**:153-164.
- [6] Istvan Miklos, Sandor Z. Kiss, Eric Tannier (2013) On sampling SCJ rearrangement scenarios, <http://arxiv.org/abs/1304.2170>
- [7] Miklós, I., Mélykúti, B., Swenson, K.: The Metropolized Partial Importance Sampling MCMC mixes slowly on minimum reversal rearrangement paths *ACM/IEEE Transactions on Computational Biology and Bioinformatics*, vol. 4, no. 7, 763-767, 2010.
- [8] Miklós, I., Smith, H.: Sampling and counting genome rearrangement scenarios, *BMC Bioinformatics*, **16(Suppl 14)**: S6. (2015)
- [9] Miklós, I., Tannier, E. (2012) Approximating the number of Double Cut-and-Join scenarios, *Theoretical Computer Science* 439:30-40.
- [10] Ouangraoua, A., Bergeron, A. (2009) Parking Functions, Labeled Trees and DCJ Sorting Scenarios. *LNCS*, **5817**, 24-35.
- [11] Stratton, B.R., Campbell, P.J., Futreal, P.A.: The cancer genome. *Nature* 458, 719-724 (2009).

Team Work Scheduling

Gyorgy Dosa
Department of Mathematics
University of Pannonia,
Veszprém, Hungary
dosagy@almos.vein.hu

Hans Kellerer
Institut für Statistik und
Operations Research,
Universität Graz, Austria
hans.kellerer@uni-
graz.at

Zsolt Tuza^{*}
Alfréd Rényi Institute of
Mathematics, Hungarian
Academy of Sciences
tuza@dcs.uni-pannon.hu

ABSTRACT

We introduce a quite general scheduling model we call Team Work Scheduling. It mainly means that a team works together to process any job. Its special version is recently defined as MultiProfessor scheduling, and even a more special version is the RAR problem. This last one means that parallel machine scheduling is considered with job assignment restrictions, i.e., each job can only be processed on a certain subset of the machines. Moreover, each job requires a set of renewable resources. Any resource can be used by only one job at any time. The objective is to minimize the makespan. We present approximation algorithms with constant worst-case bound in the case that each job requires only a fixed number of resources. For some special cases optimal algorithms with polynomial running time are given. On the other hand we prove that the problem is APX-hard, even when there are just three machines and the input is restricted to unit-time jobs.

Keywords

multiprocessor scheduling, approximation algorithm

1. INTRODUCTION

We define a general problem we call *Team Work Scheduling*, TWS for short. In this model given jobs (as usual in the area of scheduling), but now each job is executed simultaneously by certain machines, i.e. a *team*. We minimize the makespan. Now we give the exact definition of the new model as below.

Given m machines, and n jobs, the set of machines is denoted by M , and the set of jobs is denoted by N . Moreover for any job $j \in N$,

- given an integer $1 \leq t_j \leq m$, this parameter means

^{*}another affiliation is: Department of Computer Science and Systems Technology, University of Pannonia, Veszprém, Hungary

that the *team* that processes job j consists of t_j different *types* of collaborators/machines,

- also given a *collection* of t_j sets $C_j = (M_j^1, M_j^2, \dots, M_j^{t_j})$, these sets are pairwise disjoint and $M_j^k \subseteq M$ for any $k \in \{1, \dots, t_j\}$,
- furthermore given a collection of t_j integer *numbers* $n_j^1, n_j^2, \dots, n_j^{t_j}$, such that $n_j^k \leq |M_j^k|$. Then n_j^k means the required number of machines from set M_j^k . (That is, from the k -th type of machines specified for the job, there are M_j^k possible machines, and from these machines "only" n_j^k machines will be chosen.) Let $n_j = \sum_{k=1}^{t_j} n_j^k$, this integer means that altogether exactly n_j machines will be chosen to process job j , from all types.

When we schedule the jobs, for any job j , the n_j^k required number of machines must be chosen from set of machines M_j^k , for any k . In case $n_j^k < |M_j^k|$ these machines are *elective*, we can freely choose any n_j^k machines from the $|M_j^k|$ machines. Otherwise, if $n_j^k = |M_j^k|$ these machines are *mandatory*, all of them are needed for the execution of the job. The chosen machines are denoted by T_j and called the *team* (chosen for job j).

Finally, given the processing time $p_{i,j}$ for any (i, j) pair ($i \in M, j \in N$), this is the time needed to execute job j by machine i . Each job will be executed by the team chosen for the job, so, all these intended machines will run in parallel. For any job j , we choose the team, and we take the maximum of the $p_{i,j}$ processing times for the chosen machines (i.e. the team). This is the processing time of the job by the team, denoted by q_j . Naturally, q_j is not given in advance, it depends on the choice of the team to execute job j . Any machine can process at most one job at any time, and if a team is chosen for a job, no matter if some machine's processing time is smaller and another machine's processing time is larger in the team, all machines of the team are considered busy during the longest $p_{i,j}$ processing time for $i \in T_j$.

We ask for the minimum time (i.e. makespan) until all jobs are executed by the machines. We are interested in both the offline and online case.

2. APPLICATIONS

A typical *online* model is the following one. Accidents happen in an unpredictable way in a city, and the injured people are taken into a hospital to perform the necessary operations for them. These operations are the jobs. For any operation, according to the nature of the injury, a special team is needed, the members of the team play the role of the machines. Let us consider one operation. It is possible that the presence of some doctors is indispensable. For example only one doctor can make the anesthesia, so he/she will surely be there during the operation. Also, there is an expert, the only one who can make a special kind of operation. So both of them will be there, they play the role of some mandatory machines. Moreover there are also several nurses who are free at that time, and either of them can be chosen as the one who helps the doctors. For example from five such persons three must be selected, they play the role of elective machines. Suppose there are two operating rooms that are available at moment, one of them must be chosen, this is also an elective machine in our model. Naturally, the injured person plays the role also of a mandatory machine. The duration of the operation may depend on the chosen persons (as a proficient worker makes some activity faster than a beginner).

For another (*offline*) application let us consider a fast food restaurant, where some kinds of salads are made (among other foods). The machines are of different types.

a, Members of the staff (called makers) who make the salad.

b, Machines for mixing, heating, and other preparing operations.

c, Ingredients. For example mustard is stored in some bottle, and the whole bottle is reserved for some salad-maker during he makes the salad, but not all content will be used, only some portion. So the battle of the mustard is a (mobil) machine.

Then all machines (i.e. the member who makes the salad, the devices that are needed, and all ingredients) are collected together, and by use of them the salad will be made ready. We want to make ready all ordered foods as soon as possible.

3. RELATED MODELS

Multiprofessor Scheduling (MPS for short). The MPS problem is characterized by the following settings: For any job j , $1 \leq t_j \leq 2$. The team that processes job j consists of at most 2 different types of collaborators, if $t_j = 2$, then one type is mandatory, another is elective. The set of mandatory machines contains several machines, also the set of elective machines, but exactly one machine must be chosen from the elective machines. The problem is defined and considered in [1]. It is evident, that MPS is a special case of the TWS problem.

To explain better the MPS model, in this model we have a set $\mathcal{P} = \{P_1, \dots, P_u\}$ of professors and a set $\mathcal{L} = \{L_1, \dots, L_n\}$ of lectures with two sets \mathcal{C} and \mathcal{C}^* of conditions given by

pairs: $(P_i, L_j) \in \mathcal{C}$ means that professor P_i can deliver lecture L_j if it is assigned to him, while $(P_s, L_t)^* \in \mathcal{C}^*$ means that professor P_s has to be present when L_t is delivered by some *other* professor, who is assigned to this lecture. The MPS problem is still quite general, in [1] many (other) applications are also given. For example, MPS is still generalization of the *Restricted assignment* (RA for short) problem or the *Hierarchical scheduling problem* (HS for short).

Restricted Assignment with Resources Problem (RAR for short, [3]). Finally we define an even more special case of the TWS model, which is a special case of the MPS model. We are given n independent jobs $1, \dots, n$ that are to be scheduled on m' parallel machines $M_1, \dots, M_{m'}$. In the *restricted assignment problem* (RA, for short, [2]) each job j can be executed on a specific subset $\mathcal{M}(j)$ of the machines, and on those machines the processing time of job j is p_j . The objective is to minimize the makespan. In the three field notation, we abbreviate this problem by $R|p_{ij} \in \{p_j, \infty\}|C_{\max}$. Assume that additionally there are μ renewable resources R_1, \dots, R_μ (then $m' + \mu = m$). Let Λ_k be the set of jobs which require resource R_k , and let λ_k denote the cardinality of set Λ_k , $k = 1, \dots, \mu$. Job j requires simultaneous availability of all resources in the set $\mathcal{R}(j) \subseteq \{R_1, \dots, R_\mu\}$ for processing; we denote by ρ_j the cardinality of $\mathcal{R}(j)$, $j = 1, \dots, n$. Any resource can be used by only one job at any time. It means that two jobs which require the same resource cannot be processed simultaneously. We abbreviate this problem by $R|p_{ij} \in \{p_j, \infty\}, res_\mu|C_{\max}$. The *degree* of the problem is defined as the quantity $B = \max_{j=1, \dots, n} \rho_j$, that is the maximum number of resources required by a job.

4. RESULTS

Professors of the MPS model correspond to machines of the RAR model; the lectures are the jobs, the duration of a lecture means the processing time. But RAR is only a particular case of MPS: distinction between machines and resources means a partition of professors into two classes: those only delivering lectures ('Professors'), and the others only attending ('Instructors'). This special case of MPS is termed the *PI model*.

Among several results, it is proved in [1] that PI with unit-time lectures is NP -hard to $O(n^{1-\epsilon})$ -approximate for any fixed $\epsilon > 0$ if there are n professors and $O(n^2)$ instructors, even in the more restricted PI model where \mathcal{C} establishes a bijection between lectures and professors and when it is assumed further that any instructor is involved in just two conditions of \mathcal{C}^* . On the other hand, still considering unit times, if the number of professors, or the number of lectures is fixed, then MPS can be solved in linear time.

For the RAR model, we can prove inapproximability results and design approximation algorithms. Our main negative result is that the problem with unit-time jobs is APX -hard, already on three machines. In the case that each job requires only a bounded number of resources, we design approximation algorithms with constant worst-case bound, without any restrictions on processing times. For some special cases (e.g., unit-time jobs with degree $B = 1$) we design optimal algorithms with polynomial running time. To derive the main negative result, we prove a theorem on graph coloring,

which seems to be of interest on its own right, too. It states *APX*-hardness of the chromatic number on a restricted class of graphs.

5. ACKNOWLEDGMENTS

The first author is partially supported by the project VKSZ_12-1-2013-0088 Development of cloud based smart IT solutions by IBM Hungary in cooperation with the University of Pannonia. All three authors are partially supported by Stiftung Aktion Österreich-Ungarn, under grant 92öu1. The first and the last author are supported in part by the National Research, Development and Innovation Office – NKFIH under the grant SNN 116095.

6. REFERENCES

- [1] G. Dosa, Zs. Tuza, Multiprofessor Scheduling, online first, DAM, 2016, <http://dx.doi.org/10.1016/j.dam.2016.01.035>.
- [2] C. A. Glass and H. Kellerer, Parallel machine scheduling with job assignment restrictions, Naval Research Logistics, 54 (3), 250–257, 2007.
- [3] G. Dosa, H. Kellerer, Zs. Tuza, Restricted Assignment Scheduling with Resource Constraints, manuscript, 2016.

Incremental 2-D nearest-point search with evenly populated strips

David Podgorelec University of Maribor
Faculty of Electrical Engineering and Computer
Science
Maribor, Slovenia
david.podgorelec@um.si

Denis Špelič University of Maribor
Faculty of Electrical Engineering and Computer
Science
Maribor, Slovenia
denis.spelic@um.si

ABSTRACT

The incremental nearest-point search successively inserts query points into the space partition data structure, and the nearest-point for each of them is simultaneously found among the previously inserted points. The paper introduces a new approach to solve this problem in 2D-space. Dynamic partitioning successfully prevents situations with over-populated strips but still fails to reach optimality. A variant with two perpendicular partitions and four types of deterministic skip lists is therefore discussed as a possible extension.

Categories and Subject Descriptors

E.1 [Data Structures]: Lists, Stacks and Queues, Trees;
F.2.2 [Analysis of Algorithms and Problem Complexity]: Nonnumerical Algorithms and Problems—*geometrical problems and computations, sorting and searching*

General Terms

Algorithms, Performance, Theory

Keywords

Incremental nearest-point, dynamic partition, deterministic skip list

1. INTRODUCTION

The nearest-point search means a search for the target point $p_i \in S = \{p_1, \dots, p_n\}$, such that the distance between p_i and a given query point p is minimal. It enables or at least facilitates solving numerous practical problems from various research and application areas, such as computational geometry [12], GIS [6], motion planning [9], and computer graphics [1]. Note that the distance need not refer to pure geometric relation between two spatial points (e.g. Euclidean distance). This generalization extends the usability of the nearest-point search to database querying in the most versatile applications.

If the distance is computable in $\theta(1)$ time, the nearest-point

search is trivially handled in $\theta(n)$ time, but the problem becomes more demanding when a recurring nearest-point problem has to be solved. A straightforward repetition of the basic nearest-point search results in $\theta(n^2)$ time when applied to $\theta(n)$ query points. More advanced approaches use space partitioning to bound the number of possible nearest-point candidates in each iteration [11]. The partition is accomplished by constructing a hierarchical or a grid data structure, typically a tree [4], the Voronoi diagram [7], a regular grid [2], or a multi-level organization of these structures [11]. Such a data structure is aimed to accelerate solving the point-location problem i.e. determination of the region where a query point lies. A static partition does not adapt itself to the point distribution. On the contrary, a dynamic partition maintains the numbers of points in all cells within previously determined limits. Particularly in higher dimensions, where either query time or storage space must be sacrificed, a user may also be satisfied by approximate solutions provided by the reasonably fast locality sensitive hashing technique [10].

In this paper, we introduce an original dynamic plane partition into parallel strips and utilize it to handle the so-called incremental nearest-point search in 2-D space. This represents a special case of the recurring nearest-point search where: (1) the set of target points S and the set of query points coincide, and (2) the points p_1, \dots, p_n are successively inserted into the data structure and their nearest-points are simultaneously found. The incremental search adequately models interactive processing of database queries where the results of previous queries are usually irrelevant for processing the current one. In computational geometry, a remarkably fast incremental Delaunay triangulation algorithm is based on the incremental nearest-point search [12].

2. DP-DSL APPROACH TO INCREMENTAL NEAREST-POINT SEARCH

The Voronoi diagram enables optimal $O(n \log n)$ time in the preliminary points arrangement approach, but the incremental nearest-point search requires some of the incremental Voronoi diagram construction algorithms which all, although fast on average, require quadratic time in the worst case [5]. For this reason and because of a relatively complex maintenance of the Voronoi diagrams, we preferably study other space partitioning techniques. First of all, we wish to keep practical advantages of the HT-DSL approach [11] and, simultaneously, to improve its theoretical behaviour. The pioneering HT-DSL approach represents even nowadays the only work where the incremental nearest-point search is explicitly

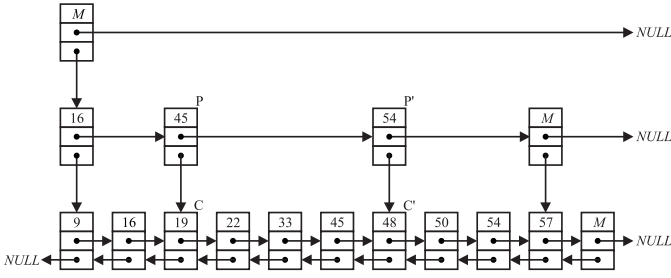


Figure 1: (1, 3)-deterministic skip list.

considered. It is based on a uniform plane subdivision into parallel strips. These static strips are directly accessible in $O(1)$ time through a hash table (HT). On the other hand, our DP-DSL approach uses a dynamic partition (DP) into evenly populated strips. In both methods, the points in a particular strip are stored in (a, b) -deterministic skip list (DSL) [8], providing a point insertion in $O(\log n)$ time and, on the average, efficient nearest-point search inside the strip. The DP-DSL approach must additionally provide the functionality of DSL splitting when an over-populated strip is split into two (or three) strips.

2.1 Deterministic skip lists

Our implementation of (a, b) -DSL, inherited from [11], consists of a doubly linked list of points sorted regarding the x-coordinate. Double connectivity assures that the move from an arbitrary point to its direct predecessor or successor takes $O(1)$ time. This list represents the basic level (level 1) of the DSL. Its nodes (leaves) are accessible from simply linked lists of the internal nodes at higher levels. Each parent node (for example P in Fig. 1) at level $h, h > 1$, points to a single child node (C) at level $h - 1$. Nodes at level $h - 1$, between the child nodes (C and C') of two successive parent nodes (P and P') form a gap. The gap size must be in range $[a, b]$. Values stored in a gap are lower or equal to the value in the parent node. Consequently, value M must be set to some "safely" high value.

To access a particular leaf or to insert a new one, at most one child node and the successive gap must be examined at each of the $\lceil \log_b(n + 1) \rceil$ levels, resulting in $O(b \log n)$ worst time. By keeping b small, the logarithmic access time is provided. Typical pairs (a, b) in practice are (1, 2), (1, 3), (2, 5), and (3, 7). Fig. 1 shows a (1,3)-DSL.

The actual search for the nearest point to the query point p was also inherited from [11]. It consists of the local search in the strip where p was inserted, and the *inter-cluster* search which progresses up and/or down through the adjacent strips.

2.2 Dynamic partition

The HT-DSL is remarkably fast for nearly uniform point distributions. However, examples with much slower performance and also strongly affected by the points ordering can effortlessly be constructed and, not rarely, also met in practice. Example in Fig. 2a consists of a few over-populated strips and, on the other hand, of a large majority of strips containing only few points. The DP-DSL approach is dir-

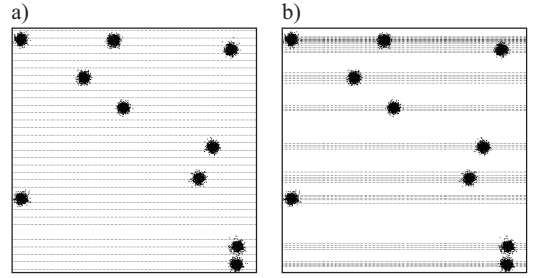


Figure 2: a) HT-DSL and b) DP-DSL approach employed on clusters of points.

ectly designed to prevent from such situations. The idea is straightforward: when a particular strip contains too many points, the algorithm splits it into a pair of strips, each containing half of the points of the original strip. Under certain conditions, splitting may also result in three strips. The DP-DSL approach in Fig. 2b cuts the clusters by many narrow strips, and leaves wide undivided strips between the clusters.

The DP-DSL approach requires additional data structure to store the strips' borders. We use additional DSL named *Borders* for this purpose. It plays the same role as the hash table in the HT-DSL approach, but requires longer search time (logarithmic instead of constant) and dynamic construction. Two types of strips are stored in *Borders*. A *line strip* is a horizontal line, and an *interval strip* is a region between two horizontal lines. The role of line strips is to keep sizes of the interval strips limited. A line strip is introduced when the y-coordinates of two or more points correspond to the splitting threshold.

Points in each DSL are sorted according to x-coordinates, but an over-populated strip should be split with regard to y . All the points in a line strip have the same y-coordinate and, therefore, splitting is only sensible for the interval strips. The splitting algorithm must firstly determine the splitting threshold. We utilize the well-known SELECT algorithm [3] which performs this task in linear time. The physical DSL splitting is realized by the original bricklaying approach. This firstly constructs level 1 for each of the two or three separate DSLs. This is achieved by moving the leaves of the input DSL, one after another, to the end of the corresponding separate list. Upper levels are then built from the elements of the so-called *global list of recyclable nodes*, consisting of the eventual unused nodes from previous splitting operations, the input DSL's internal nodes and, only if necessary, from newly allocated nodes. At each level, the algorithm groups the nodes into gaps of size $b - gsc$, where gsc is a user-selected gap size correction parameter.

3. RESULTS AND ANALYSIS

The number of strips in HT-DSL was experimentally determined in the range $m = \theta(\sqrt{n})$. Consequently, the number of points in a strip is $O(\sqrt{n})$ in an optimal case of the uniform point distribution. We have retained this result in the DP-DSL approach as well, and experimentally determined the best performance by splitting a strip when its size reaches $q = \lceil 3\sqrt{n} \rceil$. We use (1, 3)-DSLs in the HT-DSL approach, and (2, 5)-DSLs in the DP-DSL approach. We have

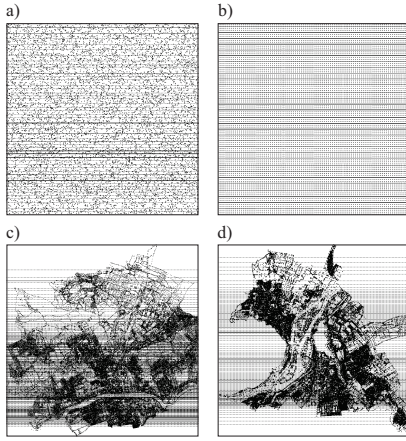


Figure 3: Dynamic partition into strips: a) uniform point distribution, b) grid, and c-d) GIS datasets.

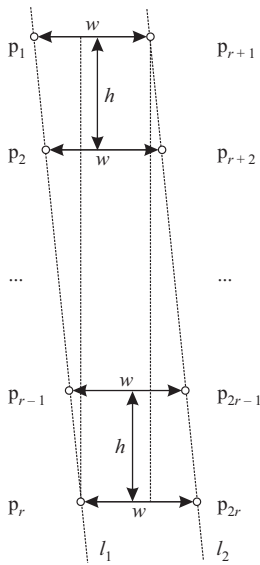


Figure 4: Point organization in the ladder example.

also determined the best long-term performance by using the gap size correction parameter $gsc = 1$.

In Table 1, the comparison between the HT-DSL and DP-DSL approach is given. The time ratios in the second column were obtained for configurations of 5.000.000 points. Figs. 2, 3 and 5 show the tested distributions with reduced numbers of points. The realistic examples in Figs. 3c-d consist of 70.334 and 193.360 points, respectively. Expected time complexities are listed in the last two columns. The examples from Figs 5a and 5c entitled the ladder, were synthetically generated and represent the worst-case for the local search. The construction is emphasized in Fig. 4. The condition $x_r - x_1 < w < h$ results in $\theta(r^2)$ time for a ladder with $2r = n - 1$ points in the same strip. Thus the HT-DSL approach spends $\theta(n^2)$ local search time for the example from Fig. 5c.

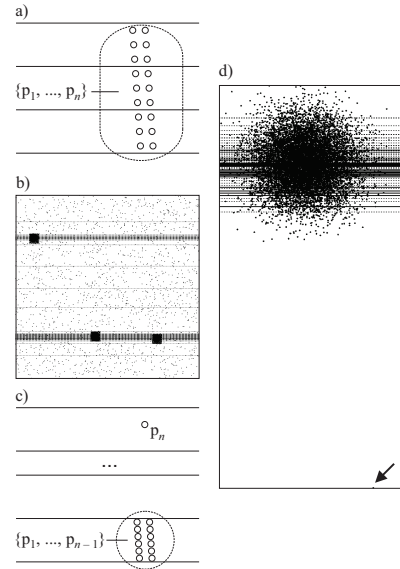


Figure 5: Testing examples of a) a ladder across all strips, b) clusters with uniform noise, c) a ladder in a single strip of the HT-DSL approach, and d) Gaussian distribution with additional point.

Table 1: Comparison of times between HT-DSL and DP-DSL approaches

Fig.	HT/DP	Time HT	Time DP
3a	0.79	$O(n \log n)$	$O(n \log n)$
3b	0.84	$O(n \log n)$	$O(n \log n)$
3c	0.50	$O(n \log n)$	$O(n \log n)$
3d	0.47	$O(n \log n)$	$O(n \log n)$
4b	1.46	$O(n\sqrt{n})$	$O(n \log n)$
2	7.21	$O(n\sqrt{n})$	$O(n \log n)$
4d	9.91	$O(n\sqrt{n})$	$O(n \log n)$
4a	0.59	$O(n\sqrt{n})$	$O(n\sqrt{n})$
4c	203.73	$\theta(n^2)$	$O(n\sqrt{n})$

3.1 Theoretical time complexity analysis

Table 2 gives expected worst-case time complexities of all phases of both approaches. The construction of strips and maintenance of DSLs are optimal in both cases, assuring the desired $O(n \log n)$ time. On the other hand, the local search and the inter-strip search time are both above this limit. Note that the local search time of the DP-DSL approach could be improved to $O(n \log n)$ by splitting the DSLs of size $q = O(\log n)$ instead of current $q = \lceil 3\sqrt{n} \rceil$. This change does not modify theoretical worst-case time complexities of other phases above the desired limits, but it usually results in slower practical performance due to the increased number of DSL splits and initial positioning operations in much more DSLs during the inter-strip search.

We have also managed to construct an example that requires $O(n^2)$ inter-strip search time in the DP-DSL approach. The construction is too extensive to find place in this paper. It is based on a geometric progression of x-coordinates with common ratio 2. Even for relatively low n , the exponential growth will quickly produce x-coordinates out of the range of

Table 2: Time complexities of particular phases in both approaches.

Phase	HT-DSLDP	DP-DSL
Strip identification	$O(n)$	$O(n \log n)$
Point insertion	$O(n \log n)$	$O(n \log n)$
DSL splitting	0	$O(n)$
Maintenance of Borders	0	$O(\sqrt{n} \log n)$
Local search	$O(n^2)$	$O(n\sqrt{n})$
Inter-strip search	$O(n^2)$	$O(n^2)$

Table 3: Comparison of three approaches in considered critical cases: I - ladder, II - rotated ladder, III - geometric progression, IV - rotated geometric progression, V - "regular".

Example	HT-DSLDP	DP-DSL	DP-4DSLs	Winner
I	$O(n^2)$	$O(n\sqrt{n})$	$O(n \log n)$	YH
II	$O(n^2)$	$O(n^2)$	$O(n \log n)$	XV
III	$O(n^2)$	$O(n^2)$	$O(n \log n)$	XV
IV	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$	YH
V	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$	various

the IEEE 754 floating-point specification, making this construction fully theoretical, as we do not expect such extreme values in industrial, GIS and other practical applications. However, a rotated ladder can, with some additional constraints, also represent the $O(n^2)$ inter-strip time example.

3.2 DP-4DSLs approach

We have recently developed an engineering solution which handles all the considered problematic examples in the desired (optimal) time bounds. Besides the horizontal DP, it additionally performs the vertical DP. In each strip, two orthogonal DSLs are constructed, the horizontal one sorted regarding the x-coordinate, and the vertical one sorted regarding the y-coordinate. Each point is therefore placed into four DSLs: XH-DSL (the one used in the DP-DSL approach) and YH-DSL are assigned to each horizontal strip, and XV-DSL and YV-DSL are constructed in each vertical strip. In each iteration of the local search, the method performs one move in each DSL which are all addressing the same radius r . The nearest-point of q is found when the first DSL (the winner) manages to examine all the points within the distance r around q . We have not managed to theoretically prove optimal time complexity but the performance in the considered problematic cases appears promising as seen in table 3. On the other hand, the HT-DSL and the DP-DSL outperform the DP-4DSLs approach in "regular" cases as maintenance of two partitions and four DSLs is quite expensive.

4. CONCLUSION

The paper considers a new (DP-DSL) approach to the incremental nearest-point search in 2-D. It guarantees $\theta(\sqrt{n})$ strips, each containing $O(\sqrt{n})$ points and, therefore, successfully prevents situations with over-populated strips and decreases the local search time from $O(n^2)$ to $O(n\sqrt{n})$. In our opinion, this is an important acceleration, although the algorithm still fails to achieve an optimal $O(n \log n)$ time performance characteristic for the preliminary points arrangement approach. In addition, examples can be constructed (although hardly met in practice) which, just as the "tradi-

tional" HT-DSL approach still achieve quadratic inter-strip search time. The DP-4DSLs variant seems to solve the considered problematic examples in optimal time, but a formal proof is still missing. Construction of the Voronoi diagram on $\theta(\sqrt{n})$ points and utilization of two perpendicular DSLs in each Voronoi cell could have a potential, but one should first prove that such dynamic partition is generally possible, and then provide an efficient region splitting algorithm.

5. REFERENCES

- [1] S. Bouaziz, A. Tagliasacchi, and M. Pauly. Sparse iterative closest point. In *Computer graphics forum*, volume 32, pages 113–123. Wiley Online Library, 2013.
- [2] J. G. Cleary. Analysis of an algorithm for finding nearest neighbors in euclidean space. *ACM Transactions on Mathematical Software (TOMS)*, 5(2):183–192, 1979.
- [3] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms*, volume 6. MIT press Cambridge, 2001.
- [4] E. Gómez-Ballester, L. Micó, and J. Oncina. Some approaches to improve tree-based nearest neighbour search algorithms. *Pattern Recognition*, 39(2):171–179, 2006.
- [5] L. J. Guibas, D. E. Knuth, and M. Sharir. Randomized incremental construction of delaunay and voronoi diagrams. *Algorithmica*, 7(1-6):381–413, 1992.
- [6] C. S. Jensen, J. Kolářvr, T. B. Pedersen, and I. Timko. Nearest neighbor queries in road networks. In *Proceedings of the 11th ACM international symposium on Advances in geographic information systems*, pages 1–8. ACM, 2003.
- [7] T. Kanda and K. Sugihara. Comparison of various trees for nearest-point search with/without the voronoi diagram. *Information processing letters*, 84(1):17–22, 2002.
- [8] J. I. Munro, T. Papadakis, and R. Sedgewick. Deterministic skip lists. In *Proceedings of the third annual ACM-SIAM symposium on Discrete algorithms*, pages 367–375. Society for Industrial and Applied Mathematics, 1992.
- [9] J. Nagasue, Y. Konishi, N. Araki, T. Sato, and H. Ishigaki. Slope-walking of a biped robot with k nearest neighbor method. In *Innovative Computing, Information and Control (ICICIC), 2009 Fourth International Conference on*, pages 173–176. IEEE, 2009.
- [10] H. Wang, J. Cao, L. Shu, and D. Rafiei. Locality sensitive hashing revisited: filling the gap between theory and algorithm analysis. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 1969–1978. ACM, 2013.
- [11] M. Zadavec, A. Brodnik, M. Mannila, M. Wanne, and B. Žalik. A practical approach to the 2d incremental nearest-point problem suitable for different point distributions. *Pattern Recognition*, 41(2):646–653, 2008.
- [12] M. Zadavec and B. Žalik. An almost distribution-independent incremental delaunay triangulation algorithm. *The Visual Computer*, 21(6):384–396, 2005.

Exploratory Equivalence on Hypercube Graphs

Jurij Mihelič
Faculty of Computer and
Information Science,
University of Ljubljana
Večna pot 113
Ljubljana, Slovenia
jurij.mihelic@fri.uni-lj.si

Uroš Čibej
Faculty of Computer and
Information Science,
University of Ljubljana
Večna pot 113
Ljubljana, Slovenia
uros.cibej@fri.uni-lj.si

Luka Fürst
Faculty of Computer and
Information Science,
University of Ljubljana
Večna pot 113
Ljubljana, Slovenia
luka.fuerst@fri.uni-lj.si

ABSTRACT

An exploratory equivalent (EE) partition of the vertex set of a graph G comprises sets of vertices that can be regarded as interchangeable when searching for copies of G in some other graph. This property may be used to speed up the search process. Since a graph may have multiple EE partitions, a natural problem is to find one that gives rise to a greatest speedup factor, i.e., a maximum EE partition. This problem is GI-hard for general graphs, so it makes sense to study restricted graph classes. In this paper, we focus on the challenging class of hypercube graphs. We present a set of rules to construct an EE partition for any such graph and prove that the resulting partition is maximum.

Categories and Subject Descriptors

G.2 [Discrete Mathematics]: Graph Theory; F.2 [Analysis of algorithms and problem complexity]: Nonnumerical Algorithms and Problems

General Terms

Graph Theory, Algorithm

Keywords

exploratory equivalence, hypercube, algorithm, Hamming distance

1. INTRODUCTION

In the world of planetary-scale networks, efficient subgraph search is of paramount importance. However, the problem of determining whether a pattern graph G is a subgraph of a host graph H (the *subgraph isomorphism problem*) is \mathcal{NP} -complete, and although several algorithms perform reasonably well in practice [1, 5, 6], they may fail if G has many isomorphisms. In such cases, *exploratory equivalence* [3] can be used to speed up the search. In particular, an *exploratory equivalent (EE) partition* of G comprises equivalence classes (disjoint sets) of vertices that can be regarded as interchangeable during the search for copies of G in H . It

can be proved that an EE partition consisting of equivalence classes P_1, \dots, P_s reduces the number of subgraph isomorphisms $f: G \rightarrow H$ that have to be considered during the search by a factor of $\prod_{i=1}^s |P_i|!$. The goal of the *maximum EE partition problem* (MAXEE) is to find an EE partition that maximizes the reduction factor.

For general graphs, the MAXEE problem is GI-hard [2]. Therefore, it makes sense to study restricted classes of graphs. In this paper, we deal with *hypercube graphs*, which are important both theoretically and practically. Being regular and hence highly symmetric, these graphs are considered a suitable choice for the topology of a communication network [4].

The MAXEE problem on hypercube graphs has turned out to be surprisingly challenging. Nevertheless, we have devised a set of simple rules to construct a maximum EE partition for any such graph. Following the necessary definitions (Section 2), we give the construction rules and prove that they indeed produce a maximum EE partition (Section 3). Section 4 concludes the paper.

2. PRELIMINARIES

Let $G = (V, E)$ with the vertex set $V = \{1, \dots, n\}$ and the edge set $E \subseteq V \times V$ be a simple undirected graph. Given another simple undirected graph, $H = (U, F)$, an *isomorphism* $f: G \rightarrow H$ is a bijective mapping such that $(f(u), f(v)) \in F$ iff $(u, v) \in E$. An *automorphism* G is an isomorphism from G to itself, and a *subgraph isomorphism* $f: G \rightarrow H$ is an isomorphism between G and a subgraph of H .

The set of automorphisms of a graph G , $\text{Aut}(G)$, forms a group under composition. A set $A \subseteq \text{Aut}(G)$ covers a set $P \subseteq V$ (denoted $\text{cover}(A, P)$) if for each permutation σ of P there exists an automorphism $a \in A$ such that $a(i) = \sigma(i)$ for all $i \in P$. The *pointwise stabilizer* of a set $A \subseteq \text{Aut}(G)$ with respect to a set $P \subseteq V$ is the set $\text{PointStab}(A, P) = \{a \in A \mid \forall i \in P: a(i) = i\}$. An ordered partition $\langle P_1 \mid P_2 \mid \dots \mid P_s \rangle$ of G (with $\bigcup_{i=1}^s P_i = V$, $\forall i, j, i \neq j: P_i \cap P_j = \emptyset$, and $\forall i: P_i \neq \emptyset$) is *exploratory equivalent* if $\text{cover}(A_{i-1}, P_i)$ and $A_i = \text{PointStab}(A_{i-1}, P_i)$ for all $i \in \{1, \dots, s\}$, where $A_0 = \text{Aut}(G)$.

If $\langle P_1 \mid \dots \mid P_s \rangle$ is an EE partition of G and $P_i = \{v_{i1}, \dots, v_{ik_i}\}$, then for each copy G' of G in H there exists an isomorphism $f: G \rightarrow G'$ with $f(v_{i1}) < \dots < f(v_{ik_i})$ for all $i \in \{1, \dots, s\}$. The number of subgraph isomorphisms

to be considered during the search for copies of G in H is thus reduced by $\prod_{i=1}^s |P_i|!$ — the *score* of an EE partition $\langle P_1 \mid \dots \mid P_s \rangle$. The goal of the MAXEE *problem* is to find a maximum-score EE partition of G .

For example, the maximum EE partition of a graph G with $n = 4$ and $E = \{(1, 2), (2, 3), (3, 4), (1, 4)\}$ (a 4-cycle and also a 2-hypercube) is $\langle 1, 3 \mid 2, 4 \rangle$. The set $\text{Aut}(G)$ includes the automorphisms **1234** and **3214** ($1 \mapsto 3, 2 \mapsto 2, 3 \mapsto 1, 4 \mapsto 4$), which cover the set $P_1 = \{1, 3\}$, and the set $\text{PointStab}(\text{Aut}(G), P_1) = \{1234, 1432\}$ covers the set $P_2 = \{2, 4\}$. The partitions $\langle 1, 2 \mid 3 \mid 4 \rangle$ and $\langle 1 \mid 2 \mid 3 \mid 4 \rangle$ are also EE but not maximum.

The *Hamming distance* between binary vectors $p = (p_1, \dots, p_d)$ and $q = (q_1, \dots, q_d)$ is $h(p, q) = \sum_{i=1}^d |p_i - q_i|$. A binary vector $\text{bin}_d(r) = (b_1, \dots, b_d)$ is the *binary representation* of an integer r if $r = \sum_{i=1}^d 2^{d-i} b_i$. The *d-hypercube graph* (or simply the *d-hypercube*) is the graph Q_d with $n = 2^d$ and $E = \{(u, v) \mid h(\text{bin}_d(u), \text{bin}_d(v)) = 1\}$. The vertices of a d -hypercube will be labeled $\text{bin}_d(1), \dots, \text{bin}_d(2^d)$ rather than $1, \dots, 2^d$.

3. HYPERCUBES

In this section we focus on the exploratory equivalence of *hypercube* graphs. Such a graph contains vertices and edges of a d -dimensional hypercube and is denoted with Q_d . It contains 2^d vertices, $d2^{d-1}$ edges, and is a *regular* graph of degree d . Its number of automorphisms is $|\text{Aut}(Q_d)| = d! 2^d$.

A straightforward procedure to generate the hypercube of a given dimension d is to create a vertex for each d -digit binary number and connect two vertices with an edge if their Hamming distance is one. See Figure 1 for several examples of hypercubes of dimensions from 1 to 4 as well as their respective maximum exploratory equivalent partitions; for example, the corresponding partition of Q_4 is $\langle 0000, 1111 \mid 1000, 0100, 0010, 0001 \rangle$.

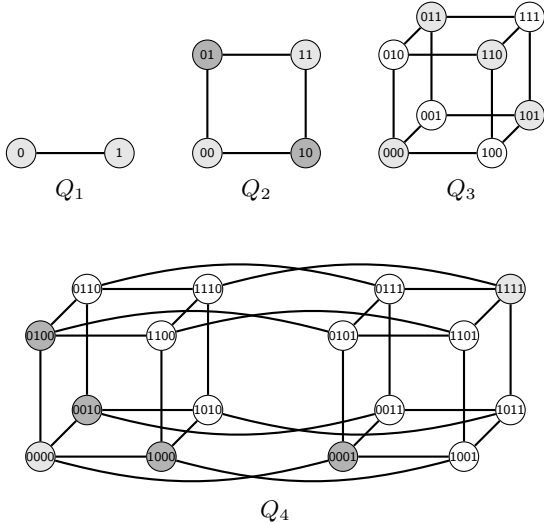


Figure 1: Several hypercubes and their maximum exploratory equivalent partitions. Vertices in the same class are of the same shade of gray; singletons are white.

In what follows, we first show how to construct maximum exploratory equivalent partition of a hypercube graph Q_d of dimension d , then we prove its correctness and optimality.

3.1 Construction

There are at most two non-singleton classes in an optimal exploratory equivalent partition of Q_d for any dimension d . In particular, hypercubes Q_1 and Q_3 result in one such class, whereas all other Q_d 's result in two such classes. Our construction (described below and denoted with HCEE) results in an exploratory equivalent partition, which is also optimal for any hypercube Q_d except for Q_3 which we deal with separately. The classes are as follows:

- The first class consists of any two vertices which are the farthest apart, i.e., their Hamming distance is d . For example, Q_1 gives $\{0, 1\}$, Q_2 gives $\{00, 11\}$ or $\{01, 10\}$, and Q_4 gives $\{0000, 1111\}$, etc.
- The second class (when $d \geq 2$) consists of all the vertices adjacent to one of the vertices in the first class. For example, taking the vertex labeled with d zeros, Q_2 gives $\{10, 01\}$, and Q_4 gives $\{1000, 0100, 0010, 0001\}$ as the second class.
- All other classes (when $d \geq 4$) are singletons, i.e., each vertex not in the first or the second class is a separate class.

Several examples of such construction of exploratory equivalent partitions are shown in Figure 1 (for Q_1, Q_2 , and Q_4), and Figure 2 a) (for Q_3 , non-optimal).

LEMMA 1. *Given a hypercube graph Q_d of dimension d , where $d \geq 1$, the HCEE construction produces an exploratory equivalent partition.*

PROOF. The first class is clearly exploratory equivalent; indeed, any two vertices of Q_d would suffice, but selecting the two farthest apart (denoted here with u and v) leaves the most room for the second class. There is no other class in Q_1 . Alternatively, notice that, $\mathcal{N}(u) = \mathcal{N}(v)$ in Q_2 , otherwise, when $d \geq 3$, neighborhoods are disjoint, i.e., $\mathcal{N}(u) \cap \mathcal{N}(v) = \emptyset$ (since $h(u, v) = d$, hence, the distance between vertices from $\mathcal{N}(u)$ and $\mathcal{N}(v)$ is at least $d - 2 \geq 1$).

Fix both vertices u and v from the first class, i.e., $A_1 = \text{PointStab}(\text{Aut}(Q_d), \{u, v\})$. Observe that, $\text{cover}(A_1, \mathcal{N}(u))$ is satisfied, since interchanging any two vertices from $\mathcal{N}(u)$ is possible (and leaving all other from $\mathcal{N}(u)$ on their position). Consequently, all possible permutations of $\mathcal{N}(u)$ are attainable. \square

Observing that $|\mathcal{N}(u)| = |\mathcal{N}(v)| = d$, gives the following corollary.

COROLLARY 1. *Given a hypercube graph Q_d of dimension d , where $d \geq 2$, the HCEE construction produces an exploratory equivalent partition having $d + 2$ vertices in its non-singleton classes, and $2^d - d - 2$ in its singleton classes. The score of such partition is $2^d d!$.*

Following the steps of the construction gives rise to many different exploratory equivalent partitions. We give their count in the following lemma.

LEMMA 2. *The HCEE construction can produce 2^d different exploratory equivalent partitions of the hypercube graph Q_d of dimension d , where $d \geq 3$.*

PROOF. The first pair of vertices may be selected on 2^{d-1} ways, but then there are only two available neighborhoods to select from. \square

Representing a hypercube Q_d explicitly with a list of vertices and edges is deemed very inefficient. Hence, we assume the input to the partition construction algorithm is only a number d of dimensions, which is, in general, of $n = O(\lg d)$ bits long. In this sense, even outputting one vertex label requires exponential time, i.e., $O(d) = O(2^n)$. Furthermore, when $d \geq 2$, there are $2 + d$ vertices in non-singleton classes. Thus, we have the following lemma.

LEMMA 3. *The time complexity of the HCEE construction is $O(d^2)$.*

In practice, when d is small enough, i.e., on today's architectures $d \leq 64$, one can assume that outputting a d -bit number is $O(1)$.

3.2 Optimality

To prove the optimality of our construction we need an additional notion of vertices whose distance from each other is the same. First, for each graph Q_d , we define a parameterized family of sets containing vertices of Q_d , where the Hamming distance between any two nodes in the set equals to h , i.e.,

$$\mathcal{H}_d(h) = \{H \subseteq V(Q_d) \mid \forall u, v \in H : h(u, v) = h\}.$$

Now, we determine the size of a maximum set in $\mathcal{H}_d(h)$, i.e.,

$$\xi_d(h) = \max_{H \in \mathcal{H}_d(h)} |H|.$$

In what follows we are interested into an upper bound on $\xi_d(h)$, since exploratory equivalent classes are subsets of $\mathcal{H}_d(h)$. In particular, we have the following lemma.

LEMMA 4. *Given a hypercube graph Q_d of dimension d and its exploratory equivalent partition $\langle P_1, \dots, P_s \rangle$, for any $1 \leq i \leq s$, it holds that $P_i \in \mathcal{H}_d(h)$ for some h .*

PROOF. Singleton classes are contained in $\mathcal{H}_d(0)$, and any two-vertex class $\{u, v\} \in \mathcal{H}_d(h(u, v))$. Now, consider three-vertex class $\{u, v, w\}$. If we interchange u and v , then w remains fixed only if $h(u, w) = h(v, w)$, and, similarly, for all other pairs in the class. \square

The following two theorems specify $\xi_d(h)$ for odd and even h , respectively, and are also of its own interest (see also Figure 1).

d \ h	0	2	4	6	8	10	12	14	16
1	1								
2	1	2							
3	1	4							
4	1	4	2						
5	1	5	2						
6	1	6	4	2					
7	1	7	4	2					
8	1	8	4	2	2				
9	1	9	4	4	2				
10	1	10	5	4	2	2			
11	1	11	5	4	2	2			
12	1	12	6	4	4	2	2		
13	1	13	6	4	4	2	2		
14	1	14	7	4	4	2	2	2	
15	1	15	7	5	4	4	2	2	
16	1	16	8	5	4	4	2	2	2

Table 1: A tabular representation of $\xi_d(h)$ from Theorem 2. Exceptions using the third case are framed. Empty cells are zeros.

THEOREM 1. *Given a hypercube graph Q_d of dimension d , where $d \geq 1$, if h is odd then $\xi_d(h) = 2$.*

PROOF. For $d = 1$ this is straightforward. Without loss of generality, consider the vertex labeled $0 \dots 0$. Any vertex with the distance h from it contains h ones and $d - h$ zeros. Now to obtain the third vertex with the distance h from the second toggle p ones to zero, and q zeros to one, where $p + q = h$. However, the third cannot be on the distance h from the first: their distance is $h - p + q = 2h - 2p \neq h$, since h is odd. \square

THEOREM 2. *Given a hypercube graph Q_d of dimension d , where $d \geq 1$, if h is even then*

$$\xi_d(h) = \begin{cases} 0 & d < h \\ 1 & h = 0 \\ 4 & 3/2h \leq d < 2h \\ \lfloor \frac{d}{h/2} \rfloor & h \leq d < 3/2h \vee d \geq 2h. \end{cases}$$

PROOF. The first two cases are obvious: there are no vertices $u, v \in Q_d$ with $h(u, v) > d$ and $h(u, v) = 0$ if only if $u = v$.

Now, consider the last two cases, where $0 < h \leq d$. To construct the maximum size set begin from any vertex and observe positions (in binary representation) altered when constructing the next vertex. Obviously, at each step there are h positions altered, but to permit further steps $h/2$ of them are the ones just altered in the previous step, and $h/2$ of them are the new ones (i.e., not yet altered). Observe that, any other technique blocks further steps. Proceeding in this manner one can produce $\lfloor d/(h/2) \rfloor$ vertices, since there are at most d positions altered in total. For example, for Q_4 and $h = 2$ we get 1000, 0100, 0010, 0001, and for

Q_9 and $h = 4$ we get 110000000, 001100000, 000011000, 000000110.

There is an exception to the technique (represented by the second case), when only three vectors are generated, e.g., for Q_3 and $h = 2$ we get 100,010,001, but better solution is 000, 110, 101, 011. Notice, that the second condition gives $\lfloor \frac{d}{h/2} \rfloor + 1 = 4$. \square

Using these two theorems and Lemma 4 we now have the upper bound specified by the following corollary.

COROLLARY 2. *Given a hypercube graph Q_d of dimension d and its exploratory equivalent partition $\langle P_1, \dots, P_s \rangle$, for any $1 \leq i \leq s$, it holds that*

$$|P_i| \leq \begin{cases} 4 & d = 3 \\ d & d \neq 3. \end{cases}$$

Now, separately consider the hypercube Q_3 . Its optimal solution is shown in Figure 2 b). Indeed, observe that it is exploratory equivalent (by interchanging any two vertices in the non-singleton partition the other vertices remain fixed). Additionally, its size is 4, which is, due to Theorem 2, at most $\xi_3(h) \leq 4$ for any h . The solution is thus optimal, and we have the following theorem.

THEOREM 3. *A maximum exploratory equivalent partition of the hypercube graph Q_3 is $\{000, 011, 101, 110\}$ with the score of 24.*

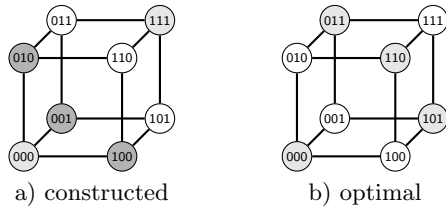


Figure 2: 3-dimensional hypercube: constructed vs. optimal exploratory equivalent partition.

Our main results is summarized in the following theorem.

THEOREM 4. *Given a hypercube graph Q_d of dimension d , where $d \geq 4$, the HCEE construction produces a maximum exploratory equivalent partition with the score $2d!$.*

PROOF. We will prove that 1) we cannot improve the two constructed classes and 2) we will show that using more than two classes would also produce a lower score. The optimality of the obtained solution thus follows.

1) Since the size of any class is $\leq d$, we cannot increase the second class, i.e., the $d!$ factor. Assume, that we can improve upon the first class, say $k = |P_1| \geq 3$, potentially decreasing the size of the second partition by x . The improved score would thus be $k!(d-x)! > 2d!$. Furthermore, $k^{\frac{k-2}{2}} > d^{\frac{x}{2}}$.

Thus, $x \leq k-3$ and the size of the second class $\geq d-(k-3)$. Observe that a class of size l alters at least $l-1$ positions in its containing vectors. Together both classes would now alter $(k-1) + (d-(k-3)-1) = d+1$ positions, which is impossible on Q_d .

2) Notice also that, $d-k+2$ is the upper bound on the size of the second class as well as on the total size of all non-singleton classes except the first. Thus, using even more than two classes cannot improve the score, since $c! \geq a!b!$, where $c = a+b$. \square

4. CONCLUSIONS

In this article we dealt with the exploratory equivalence on d -dimensional hypercube graphs. We presented an efficient construction algorithm for an exploratory equivalent partition, which was further proven to be optimal.

While proving the optimality we also observed another interesting property of hypercubes: the maximum cardinality of the vertex sets with a given Hamming distance (pairwise).

For the future work we will explore a generalisation of hypercubes, i.e. hypergrids.

5. ACKNOWLEDGMENTS

This work was partially supported by the Slovenian Research Agency and the projects "P2-0095 Parallel and distributed systems" and "N2-0053 Graph Optimisation and Big Data".

6. REFERENCES

- [1] L. P. Cordella, P. Foggia, C. Sansone, and M. Vento. A (sub)graph isomorphism algorithm for matching large graphs. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 26(10):1367–72, Oct. 2004.
- [2] L. Fürst, U. Čibej, and J. Mihelič. Maximum exploratory equivalence in trees. In *2015 Federated Conference on Computer Science and Information Systems, FedCSIS 2015, Łódź, Poland, September 13-16, 2015*, pages 507–518, 2015.
- [3] J. Mihelič, L. Fürst, and U. Čibej. Exploratory equivalence in graphs: Definition and algorithms. In *2014 Federated Conference on Computer Science and Information Systems, Fedcsis 2014, Warsaw, Poland, September 7-10, 2014*, pages 447–456, 2014.
- [4] T. H. Szymanski. On the permutation capability of a circuit-switched hypercube. In *International Conference on Parallel Processing (ICPP '89)*, pages 103–110, 1989.
- [5] J. R. Ullmann. An Algorithm for Subgraph Isomorphism. *J. Assoc. for Computing Machinery*, 23:31–42, 1976.
- [6] U. Čibej and J. Mihelič. Improvements to Ullmann's algorithm for the subgraph isomorphism problem. *International Journal of Pattern Recognition and Artificial Intelligence*, 29(07):1550025, 2015.

Partitioning polyominoes into polyominoes of at most 8 vertices, mobile vs. point guards

Ervin Györi^{a*} Tamás Róbert Mezei^{b†}

August 29, 2016

Abstract

We prove that every simply connected polyomino of n vertices can be partitioned into $\lfloor \frac{3n+4}{16} \rfloor$ (simply connected) polyominoes of at most 8 vertices. It yields a new and shorter/simpler proof of the theorem of A. Aggarwal that $\lfloor \frac{3n+4}{16} \rfloor$ mobile guards are sufficient to control the interior of an n vertex orthogonal polygon. Moreover, we strengthen this result by requiring combinatorial guards (visibility is only needed at the endpoints of patrols) and prohibiting intersecting patrols. This yields positive answers to two questions of O'Rourke [7, Section 3.4]. Our result is also a further example of the metatheorem that (orthogonal) art gallery theorems are based on partition theorems. We also found an interesting sharp bound on the ratio of the necessary number of appropriate mobile and point guards

Kahn, Klawe and Kleitman in 1980 proved that $\lfloor \frac{n}{4} \rfloor$ guards are sometimes necessary and always sufficient to cover the interior of an orthogonal polygon

^aMTA Renyi Institute/CEU Math. Dept. (Budapest) gyori@renyi.hu

[†]CEU Math. Dept. (Budapest) tamasrobert.mezei@gmail.com

of n vertices. Later the first author of this paper provided a simple and short proof of

Theorem 1 ([3], [7, p. 68]). *Every simply connected polyomino of n vertices can be partitioned into $\lfloor \frac{n}{4} \rfloor$ polyominoes of at most 6 vertices.*

[?] is a deeper result than that of Kahn, Klawe and Kleitman, and gave the first hint of the existence of a “metatheorem”: (orthogonal) art gallery theorems have underlying partition theorems. The general case was proved by Hoffmann [5].

Theorem 2 ([5]). *Every (not necessarily simply connected!) polyomino with n vertices can be covered by $\lfloor n/4 \rfloor$ guards.*

Hoffmann’s method (partitioning into smaller polyominoes that can be covered by one guard) is another proof of the metatheorem.

In this paper, we present further evidence that the metatheorem holds, namely we prove the following partition theorem:

Theorem 3. *Any simple polyomino of n vertices can be polyomino-partitioned into at most $\lfloor \frac{3n+4}{16} \rfloor$ polyominoes of at most 8 vertices.*

The mobile guard art gallery theorem for simple orthogonal polygons follows immediately, as a polyomino of at most 8 vertices can be covered by a guard.

Theorem 4 ([1], proof in [7, p. 91]). $\lfloor \frac{3n+4}{16} \rfloor$ *mobile guards are sufficient for covering an n vertex simple orthogonal polygon.*

The proof of Aggarval is about 20 pages, so we not only present a shorter proof, but also provide a stronger result which is very interesting on its own. It fits into the series of results in [3], [5], [7, p. 68] showing that rectilinear art gallery theorems are based on theorems on partitions of polyominoes into smaller (one guardable) polyominoes.

The proof of the main theorem is similar to O'Rourke's proof in that it finds a suitable cut and then uses induction on the parts created by the cut. However, here a cut along a line connecting to concave vertices is not automatically good. In case we have no such cuts, we also rely heavily on a tree structure of the polyomino. However, we must consider L-shaped cuts too, which are responsible for most of the extra complexity of our analysis.

The proof yields an $O(n^2)$ algorithm partitioning P into at most $\frac{3n+4}{16}$ simple polyominoes. The running time can be improved to $O(n)$ by using linear-time triangulation (Chazelle). Theorem 4 fills a gap between two already established (sharp) results: in [3] it is proved that polyominoes can be partitioned into at most $\frac{n}{4}$ polyominoes of at most 6 vertices, and in [4] it is proved that any polyomino in general position (a polyomino without 2-cuts) can be partitioned into $\frac{n}{6}$ polyominoes of at most 10 vertices. However, we do not know of a sharp theorem about partitioning polyominoes into polyominoes of at most 12 vertices. Furthermore, for $k \geq 4$, not much is known about partitioning (not necessarily simply connected) orthogonal polygons into polyominoes of at most $2k$ vertices. According to the metatheorem, the first step in this direction would be proving that an orthogonal polygon of n vertices with h holes can be partitioned into $\frac{3n+4h+4}{16}$ polyominoes of at most 8 vertices. This would generalize the corresponding art gallery result in [4, Thm. 5].

From the mentioned results we can read off that from an extremal point of view point guards are $3/4$ as efficient as mobile guards. The following theorem provides insight into why this is the case, as the $3/4$ bound appears already for a single polygon.

Theorem 5. *Given a P simple orthogonal polygon let m_V be the minimum number of vertical mobile guards necessary to cover P , and let m_H be defined analogously for horizontal mobile guards, and finally let p be the minimum number of point guards necessary to cover P . Then $\frac{m_V+m_H-1}{p} \geq \frac{3}{4}$.*

The proof of this theorem can be turned into an $8/3$ -approximation algo-

rithm for covering simple orthogonal polygons with point guards.

References

- [1] A. Aggarwal, The art gallery theorem: its variations, applications, and algorithmic aspects, Ph.D. thesis, Johns Hopkins Univ. (1984)
- [2] Chazelle, B., 1991. Triangulating a simple polygon in linear time. *Discrete and Computational Geometry* 6, 485–524.
- [3] E. Györi, A short proof of the rectilinear art gallery theorem, *SIAM J. Alg. Disc. Meth.* 7 (1986), 452-454.
- [4] Györi, E., Hoffmann, F., Kriegel, K., Shermer, T., Generalized guarding and partitioning for rectilinear polygons. *Computational Geometry* 6 (1996), 21–44.
- [5] F. Hoffmann, On the rectilinear art gallery problem, *Proc. ICALP '90, Lecture Notes in Comp. Sci.*, 443 (1990), pp. 717–728
- [6] J. Kahn, M. Klawe, and D. Kleitman, Traditional galleries require fewer watchmen, *SIAM J. Alg. Disc. Meth.* 4 (1983), 194-206.
- [7] J. O'Rourke, Art gallery theorems and algorithms, *Oxford University Press* 57 (1987)

On Linear Grammars with Exact Control

Dávid Angyal
Department of Computer Science
Faculty of Informatics
University of Debrecen
angyal.david@inf.unideb.hu

Benedek Nagy
Department of Mathematics
Faculty of Arts & Sciences
Eastern Mediterranean University
nbenedek.inf@gmail.com

ABSTRACT

Grammars with exact control are controlled grammars with the condition that every word of the control language results at least one word of the derived language. In this paper, an infinite family of semi-linear languages is presented where the base grammar is a linear grammar and the control language is a linear language or a language class obtained in this manner. Already the class of languages generated by linear grammars with exact linear control contains some non context-free languages. Normal form result for these systems and pumping lemmas are shown to help to prove the infinite hierarchy.

Keywords

linear grammars, controlled grammars, parsing, hierarchy

1. INTRODUCTION

The class of linear context-free languages is already defined by Chomsky, and it is properly between the regular and context-free language classes, even some of its properties is inherited from the class of regular languages. They are recognized by finite automata equipped by two reading heads, starting from the two extremes of the input word, see [6, 7]. Because of the big gaps between language classes there are various formalisms to obtain other language classes. Several such formalisms are belonging to the field of regulated rewriting [2, 5]. As examples, matrix grammars and controlled grammars are mentioned here. Let us see, this latter ones in more details. In a controlled grammar there is a base grammar and a control language. The control language is used to filter the derivations of the base grammar: only those derivations are valid in these systems in which the derivation word is belonging to the control language. We could say that one can obtain only words that belong to derivations of the intersection of the control language and the Szilard language. If the control language is the Szilard language [4] of the base grammar, then exactly the same language is generated without and with this control. Grammars with exact

control were introduced in [1]. In these systems only those control languages are allowed that are subsets of the Szilard language (of the base grammar). In this way new classes of languages are obtained. In this paper, our starting point is the class of linear grammars, and they with linear exact control. We can also use the languages obtained in this manner as control languages for linear base grammars, and so on.

2. DEFINITIONS AND PRELIMINARIES

Let FIN, REG, LIN, CF, CS, RE denote the family of finite, regular, linear, context-free, context-sensitive, recursively enumerable languages, respectively. Let λ denote the empty word.

Definition 1. [3] A linear grammar is a quadruple $G = (N, T, S, P)$, where N is the finite set of nonterminal symbols, T is the finite set of terminal symbols, $S \in N$ is the start-symbol, and P is the finite set of productions of the form $R \rightarrow aQb$, where $R \in N$, $Q \in N \cup \{\lambda\}$, and $a, b \in T^*$. For $uxv, uyv \in (N \cup T)^*$, $uxv \Rightarrow uyv$, if $x \rightarrow y \in P$, and \Rightarrow^* is the reflexive and transitive closure of \Rightarrow . The language generated by G is defined as $L(G) := \{w \mid S \Rightarrow^* w \wedge w \in T^*\}$.

Now we recall a class of 2-head finite automata.

Definition 2. (Based on [6, 7]) A 2-head finite automaton (Q, T, q_0, δ, F) is defined, similarly to non-deterministic finite automaton, with the finite set of states Q , input (or tape) alphabet T , initial state $q_0 \in Q$, transition function $\delta : Q \times (T \cup \{\lambda\})^2 \rightarrow 2^Q$, and set of final (or accepting) states $F \subseteq Q$. Initially the 2 heads are located at the two extremes of the input word, and can read the first and the last letter of the input, respectively (if any). The next state is chosen according to δ based on the actual state and the letters read by the two heads (also it is allowed to read the empty word by any or both heads). The heads are moving opposite directions. The input word is accepted, if the automaton reaches a final state when the heads meet. The accepted language consists of every word that can be accepted by the automaton.

LEMMA 1. [6, 7] *The class of 2-head finite automata accepts exactly the class of linear context-free languages.*

LEMMA 2. (**Bar-Hillel Lemma**) [3] *For every $L \in \text{CF}$, there exists a constant $n \in \mathbb{N}$, such that if $z \in L$ and $|z| \geq n$,*

then z can be written as $z = x_0 y_1 x_1 y_2 x_2$ such that $y_1 y_2 \neq \lambda$, $|y_1 x_1 y_2| \leq n$, and $\forall i \in \mathbb{N} : x_0 y_1^i x_1 y_2^i x_2 \in L$.

Definition 3. [1] $G = (N, T, S, P, C)$ is a grammar with exact control, if

- N is the finite set of nonterminal symbols,
- T is the finite set of terminal symbols,
- $S \in N$ is the start-symbol,
- P is the finite set of productions of the form $\alpha : u \rightarrow v$, where

- α is the label (id) of the production,
- $u \in (N \cup T)^* N (N \cup T)^*$,
- $v \in (N \cup T)^*$,

- and C is a set of words over the alphabet $Labels(P) := \{\alpha \mid \alpha : u \rightarrow v \in P\}$,

- moreover the following constraint must hold:

$$\forall c_1 c_2 \dots c_n \in C : \exists w \in T^* : S \Rightarrow_{c_1} \dots \Rightarrow_{c_n} w;$$

where $xyy \Rightarrow_c xvy$, if $c : u \rightarrow v \in P$.

The generated language defined as

$$L(G) := \{w \mid \exists c_1 c_2 \dots c_n \in C : (S \Rightarrow_{c_1} \dots \Rightarrow_{c_n} w \wedge w \in T^*)\}.$$

We say that G is an X grammar with exact Y control, if

- $(N, T, S, \{u \rightarrow v \mid \alpha : u \rightarrow v \in P\})$ is a grammar of type X and
- C is a language of type Y,

where $X, Y \in \{\text{FIN, REG, LIN, CF, CS, RE}\}$. We denote the family of languages generated by X grammars with Y exact control by $\text{EC}(X, Y)$.

Definition 4. Let

- $\text{EC}^1(X, Y) := \text{EC}(X, Y)$,
- $\text{EC}^n(X, Y) := \text{EC}(X, \text{EC}^{n-1}(X, Y))$, if $n > 1$.

3. A SIMPLE PARSING ALGORITHM FOR LINEAR GRAMMARS

In this section, we assume that in linear grammars, every production is in one of the following forms: $B \rightarrow aC$, $B \rightarrow Ca$, or $B \rightarrow \lambda$ (where $B, C \in N$ and $a \in T$).

Algorithm 1. Let $w = w_1 \dots w_n$ be the input word, where $w_i \in T$ ($i \in \{1, \dots, n\}$). Let M denote an $(n+1) \times (n+1)$ upper-left-triangular matrix. We index the rows and columns from 0 to n . Each matrix entry is a subset of the nonterminal symbols, initially the empty set.

Apply the following rules until no new nonterminal symbol can be added to the matrix.

- Add S to $M(0, 0)$.
- For $i > 0$: Add A to $M(i, j)$, if $\exists B \in M(i-1, j) : B \rightarrow Aw_{n+1-i} \in P$.
- For $j > 0$: Add A to $M(i, j)$, if $\exists B \in M(i, j-1) : B \rightarrow w_j A \in P$.

After completing the matrix, check the diagonal entries for nonterminal symbols that can be erased, where $M(i, j)$ is a diagonal entry, if and only if, $i + j = n$. The word w is in the language generated by the given grammar, if and only if, there exists $E \in N$, such that E is in at least one of the diagonal entries of M , and $E \rightarrow \lambda \in P$.

Example 1. Let $G = (\{S, A, E\}, \{x, y\}, S, \{S \rightarrow yA, S \rightarrow Sx, S \rightarrow yE, A \rightarrow yS, E \rightarrow \lambda\})$ and $w = yyyxx$.

		y	y	y	x	x
	S	A, E	S	A, E		
x	S	A, E	S	A, E		
x	S	A, E	S	A, E		
y						
y						
y						

We find that $w \in L(G)$, since $E \in M(2, 3)$ and $E \rightarrow \lambda \in P$.

4. THE FAMILIES $\text{EC}^n(\text{LIN}, \text{LIN})$

In this section, we investigate some interesting properties of the language family $\text{EC}^n(\text{LIN}, \text{LIN})$ ($n \in \mathbb{N}$).

THEOREM 1. *Every language in $\text{EC}^n(\text{LIN}, \text{LIN})$ ($n \in \mathbb{N}$) is semi-linear (in Parikh-sense).*

CLAIM 1. *There exists $L \in \text{EC}(\text{LIN}, \text{LIN})$ such that L is non-context-free.*

CLAIM 2. *$\text{EC}^n(\text{LIN}, \text{LIN})$ is closed under union ($n \in \mathbb{N}$).*

LEMMA 3. *If $L \in \text{EC}^n(\text{LIN}, \text{LIN})$ and c is a terminal letter, then $L \cdot \{c\}, \{c\} \cdot L \in \text{EC}^n(\text{LIN}, \text{LIN})$ ($n \in \mathbb{N}$).*

THEOREM 2. *$\text{EC}^n(\text{LIN}, \text{LIN})$ is closed under (erasing) homomorphisms ($n \in \mathbb{N}$).*

5. NORMAL FORMS

LEMMA 4. $\text{LIN} = \text{EC}(\text{LIN}, \text{REG})$

Definition 5. A linear context-free grammar is in *simple form*, if it is given as a linear grammar with regular exact control

- having a single nonterminal symbol (i.e., $N = \{S\}$), and
- having no chain rule in its production set (i.e., there is no production $S \rightarrow S$).

LEMMA 5. *For every grammar G of type LIN there exists a grammar G' of type $\text{EC}(\text{LIN}, \text{REG})$ that is in simple form and $L(G) = L(G')$.*

Definition 6. A grammar G of type $\text{EC}^n(\text{LIN}, \text{LIN})$ ($n \in \mathbb{N}$), is in *simple form* if it has a single nonterminal symbol, and it has no chain rules.

THEOREM 3. *For every grammar G of type $\text{EC}^n(\text{LIN}, \text{LIN})$, there exists a grammar G' of type $\text{EC}^n(\text{LIN}, \text{LIN})$ in simple form ($n \in \mathbb{N}$).*

6. PUMPING LEMMAS FOR LANGUAGES GENERATED BY EXACT CONTROL

LEMMA 6. *For every $L \in \text{EC}(\text{LIN}, \text{LIN})$, there exists a constant $k \in \mathbb{N}$, such that if $w \in L$ and $|w| \geq k$, then w can be written as $w = x_0 y_1 x_1 y_2 x_2 y_3 x_3 y_4 x_4$ such that*

- $y_1 y_2 y_3 y_4 \neq \lambda$,
- $\forall i \in \mathbb{N} : x_0 y_1^i x_1 y_2^i x_2 y_3^i x_3 y_4^i x_4 \in L$.

PROOF. Let $G = (N, T, S, P, C)$ be a grammar of type $\text{EC}(\text{LIN}, \text{LIN})$ in simple form that generates L . The control language C is linear, therefore Lemma 2 can be applied. Let $n \in \mathbb{N}$ denote a pumping constant of C , and let $C|_n := \{w \in C \mid |w| < n\}$. Clearly $C|_n$ is a finite set. Consider the grammar $G' := (N, T, S, P, C|_n)$ of type $\text{EC}(\text{LIN}, \text{FIN})$. $L(G') \in \text{FIN}$ and let

$$k := 1 + \max_{w \in L(G')} \{|w|\}.$$

We claim that k is a pumping constant for L . The number n was a pumping constant for the control language of L , therefore every control word having length greater than or equal to n can be pumped. By the definition of k , $k - 1$ is the maximal length for a word in L that can be generated by control words having length less than n . Therefore, if a word in L has length of at least k , it has to be obtained by a control word having length at least n , i.e., a control word which can be pumped.

Suppose $w \in L$, $|w| \geq k$ and c is a control word for w . Then $|c| \geq n$ and by Lemma 2 $c = x_0 y_1 x_1 y_2 x_2$ and the subwords

y_1 and y_2 can be pumped. Let $y_1 = y_{1,1} \dots y_{1,l(1)}$ and $y_2 = y_{2,1} \dots y_{2,l(2)}$, where $y_{i,j} : S \rightarrow a_{i,j} S b_{i,j}$ is a production ($S \in N$, $a_{i,j}, b_{i,j} \in T^*$), for every $j \in \{1, \dots, l(i)\}$ and $i \in \{1, 2\}$. Finally, $w = x'_0 y'_1 x'_1 y'_2 x'_2 y'_3 x'_3 y'_4 x'_4$, where $y'_1 = a_{1,1} \dots a_{1,l(1)}$, $y'_2 = a_{2,1} \dots a_{2,l(2)}$, $y'_3 = b_{2,l(2)} \dots b_{2,1}$, $y'_4 = b_{1,l(1)} \dots b_{1,1}$. It is easy to see that as we pump the control word $c = x_0 y_1 x_1 y_2 x_2$ for any $i \in \mathbb{N}$, the control word $x_0 y_1^i x_1 y_2^i x_2$ will generate the word $x'_0 y_1^i x'_1 y_2^i x'_2 y_3^i x'_3 y_4^i x'_4$. \square

THEOREM 4. *For every $L \in \text{EC}^n(\text{LIN}, \text{LIN})$, there exists a constant $k \in \mathbb{N}$, such that if $w \in L$ and $|w| \geq k$, then w can be written as $w = x_0 y_1 x_1 y_2 x_2 \dots y_{2n+1} x_{2n+1}$ such that*

- $y_1 y_2 \dots y_{2n+1} \neq \lambda$,
- $\forall i \in \mathbb{N} : x_0 y_1^i x_1 y_2^i x_2 \dots y_{2n+1}^i x_{2n+1} \in L$.

PROOF. By induction on n .

- Base case: $n = 1$.
Proved in Lemma 6.
- Suppose the claim is true for every $m < n$, we show that it is also true for n .
If $L \in \text{EC}^n(\text{LIN}, \text{LIN})$ and $G = (N, T, S, P, C)$ is a grammar in simple form that generates L , then $C \in \text{EC}^{n-1}(\text{LIN}, \text{LIN})$. By the induction hypothesis, there exists $k \in \mathbb{N}$ such that every word in C having length at least k can be pumped. Now consider the grammar $G' = (N, T, S, P, \{w \in C \mid |w| < k\})$. Let

$$k' := 1 + \max_{w \in L(G')} \{|w|\}.$$

In other words, every word in L with at least length k' can be generated by a control word that can be pumped. Let $y_i := y_{i,1} \dots y_{i,l(i)}$ denote the subwords of the control word to be pumped and let $y_{i,j} : S \rightarrow a_{i,j} S b_{i,j} \in P$ ($S \in N$, $a_{i,j}, b_{i,j} \in T^*$), for every $j \in \{1, \dots, l(i)\}$ and $i \in \{1, \dots, 2^n\}$. Next we choose the subwords of the generated word: let

$$y'_i := \begin{cases} a_{i,1} \dots a_{i,l(i)}, & \text{if } 1 \leq i \leq 2^n, \\ b_{2^{n+1}-i+1, l(2^{n+1}-i+1)} \dots b_{2^{n+1}-i+1, 1}, & \text{if } 2^n + 1 \leq i \leq 2^{n+1}. \end{cases}$$

This formula is the generalised form of the formula shown in Lemma 6. There are 2^{n+1} subwords. The first 2^n subwords get the $a_{i,j}$ words in order, and the second 2^n subwords get the $b_{i,j}$ words in reverse order. It is easy to see that as the control word is pumped, the generated word pumped too, i.e., for any $i \in \mathbb{N}$ the control word $x_0 y_1^i x_1 y_2^i \dots y_{2^n}^i x_{2^n}$ generates the word $x'_0 y_1^i x'_1 y_2^i \dots y_{2^{n+1}}^i x'_{2^{n+1}}$.

\square

7. INFINITE HIERARCHY

Now we are ready to establish our main result.

THEOREM 5. $EC^n(\text{LIN}, \text{LIN}) \subsetneq EC^{n+1}(\text{LIN}, \text{LIN})$ ($n \in \mathbb{N}$).

PROOF. We are going to use the language family $L(k) := \{(a^n b^n)^k \mid n \in \mathbb{N}\}$ as separating languages, where $k \in \mathbb{N}$. Clearly $L(1) \in \text{LIN}$, for instance $G = (\{S\}, \{a, b\}, S, \{S \rightarrow aSb, S \rightarrow ab\})$ generates it.

We show that $L(2^k) \in EC^k(\text{LIN}, \text{LIN})$. In every case, the nonterminal alphabet, the terminal alphabet, the production set and the start symbol are always the same, only the control language changes.

- $N = \{S\}$,
- $T = \{a, b\}$,
- $P = \{\alpha : S \rightarrow aSb, \beta : S \rightarrow bSa, \gamma : S \rightarrow \lambda\}$.

Let $h : T^* \rightarrow \{\alpha, \beta\}^*$ be defined as $h(a) := \alpha, h(b) := \beta$. Note that h can be interpreted as a simple letter substitution. There is no need for a general homomorphism.

Induction by k :

- Base case: $k = 1$:
 $L(2)$ is generated with the control set $h(L(1)) \cdot \{\gamma\}$, i.e., $\{\alpha^n \beta^n \gamma \mid n \in \mathbb{N}\} \in \text{LIN}$.
- Suppose the claim is true for $m < k$ where $k > 1$, we show that it is also true for k :
 $L(2^k) \in EC^k(\text{LIN}, \text{LIN})$ can be generated with the control set $h(L(2^{k-1})) \cdot \{\gamma\} \in EC^{k-1}(\text{LIN}, \text{LIN})$.

See Lemma 3 on the control set modifications.

Now, we show that $L(2^{k+1}) \notin EC^k(\text{LIN}, \text{LIN})$. We can use Theorem 4. Consider the word $(a^n b^n)^{k+1}$, where n is a pumping constant of $L(2^{k+1})$. Every word in $L(2^{k+1})$ has exactly 2^{k+1} occurrences of the subword ab and exactly $2^{k+1} - 1$ occurrences of the subword ba , and therefore the word consists of 2^{k+1} blocks of a 's and the same number of blocks of b 's. The subwords $y_1, y_2, \dots, y_{2^{k+1}}$ have to be chosen in a way such that each y_i may contain only a 's or only b 's. However, the 2^{k+1} subwords can take place in at most $2^{k+1} - 1$ blocks, leaving another at least 2^{k+1} blocks out of pumping. Pumping subwords in the selected blocks will result different number of letters in the different blocks, which contradicts the language definition. \square

8. CONCLUSIONS

Grammars with exact control are a relatively new family of generative systems. In this paper linear grammars are considered. The derivations of these grammars are simple, actually, they have regular Szilard languages. However, by using an exact linear control, they are able to generate some non context-free languages. By allowing a kind of iteration on the control language, an infinite hierarchy of language classes is presented here. We close the paper with some future directions of this research line.

We have left open some interesting questions about these classes. Is $\cup_{i=1}^{\infty} EC^i(\text{LIN}, \text{LIN})$ closed under concatenation? In other words, for arbitrary $L_1, L_2 \in EC^n(\text{LIN}, \text{LIN})$ is there a language $L \in EC^m(\text{LIN}, \text{LIN})$ such that $L_1 \cdot L_2 = L$? Is the Dyck-language contained in $EC^n(\text{LIN}, \text{LIN})$ for some $n \in \mathbb{N}$? And in general, what is the relation of the family of context-free languages to the above families?

As a matter of our future work we are working on an extension of Algorithm 1 to the family $EC(\text{LIN}, \text{LIN})$.

9. ACKNOWLEDGMENTS



EMBERI ERŐFORRÁSOK MINISZTERIUMA

Supported through the New National Excellence Program of the Ministry of Human Capacities

The authors thank the helpful comments of the anonymous reviewer.

10. REFERENCES

- [1] D. Angyal and B. Nagy. On language families generated by controlled grammars. In R. Freund, M. Holzer, N. Moreira, and R. Reis, editors, *Seventh Workshop on Non-Classical Models of Automata and Applications - NCMA 2015, Porto, Portugal, August 31 - September 1, 2015. Proceedings*, volume 318 of *books@ocg.at*, pages 59–72. Österreichische Computer Gesellschaft, 2015.
- [2] J. Dassow and Gh. Păun. *Regulated Rewriting in Formal Language Theory*, volume 18 of *EATCS Monographs in Theoretical Computer Science*. Springer-Verlag Berlin, 1989.
- [3] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, 1979.
- [4] E. Mäkinen. A bibliography on Szilard languages, technical report, department of computer sciences, university of tampere, 1998.
- [5] A. Meduna and P. Zemek. *Regulated Grammars and Automata*. Springer, 2014.
- [6] B. Nagy. On $5' \rightarrow 3'$ sensing Watson-Crick finite automata. In M. H. Garzon and H. Yan, editors, *DNA Computing, 13th International Meeting on DNA Computing, DNA13, Memphis, TN, USA, June 4-8, 2007, Revised Selected Papers*, volume 4848 of *LNCS*, pages 256–262. Springer, 2008.
- [7] B. Nagy. On a hierarchy of $5' \rightarrow 3'$ sensing Watson-Crick finite automata languages. *J. Log. Comput.*, 23(4):855–872, 2013.

Some computable functions without Brouwer fixed-points *

[Extended Abstract] †

Petrus H. Potgieter
Department of Decision Sciences
University of South Africa (Pretoria)
php@member.ams.org
potgiph@unisa.ac.za

ABSTRACT

This paper is an overview of results that show the Brouwer fixed-point theorem to be essentially non-constructive and non-computable and discusses some computable functions without computable fixed points. The counter-examples of Orevkov and Baiggeer that imply that there is no procedure for finding the fixed point in general and do so by giving an example of a computable function which does not fix any computable point. In this contribution, we discuss some examples of computable functions not fixing any computable point.

Categories and Subject Descriptors

G.1.0 [Mathematics of computing]: Numerical analysis—*General*

General Terms

Theory

Keywords

Computable analysis, Brouwer fixed-point theorem

1. INTRODUCTION

Recall that computable real number is a number for which a Turing machine exists that, on input n , produces a rational approximation with error no more than 2^{-n} . A computable

*Research supported in part by NRF incentive grant IFR2011041500051 and the EU project COMPUTAL as well as the University of South Africa's College of Economic and Management Sciences Research Committee. All opinions expressed in this work are the author's and in not necessarily endorsed or supported by his present, future or past employer(s) and/or funder(s).

†Most of the material in this paper can also be found in arXiv:0804.3199 [math.GM], with more detail and diagrams.

point is a point all the coordinates of which are computable reals. The notation

\mathbb{N}_0 for the non-negative natural numbers;
 \mathbb{R}_c for the set of computable reals;
 I_c for $I \cap \mathbb{R}_c$; and
 δX for the boundary of a set X , being $\overline{X} \cap \overline{X^c}$

is also used.

We consider the Brouwer fixed-point theorem in the following form, where the standard unit interval is denoted by $I = [0, 1]$.

THEOREM 1 (BROUWER). *Any continuous function $f : I^2 \rightarrow I^2$ has a fixed point, i.e. there exists an $x \in I^2$ such that $f(x) = x$.*

The two examples discussed use distinct definitions of a computable function of real variables.

Russian school

In the Russian school of Markov and others, a computable function maps computable reals to computable reals by a single algorithm for the function that translates an algorithm approximating the argument to an algorithm approximating the value of the functions. It need not be possible to extend a function that is computable in the Russian school to a continuous function on all of the reals. These functions are often called *Markov-computable*.

Polish school

In the Polish school of Lacombe, Grzegorzczuk, Pour-El and Richards, and others, a function is computable on a region if it maps every computable sequence of reals to a computable sequence of reals and it has a computable uniform modulus of continuity on the region [9].

2. OREVKOV'S EXAMPLE

One can construct a Markov-computable function f through a computable mapping of descriptions of computable points $x \in I_c^2$ to descriptions of $f(x) \in I_c^2$, such that

$$f(x) \neq x \quad \forall x \in I_c^2.$$

That is, no computable point is a fixed point for f . Unfortunately the f which is constructed in this way, cannot be extended to a continuous function on I^2 . This is the (Russian school) construction of [7], another instance of which can be found in [12].

3. BAIGGER'S EXAMPLE

For the nowadays more current approach of the Polish school, a counter-example was constructed by Baigger [1]. Let a be any non-computable point in I^2 . Consider the function f which moves each point half-way to a ,

$$f(x) = x + \frac{1}{2}(a - x)$$

and has a single fixed point, namely a itself. The function f is continuous and defined on all of I^2 and has no computable fixed point. Nevertheless, this is not really interesting since

- the fixed point a has no reasonable description—since it is itself not computable; and therefore
- the function f has no reasonable description—it is not computable in any sense.

One would like to see a function which is computable, defined (and therefore continuous) on all of I^2 and yet avoids fixing any of the computable points I_c^2 . The following example, having appeared in [1] and in [12], modifies the construction of Orevkov to produce a computable f defined on all of I^2 having no computable fixed point. As in the example of Orevkov, we need the following fact.

LEMMA 1 ([6], FOR EXAMPLE). *There exist computable sequences of rational numbers (a_n) and (b_n) in the interval $I = [0, 1]$ such that the intervals $J_n = [a_n, b_n]$ have the following properties.*

- (i) *If $n \neq m$ then $|J_n \cap J_m| \leq 1$.*
- (ii) *If $a_n \neq 0$ then $a_n \in \{b_0, b_1, \dots\}$ and if $b_n \neq 1$ then $b_n \in \{a_0, a_1, \dots\}$.*
- (iii) *$I_c \subsetneq \bigcup_n J_n$, i.e. the J_n cover the computable reals in $I = [0, 1]$.*

One uses the intervals $J_n = [a_n, b_n]$ of the lemma above and sets

$$C_n = \bigcup_{k, \ell \leq n} J_k \times J_\ell$$

after which one defines f progressively, using the sets C_n . The points

$$t_n = (v_n, v_n)$$

where

$$v_n = \min_{x \in I} \{x \mid (x, x) \notin C_n\}$$

are used as “target point” at each stage of the construction. Note that

$$v = \lim_{n \rightarrow \infty} v_n$$

is not a computable number and (v, v) will be one of the fixed points of f .

DEFINITION 1. *For any $W \subseteq I^2$ we define*

$$W^{\blacksquare \varepsilon} = \{x \in W \mid d(x, \delta W \setminus \delta I^2) \geq \varepsilon\}$$

and

$$W^{\square \varepsilon} = \{x \in W \mid d(x, \delta W \setminus \delta I^2) = \varepsilon\}.$$

One can define f_n such that

1. f_n moves every point in the interior of $C_n^{\blacksquare 2^{-n}}$ but is the identity outside the set, and is computable;
2. f_{n+1} agrees with f_n on $C_n^{\blacksquare 2^{-n} \cdot \frac{3}{2}}$ and therefore
3. $f = \lim_{n \rightarrow \infty} f_n$ is computable.

Every computable point eventually lies in some

$$C_n^{\blacksquare 2^{-n} \cdot \frac{3}{2}} \subset \left(C_n^{\blacksquare 2^{-n}}\right)^\circ$$

and is therefore moved by f . Clearly $f(I^2) \subseteq I^2$ and f will be as required. In fact, f has no fixed point in

$$\bigcup_n C_n = \bigcup_{k, \ell \geq 1} J_k \times J_\ell.$$

Also, f as constructed here has no isolated fixed point—its fixed points all occur on horizontal and vertical lines spanning the height and breadth of the unit square. Further details of the construction appear in Appendix A. The construction cannot be applied in the one-dimensional case because it is impossible to effect a change of direction by continuous rotation and, of course, in one dimension one can compute the fixed point.

4. THE KÖNIG LEMMA

In reverse mathematics it is known that in RCA_0 , the system of recursive comprehension and Σ_1^0 -induction, the weak König lemma, WKL_0 , is equivalent to the Brouwer FPT [11].

LEMMA 2 (WKL_0 , KÖNIG). *Every infinite binary tree has an infinite branch.*

The König lemma does not have a direct computable counterpart.

THEOREM 2 (KLEENE [5]). *There exists an infinite binary tree, all the computable paths of which are finite.*

The relation of the Kleene tree to the Baigger counterexample can be relatively easily constructed and is detailed in [8].

5. REMARKS

We close with some remarks about the fixed points of computable functions that may be of interest.

THEOREM 3. *If $g : \subseteq \mathbb{R}^n \rightarrow \mathbb{R}^n$ is computable on $[a_1, b_1] \times \dots \times [a_n, b_n]$ and has an isolated fixed point in $(a_1, b_1) \times \dots \times (a_n, b_n)$ then that fixed point is a computable point.*

This result is mentioned by [4], for example, with a proof outline.

PROOF. It is sufficient to consider

$$h(x) = \|g(x) - x\|$$

and to show that if h has an isolated zero then that zero is computable. Assume therefore that $h(z) = 0$ where

$$\begin{aligned} z \in [e_1, f_1] \times \cdots \times [e_n, f_n] &\subseteq (c_1, d_1) \times \cdots \times (c_n, d_n) \\ &\subseteq (a_1, b_1) \times \cdots \times (a_n, b_n) \end{aligned}$$

and $h(x) > 0$ whenever $x \in (c_1, d_1) \times \cdots \times (c_n, d_n) \setminus \{z\}$ and all the given interval bounds are computable numbers.

Suppose now that z were non-computable. Let (q_n) be any computable enumeration of the rational points in $(c_1, d_1) \times \cdots \times (c_n, d_n)$. Since obviously $z \notin \{q_1, q_2, \dots\}$,

$$h(q_n) > 0 \quad \forall n$$

and hence for each n it is possible to choose ε_n by setting

$$\varepsilon_n = \frac{1}{4}h(q_n)$$

so that $\varepsilon_n > 0$ for each n . Since h is computable on $[a_1, b_1] \times \cdots \times [a_n, b_n]$ it has a decreasing modulus of uniform continuity m_h . Define

$$\delta_n = m_h(\varepsilon_n)$$

so that for each n it is true that

$$\|x - q_n\| < \delta_n \quad \Rightarrow \quad h(x) > \frac{3}{4}h(q_n)$$

and that on the open ball $B(q_n, \delta_n)$ the function h is bounded strictly away from 0 by $\frac{3}{4}h(q_n) > 0$.

The $B(q_n, \delta_n)$ now form an open cover of $[e_1, f_1] \times \cdots \times [e_n, f_n] \setminus \{z\}$. For, given any $x \neq z$ in $[e_1, f_1] \times \cdots \times [e_n, f_n]$ it is the case that h is bounded away from 0 on

$$V = [e_1, f_1] \times \cdots \times [e_n, f_n] \setminus B\left(z, \frac{1}{2}\|z - x\|\right)$$

and hence there exists a $\kappa > 0$ such that $\varepsilon_n > \kappa$ whenever $q_n \in V$ and therefore

$$\delta_n > m_h(\kappa) > 0 \quad \text{for each } q_n \in V.$$

As a result, $x \in B(q_i, m_h(\kappa)) \subset B(q_i, \delta_i)$ for some $q_i \in V$.

We now have computable sequences (q_n) and (δ_n) which allow us to approximate z arbitrarily closely. For, given any $\varepsilon > 0$ the balls $B(q_n, \delta_n)$ form an open cover of the compact set

$$[e_1, f_1] \times \cdots \times [e_n, f_n] \setminus B\left(z, \frac{\varepsilon}{2}\right)$$

so, if a computable ε is given, an enumeration of the balls can be done, stopping as soon as the complement of

$$\bigcup_{i=1}^m B(q_i, \delta_i)$$

has diameter less than ε at which stage any point of

$$[e_1, f_1] \times \cdots \times [e_n, f_n] \setminus \bigcup_{i=1}^m B(q_i, \delta_i)$$

can be returned as the required approximation of z . The hypothesis that z was not computable is thereby contradicted. \square

THEOREM 4. For any $\varepsilon > 0$ there exists a computable f on I^2 that fixes no computable point but the set of fixed points of which has Lebesgue measure at least $1 - \varepsilon$.

This is a straight-forward consequence of the fact that the intervals in Lemma 1 can be chosen to be arbitrarily small.

6. FURTHER QUESTIONS

Suppose x is a non-computable fixed point of a computable function f . By the preceding, it is not an isolated fixed point but it is necessarily an accumulation point of the non-computable fixed points?

On the other hand, can an accumulation point of non-computable fixed points be a computable (fixed) point?

7. CONCLUSION

The existence of the Kleene tree can quite easily be derived from the impossibility of ensuring the existence of a computable fixed point for a computable function (in both Russian and Polish senses), in two dimensions (or higher). The ingenious constructions of Orevkov and Baigger provide a way of defining a computable function with no computable fixed point from the set of intervals derived from the Kleene tree, in a constructive manner. This correspondence is, perhaps, more attractive for the “working mathematician” than the elegant derivation of the result in reverse mathematics. In one dimension, any computable $f : I \rightarrow I$ does have a computable point $x \in I_c$ such that $f(x) = x$, which can be seen by fairly straight-forward *reductio ad absurdum*.

Non-computable fixed points of computable functions appear in a relatively complicated way.

8. REFERENCES

- [1] Günter Baigger. Die Nichtkonstruktivität des Brouwerschen Fixpunktsatzes. *Archiv für Mathematische Logik und Grundlagenforschung*, 25(3-4):183–188, 1985.
- [2] V. Brattka, S. Le Roux, and A. Pauly. Connected Choice and the Brouwer Fixed Point Theorem. June 2012. arXiv:1206.4809B.
- [3] Janusz Brzdęk, Liviu Cădariu, and Krzysztof Ciepliński. Fixed point theory and the Ulam stability. *Journal of Function Spaces*, 2014:1–16, 2014.
- [4] Jeffrey L. Hirst. Notes on Reverse Mathematics and Brouwer’s Fixed Point Theorem. Available online: <http://www.mathsci.appstate.edu/~jlh/snp/pdfs/slides/bfp.pdf>, 2000.
- [5] S. C Kleene. Recursive functions and intuitionistic mathematics. pages 679–685, Providence, R. I., 1952. Amer. Math. Soc.
- [6] Joseph S. Miller. Degrees of unsolvability of continuous functions. *The Journal of Symbolic Logic*, 69(2):555–584, 2004.

- [7] V. P. Orevkov. A constructive map of the square into itself, which moves every constructive point. *Doklady Akademii Nauk SSSR*, 152:55–58, 1963.
- [8] Petrus H. Potgieter. Computable counter-examples to the Brouwer fixed-point theorem, 2008. arXiv:0804.3199.
- [9] Marian B. Pour-El and J. Ian Richards. *Computability in analysis and physics*. Perspectives in Mathematical Logic. Springer-Verlag, Berlin, 1989.
- [10] J. Rocha, B. Rzepka, and K. Sadarangani. Fixed point theorems for contractions of rational type with PPF dependence in Banach spaces. *Journal of Function Spaces*, 2014:1–8, 2014.
- [11] Naoki Shioji and Kazuyuki Tanaka. Fixed point theory in weak second-order arithmetic. *Annals of Pure and Applied Logic*, 47(2):167–188, 1990.
- [12] Kam-Chau Wong and Marcel K. Richter. Non-computability of competitive equilibrium. *Economic Theory*, 14(1):1–27, 1999.

Appendix: details for Section 3

The constructions should guarantee that at each stage, the function f_n moves every point of

$$D_n = \left(C_n^{\blacksquare 2^{-n}} \setminus C_n^{\blacksquare 2^{-n} \cdot \frac{5}{4}} \right)^\circ$$

in the direction of t_n by an amount proportional to its distance to $C_n^{\square 2^{-n}}$.

The construction of f_1 with this property is trivial. We proceed to construct f_{n+1} from f_n .

- (i) Extend and modify f_n to $C_{n+1}^{\blacksquare 2^{-n}}$ so that every point x of

$$\left(C_{n+1}^{\blacksquare 2^{-n}} \setminus C_{n+1}^{\blacksquare 2^{-n} \cdot \frac{5}{4}} \right)^\circ$$

is moved in the direction of t_n by an amount proportional to $d\left(x, C_{n+1}^{\square 2^{-n}}\right)$.

- (ii) Modify the resulting function so that each point in

$$C_{n+1}^{\blacksquare 2^{-n}} \setminus C_{n+1}^{\blacksquare 2^{-n} \cdot \frac{9}{8}}$$

is mapped a non-negative amount proportional to its distance to $C_{n+1}^{\square 2^{-(n+1)}}$ in the direction of t_n .

- (iii) By rotation of the direction of the mapping, extend the function to $C_{n+1}^{\blacksquare 2^{-(n+1)}}$ such that every point x of

$$D_{n+1} = \left(C_{n+1}^{\blacksquare 2^{-(n+1)}} \setminus C_{n+1}^{\blacksquare 2^{-(n+1)} \cdot \frac{5}{4}} \right)^\circ$$

is mapped in the direction of t_{n+1} by an amount proportional to $d\left(x, C_{n+1}^{\square 2^{-(n+1)}}\right)$.

The final step is the only one in which we use the fact that we are working in two dimensions as this step requires the continuous (computable) rotation of a vector in the direction of t_n to a vector in the direction of t_{n+1} .


A construction is given explicitly in [1] but it should be clear from the preceding that it can be done in many different

ways. The important part of the proof is that the construction is, at each stage, extended at the boundary to “look right” from the outside. This ensures that, eventually every point is in fact moved towards one of a sequence of points that converge to the non-computable fixed point (v, v) on the diagonal. The Baigger construction is a somewhat delicate construction of a function that is in fact computable but that—somehow—mimics a simple mapping of every point in I^2 in the direction of (v, v) .

Indeks avtorjev / Author index

Angyal Dávid	95
Bánhelyi Balázs	44
Békési József	48
Berczi Kristof	61
Brodnik Andrej	36
Čibej Uroš	87
Csaba Bela	68
Csehi Csongor Gy.	13
Dávid Balázs	9
Depolli Matjaž	40
Dobravec Tomaž	28
Dömösi Pál	24
Dosa Gyorgy	80
Ercsey Zsolt	20
Farkas Márk	13
Fernández Elena	51
Fürst Luka	87
Galambos Gábor	48
Gáll József	24
Gera Imre	44
Goossens Dries	72
Györfly Lajos	32
Gyori Ervin	91
Horváth Géza	24
Jovičić Vladan	36
Kellerer Hans	80
Kiraly Zoltan	61
Konc Janez	40
Kovács Zoltán	20
Laporte Gilbert	51
Lelkes Zoltán	16
Liu Changshuo	61
London András	32
Makay Géza	32
Mezei Tamas	91
Mihály Zsolt	16
Mihelič Jurij	87
Miklos Dezso	57
Miklós István	61, 76
Nagy Benedek	95
Palangetić Marko	36
Palfi Laszlo	5
Pekec Saša	54
Pjevalica Nebojša	5
Podgorelec David	83
Potgieter Petrus H.	99
Rodríguez Pereira Jessica	51
Silai Daniel	36
Špelič Denis	83
Spieksma Frits	72
Subotić Miloš	5
Szabo Sandor	40, 65
Tihanyi Norbert	24
Tóth Ádám	13
Tuza Zsolt	80
Vangerven Bart	72

Vasarhelyi Balint.....	68
Vincze Nándor	20
x 44	
Zavalnij Bogdan	40, 65



Konferenca / Conference

Uredili / Edited by

**Middle-European Conference on Applied
Theoretical Computer Science (MATCOS 2016)**

Prof. Andrej Brodnik