Zbornik 25. mednarodne multikonference

INFORMACIJSKA DRUŽBA

Zvezek I

Proceedings of the 25th International Multiconference

INFORMATION SOCIETY

Volume I

2022

Srednjeevropska konferenca o uporabnem teoretičnem računalništvu

Middle-European Conference on Applied Theoretical Computer Science

Uredniki • Editors:
Andrej Brodnik, Gábor Galambos, Branko Kavšek

Koper, Slovenija
13.-14. oktober
13-14 October
Koper, Slovenia

http://is.ijs.si

Zbornik 25. mednarodne multikonference

INFORMACIJSKA DRUŽBA – IS 2022

Zvezek I

Proceedings of the 25th International Multiconference

INFORMATION SOCIETY – IS 2022

Volume I

Srednjeevropska konferenca o uporabnem teoretičnem računalništvu
Middle-European Conference on Applied Theoretical Computer Science

Uredniki / Editors

Andrej Brodnik, Gábor Galambos, Branko Kavšek

http://is.ijs.si

13.-14. oktober 2022 / 13-14 October 2022
Koper, Slovenija

Uredniki:

Andrej Brodnik
Univerza na Primorskem in Univerza v Ljubljani

Gábor Galambos
Univerza v Szegedu

Branko Kavšek
Univerza na Primorskem

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 2022

Informacijska družba
ISSN 2630-371X

Kataložni zapis o publikaciji (CIP) pripravili v Narodni in univerzitetni knjižnici v Ljubljani
COBISS.SI-ID 127517187
ISBN 978-961-264-248-8 (PDF)

PREDGOVOR MULTIKONFERENCI INFORMACIJSKA DRUŽBA 2022

Petindvajseta multikonferenca *Informacijska družba* je preživela probleme zaradi korone. Zahvala za skoraj normalno delovanje konference gre predvsem tistim predsednikom konferenc, ki so kljub prvi pandemiji modernega sveta pogumno obdržali visok strokovni nivo.

Pandemija v letih 2020 do danes skoraj v ničemer ni omejila neverjetne rasti IKTja, informacijske družbe, umetne inteligence in znanosti nasploh, ampak nasprotno – rast znanja, računalništva in umetne inteligence se nadaljuje z že kar običajno nesluteno hitrostjo. Po drugi strani se nadaljuje razpadanje družbenih vrednot ter tragična vojna v Ukrajini, ki lahko pljuskne v Evropo. Se pa zavedanje večine ljudi, da je potrebno podpreti stroko, krepi. Konec koncev je v 2022 v veljavo stopil not raziskovalni zakon, ki bo izboljšal razmere, predvsem leto za letom povečeval sredstva za znanost.

Letos smo v multikonferenco povezali enajst odličnih neodvisnih konferenc, med njimi »Legende računalništva«, s katero postavljamo nov mehanizem promocije informacijske družbe. IS 2022 zajema okoli 200 predstavitev, povzetkov in referatov v okviru samostojnih konferenc in delavnic ter 400 obiskovalcev. Prireditev so spremljale okrogle mize in razprave ter posebni dogodki, kot je svečana podelitev nagrad. Izbrani prispevki bodo izšli tudi v posebni številki revije Informatica (http://www.informatica.si/), ki se ponaša s 46-letno tradicijo odlične znanstvene revije. Multikonferenco Informacijska družba 2022 sestavljajo naslednje samostojne konference:

- Slovenska konferenca o umetni inteligenci
- Izkopavanje znanja in podatkovna skladišča
- Demografske in družinske analize
- Kognitivna znanost
- Kognitonika
- Legende računalništva
- Vseprisotne zdravstvene storitve in pametni senzorji
- Mednarodna konferenca o prenosu tehnologij
- Vzgoja in izobraževanje v informacijski družbi
- Študentska konferenca o računalniškem raziskovanju
- Matcos 2022

Soorganizatorji in podporniki konference so različne raziskovalne institucije in združenja, med njimi ACM Slovenija, SLAIS, DKZ in druga slovenska nacionalna akademija, Inženirska akademija Slovenije (IAS). V imenu organizatorjev konference se zahvaljujemo združenjem in institucijam, 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.

S podelitvijo nagrad, še posebej z nagrado Michie-Turing, se avtonomna stroka s področja opredeli do najbolj izstopajočih dosežkov. Nagrado Michie-Turing za izjemen življenjski prispevek k razvoju in promociji informacijske družbe je prejel prof. dr. Jadran Lenarčič. Priznanje za dosežek leta pripada ekipi NIJZ za portal zVEM. »Informacijsko limono« za najmanj primerno informacijsko potezo je prejela cenzura na socialnih omrežjih, »informacijsko jagodo« kot najboljšo potezo pa nova elektronska osebna izkaznica. Čestitke nagrajencem!

Mojca Ciglarič, predsednik programskega odbora
Matjaž Gams, predsednik organizacijskega odbora

FOREWORD - INFORMATION SOCIETY 2022

The 25th *Information Society Multiconference* (http://is.ijs.si) survived the COVID-19 problems. The multiconference survived due to the conference chairs who bravely decided to continue with their conferences despite the first pandemics in the modern era.

The COVID-19 pandemic from 2020 till now did not decrease the growth of ICT, information society, artificial intelligence and science overall, quite on the contrary – the progress of computers, knowledge and artificial intelligence continued with the fascinating growth rate. However, the downfall of societal norms and progress seems to slowly but surely continue along with the tragical war in Ukraine. On the other hand, the awareness of the majority, that science and development are the only perspective for prosperous future, substantially grows. In 2020, a new law regulating Slovenian research was accepted promoting increase of funding year by year.

The Multiconference is running parallel sessions with 200 presentations of scientific papers at eleven conferences, many round tables, workshops and award ceremonies, and 400 attendees. Among the conferences, "Legends of computing" introduce the "Hall of fame" concept for computer science and informatics. Selected papers will be published in the Informatica journal with its 46-years tradition of excellent research publishing.

The Information Society 2022 Multiconference consists of the following conferences:

- Slovenian Conference on Artificial Intelligence
- Data Mining and Data Warehouses
- Cognitive Science
- Demographic and family analyses
- Cognitonics
- Legends of computing
- Pervasive health and smart sensing
- International technology transfer conference
- Education in information society
- Student computer science research conference 2022
- Matcos 2022

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 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.

The award for life-long outstanding contributions is presented in memory of Donald Michie and Alan Turing. The Michie-Turing award was given to Prof. Dr. Jadran Lenarčič for his life-long outstanding contribution to the development and promotion of information society in our country. In addition, the yearly recognition for current achievements was awarded to NIJZ for the zVEM platform. The information lemon goes to the censorship on social networks. The information strawberry as the best information service last year went to the electronic identity card. Congratulations!

Mojca Ciglarič, 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
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
Sergio Campos-Cordobes, Spain
Shabnam Farahmand, Finland
Sergio Crovella, Italy

Organizing Committee

Matjaž Gams, chair
Mitja Luštrek
Lana Zemljak
Vesna Koricki
Mitja Lasič
Blaž Mahnič

Programme Committee

Mojca Ciglarič, chair
Bojan Orel,
Franc Solina,
Viljan Mahnič,
Cene Bavec,
Tomaž Kalin,
Jozsef Györkös,
Tadej Bajd
Jaroslav Berce
Mojca Bernik
Marko Bohanec
Ivan Bratko
Andrej Brodnik
Dušan Caf
Saša Divjak
Tomaž Erjavec
Bogdan Filipič
Andrej Gams
Matjaž Gams
Mitja Luštrek
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
Boštjan Vilfan
Baldomir Zajc
Blaž Zupan
Boris Žemva
Leon Žlajpah
Niko Zimic
Rok Piltaver
Toma Strle
Tine Kolenik
Franci Pivec
Uroš Rajkovič
Borut Batagelj
Tomaž Ogrin
Aleš Ude
Bojan Blažica
Matjaž Kljun
Robert Blatnik
Erik Dovgan
Špela Stres
Anton Gradišek

KAZALO / TABLE OF CONTENTS

Zbornik 25. mednarodne multikonference

INFORMACIJSKA DRUŽBA – IS 2022

Zvezek I

Proceedings of the 25th International Multiconference

INFORMATION SOCIETY – IS 2022

Volume I

Srednjeevropska konferenca o uporabnem teoretičnem računalništvu
Middle-European Conference on Applied Theoretical Computer Science

Uredniki / Editors

Andrej Brodnik, Gábor Galambos, Branko Kavšek

http://is.ijs.si

13.-14. oktober 2022 / 13-14 October 2022
Koper, Slovenija

PREDGOVOR

Leta 2019 – ko smo organizirali 3. MATCOS konferenco – smo trdno verjeli, da smo vzpostavili tradicionalni dogodek v Kopru. V naslednjih letih smo bili osredotočeni na premagovanje covida. Ob tem smo spoznali možnosti uporabe domačih pisarne in sodelovanja na spletnih izvedbah konferenc. Vsemu navkljub smo prepričani, da ostaja konferenca, na kateri pride do osebnega stika in pogovora, nenadomestljiva.
Zato smo pričeli s pripravo naslednje konference MATCOS, 4. po vrsti. Pri organizaciji in izvedbi smo imeli srečo, da so člani tako organizacijskega kot programskega odbora večinoma sprejeli tudi letošnje povabilo ter izdatno prispevali k organizaciji in izvedbi konference.
Vabljeno predavanje je običajno eden osrednjih dogodkov konference. Letos bo to predavanje Györgyja Turána z University of Illinois at Chicago (USA) z naslovom »Interpretability of deep-learned error-correcting codes«. Predavanje nam bo posredovalo uvid v vpliv sodobne UI na načrtovanje klasičnih kod za popravljanje napak.
Poleg vabljenega predavanja bo na konferenci predstavljen še izbor člankov iz širokega področja računalništva in informatike vključno s primeri uporabe.
Tradicionalno prihaja na konferenco večina prispevkov in avtorjev iz Madžarske in Slovenije. Vendar je naš napor letos obrodil sad, saj so se jim na naše veliko zadovoljstvo pridružili avtorji še iz sedmih drugih držav in predstavili svoje delo.
Člani tako organizacijskega kot programskega odbora so v zadnjih nekaj mesecih opravili odlično delo. Zato vsem, ki so pomagali pri organizaciji in izvedbi konference MATCOS-22, iskrena zahvala.
Zaključujemo z željo, da boste te dni uživali v Kopru in da vzpostavite nove profesionalne stike na konferenci MATCOS-22.

V imenu organizatorjev
Andrej Brodnik in Gábor Galambos
sopredsedujoča

FOREWORD

In 2019 – when we organized the 3rd MATCOS conference – we strongly believed that we established a new tradition here in Koper. Then, the next few years we had to concentrate to win over the covid. We got acquainted with the possibilities of home offices and the organization of online conferences became current. But we are sure that a conference with personal interviews and discussions are irreplaceable.

So, this year we started to organize the next MATCOS conference, the 4th one. Fortunately, the former members of the Organizing Committee and the Program Committee accepted our invitation and took part actively in organization.

The invited talk is a central point while you organize a conference. This year György Turán from the University of Illinois at Chicago (USA) will present a talk on "Interpretability of deep-learned error-correcting codes", and so, we can take a look the influence of modern AI research to the design of classical error-correcting processes.

Selecting among the submitted papers we sorted out those ones that came from a wide range of the computer science and its applications.

Following the "traditions" most of the participants come from Hungary and Slovenia, but it is really a great pleasure to see that our efforts have been successful: the authors represent new research results from 7 countries.

The members of PC and OC did an excellent job during the last few months. Thanks to everybody who helped to organise the MATCOS-22.

We hope you will enjoy these days in Koper and you can establish new professional contacts during the MATCOS-22 conference.

On behalf of the organisers
Andrej Brodnik and Gábor Galambos
co-chairs

PROGRAMSKI ODBOR / PROGRAMME COMMITTEE

Andrej Brodnik, co-chair

Gábor Galambos, co-chair

Neil Hurley

Gabriel Istrate

Ivana Kolingerova

Miklós Krész

Ujjwal Maulik

Silvano Martello

Benedek Nagy

Rolf Niedermeier

Ion Petre

Ulrich Pferschy

Gerhard Reinelt

Giovanni Rinaldi

Borut Žalik

A Neural Network Based Classification Algorithm for Asthma Using Capnography

[Extended Abstract]

József Békési
Institute of Informatics
University of Szeged
Árpád tér 2.
H-6720 Szeged, Hungary
bekesi@inf.u-szeged.hu

Gábor Galambos
Juhász Gyula Faculty of Education, Department of Applied Informatics
University of Szeged
Boldogasszony sgt. 6.
H-6725 Szeged, Hungary
GalambosGabor@szte.hu

András Kelemen
Juhász Gyula Faculty of Education, Department of Applied Informatics
University of Szeged
Boldogasszony sgt. 6.
H-6725 Szeged, Hungary
kelemen.andras.felix@szte.hu

Imre Papp
Juhász Gyula Faculty of Education, Department of Applied Informatics
University of Szeged
Boldogasszony sgt. 6.
H-6725 Szeged, Hungary
pap.imre@szte.hu

József Tolnai
Albert Szent-Györgyi Medical School, Department of Medical Physics and Informatics
University of Szeged
Korányi fasor 9.
H-6720 Szeged, Hungary
tolnai.jozsef@med.u-szeged.hu

ABSTRACT

This article presents a neural network-based method to help physicians diagnose and monitor asthma and other chronic respiratory diseases. The method is based on capnography, using measurement data from a specially developed hand-held device.

After proper preparation, various parameters are calculated on the capnographic curve from which healthcare professionals can conclude the condition of the patient's respiratory system.

Another purpose of using the calculated parameters is to serve as a learning base for an artificial intelligence application that can be used in the decision support of physicians. The shape of the capnogram obtained from the gas sample exhaled by the patient and thus the parameters calculated from it are different for healthy people and those with respiratory diseases.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Decision Support; J.3 [**Computer Applications**]: LIFE AND MEDICAL SCIENCES—*Capnography*

General Terms

Applications

Keywords

Decision support, Neural networks, Capnography

1. INTRODUCTION

Capnography is a non-invasive method for the numerical and graphical analysis of exhaled CO_2 concentration. Time-based capnography is part of routine daily patient monitoring during mechanical ventilation and anesthesia. For spontaneously breathing patients, the method has the advantage that it does not require the patient to carry out any special breathing maneuvers, the measurement is easy to perform, and therefore requires minimal cooperation. It also holds the potential for the diagnosis of obstructive airway disease, as bronchospasm severity can be quantitatively assessed [4, 6]. The feasibility of non-invasive examinations is essential in pediatrics, so it also opens up new areas of application for capnography [7, 9, 10]. Although the analysis of capnogram shape parameters is not yet a standard part of patient monitoring, it appears promising in the monitoring of chronic respiratory diseases, as it provides useful information on the pathophysiological processes of pulmonary ventilation, such as airway patency and lung recoil tendency.

In capnographic studies, the carbon dioxide content of exhaled air can be considered as a function of time or plotted against the exhaled gas volume. In the former case, we are

talking about *time-based*, while in the latter case we are talking about *volumetric capnography*.

In the first part of the article, we examine the formal properties of time-based capnograms. Possible parameters describing the shape of the curve are presented. In the second part we introduce a neural network based method that uses these parameters to help physicians in diagnosing patients.

2. THE CAPNOGRAMS AND THEIR PARAMETERS

The capnogram curve plots the partial pressure of the CO_2 content of the exhaled gas against time or volume. The partial pressure of a given gas in a gas mixture is the pressure that a gas in question would create alone if it filled the available space alone. The partial pressure of CO_2 is denoted by PCO_2.

The capnogram consists of an exhalation segment and an inhalation segment. In this study we focused only on the shape indices of the exhalation section. The three phases of the exhalation segment (Phases 1-3) contain different slopes, angles and other parameters which are described in many articles and textbooks (e.g. [2, 3, 8]).

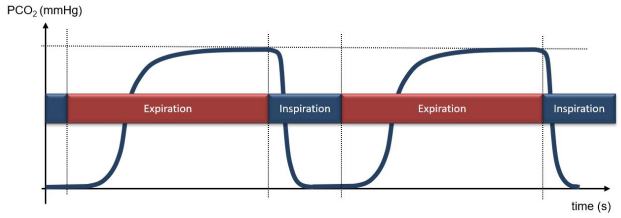

Figure 1: General form of time-based capnograms

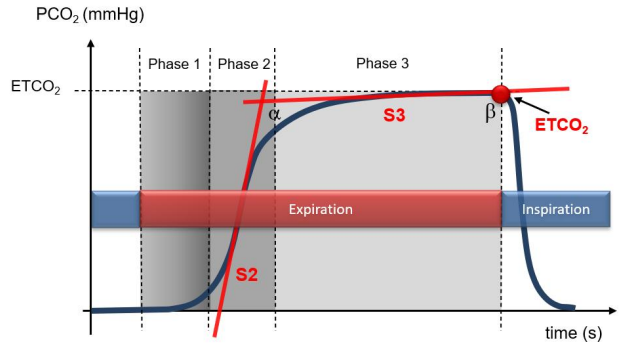

Figure 2: Phases of a capnogram with End-tidal $CO_2 (ETCO_2)$

2.1 The calculated parameters

The various morphological parameters are calculated using mathematical methods, which are presented in this subsection. The resulting capnographic indices - in the knowledge of the patients' condition - provide an opportunity to assess the characteristics of healthy and chronic respiratory patients (see [12] for more details). We aim to calculate these parameters as accurately and objectively as possible. This creates the opportunity to apply learning algorithms and automatically determine the condition of the patients studied. As a first step, faulty respiratory cycles were filtered out based on physiological rules that were supported by measurement techniques.The parameter calculator smoothed the points of the raw curve using the moving average method. In this case, each point was replaced by an average calculated from a specified number of adjacent points. For the 100Hz sampling frequency used for recording, we found the 9-point moving average to be the most suitable. Then, for each point of the smoothed curve, we calculated the first-order derivatives using the standard differential quotient. Since the curve containing the first derivatives can also be slightly noisy, we performed the previous smoothing algorithm for this as well. Then, following the same method, we calculated the curve containing the second derivative and its smoothed version. Finally, using the smoothed derivative 2 curve, the starting point of Phase 2 (local maximum) and the end point of Phase 3, i.e. the end of exhalation (local minimum) can be determined. It should be noted that the starting point of the exhalation cannot be precisely determined only from the time capnogram curve. However, before the start of Phase 2, we can find the point where the curve still takes approximately a value of 0, and then this point can be considered as the starting point of the fitting algorithm described below. We then fit a function to the exhalation sections obtained as previously described using the method introduced by Tusman et al. in [11]. The beginning of Phase 2 and the end of Phase 3 have already been determined as described above, and its post-fitting correction is not necessary. However, after fitting, the first, second, and third derivatives must be re-determined (now on the fitted curve). The end point of Phase 2 (the starting point of Phase 3) is obtained from the local maximum of the calculated third derivative.

2.1.1 *The slopes of Phases 2 and 3 (S2, S3)*

To determine the inflection point of Phase 2, we use the first-order derivative values, which mathematically represent the slope of the line drawn at a given point on the curve. The slope at the inflection point will be the largest. The slope of Phase 2 (S2) is the maximum slope that can be read at this inflection point [11]. The slope of Phase 3 (S3) is the slope of the line fitted to the middle third of Phase 3, which is a simplified but not significantly different modification of the method used by Tusman et al [11].

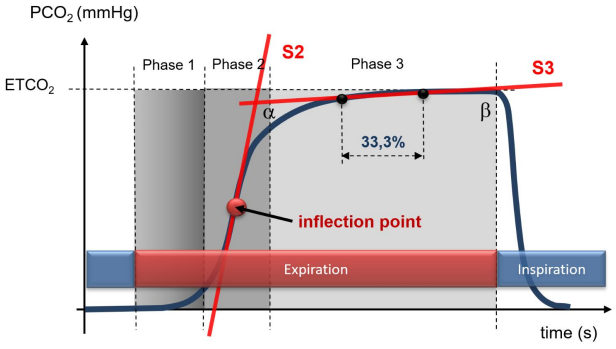

Figure 3: The slopes of time-based capnograms

2.1.2 *End-tidal CO_2 $(ETCO_2)$*

The carbon dioxide concentration increases throughout Phase 3, so it normally peaks at the end of the phase. This is the final exhalation CO_2 concentration ($ETCO_2$, $PETCO_2$), which is equal to the carbon dioxide partial pressure read at the end of Phase 3.

2.1.3 The normalized slopes of Phases 2 and 3 (Sn2, Sn3)

The normalized slopes of Phase 2 (Sn2) and Phase 3 (Sn3) are obtained by dividing the slopes of the second and third phases (S2, S3) by the value of $ETCO_2$.

2.1.4 Sn3/Sn2

The quotient of the Sn3 and Sn2 values.

2.1.5 D2min and D2max

The maximum and minimum of the second derivative, the rate of change of the start and end points of Phase 2 (the lower and upper curves).

2.1.6 The α angle (Q)

The angle enclosed by the slopes of Phases 2 and 3.

2.1.7 The area ratio (AR)

The area ratio in the section between the inflection point and the beginning of Phase 3 is the quotient of the area under the curve and the area of the entire rectangle. It is practically the shape of the transition from Phase 2 to Phase 3.

2.1.8 Squared difference (R2SUM)

The sum of the squares of the differences between the points of the raw, original curve and the fitted one. As previously described, the original capnogram curve contains higher frequency noises, which may have physiological reasons. Therefore, these sums of squares are used to examine the differences in the curves of the patients in each group.

2.1.9 Respiratory rates (RR)

In the absence of flow data, the exact length of respiratory cycles cannot be determined from the time capnogram alone. Thus, the length of the given respiratory cycle can be estimated from the combined length of Phases 2 and 3. Examining the measurements in parallel with the flow measurement, we found that the combined length of Phases 2 and 3 is about 65 percent of the respiratory cycle. Currently, we use this ratio to estimate respiratory length, from which we calculate the actual respiratory rate.

3. THE INPUT DATA AND THE STRUCTURE OF THE NETWORK

The data used for teaching the network were as follows:

- All time-based parameters calculated from mainstream measurements: S2T, S3T, $ETCO_2$, Sn2, Sn3, Sn3/Sn2, D2min, D2max, Q, AR, R2SUM, RR (Separate records for each breathing cycle).
- Gender of the patient.
- Class of the patient's age at the time of examination. (The patient's age was divided into 10-year-long classes. For example: 13 years old, 17 years old -> class: 1, 33 years old -> class: 3, 60 years old, 62 years old -> class: 6, etc. This was necessary because without classification only a few measurements would belong to some ages, which would impair the effectiveness of learning.)
- Class of the patient's body weight at the time of examination. (The patient's body weight was divided into classes of 10 kilograms, in the same way as for age.)

We used one label for teaching, which was a manual medical diagnosis of the patient for the test. (One test could include several measurements. One measurement could only belong to one test. One test could only have one diagnosis.) We only used measurements with a "healthy" or "asthmatic" diagnosis. We omitted from teaching the load measurements and the measurements marked as incorrect.

The method was implemented in Java and relied on the Deeplearning4j library [1]. The training of the neural network and the diagnosis prediction with the trained neural network ran on the following configuration: Intel Core i7 10700K CPU, 32GB DDR4 RAM, 256GB SSD, 2TB HDD, Nvidia GeForce 8500 GT video card.

The neural network had 3 hidden layers, each with 50 neurons. For each hidden layer, the activation function was the TANH function. The activation function of the output layer was the SIGMOID function. We gave 6000 epochs for teaching, but according to the log files, no significant learning took place after the 652nd epoch. The training was performed on a record of 3141 healthy and 16670 asthmatic breathing cycles, which lasted 2169 seconds on the configuration given above.

4. RESULTS

Since the training was done per respiratory cycle (the parameters are also calculated separately for each cycle), the diagnosis prediction with the trained neural network is also done per respiratory cycle. For each measurement, we calculated how many cycles of the measurement were "healthy" and how many cycles were "asthmatic". (The prediction is not performed for cycles marked as incorrect.) If the number of healthy predictions is lower than the number of asthmatic predictions, then the entire measurement is considered asthmatic. Otherwise, the entire measurement is considered healthy. The number of measurements used in the prediction was 648. Considering the "asthmatic" diagnosis as positive and the "healthy" diagnosis as negative we found the followings:

- True positive (TP): 517 (79.78%)
- True negative (TN): 107 (16.51%)
- False positive (FP): 23 (3.55%)
- False negative (FN): 1 (0.15%)

TP: The number of measurements for which the manual diagnosis of the test is "asthmatic" and the diagnosis obtained with the neural network is also "asthmatic". TN:

The number of measurements for which the manual diagnosis of the test is "healthy" and the diagnosis obtained with the neural network is also "healthy". FP: The number of measurements for which the manual diagnosis of the test is "healthy", but the diagnosis obtained with the neural network is "asthmatic". FN: The number of measurements for which the manual diagnosis of the test is "asthmatic", but the diagnosis obtained with the neural network is "healthy".

The metrics calculated from these are:

- Accuracy: 0.96,
- Precision: 0.96,
- Recall: 1.00,
- F1 Score: 0.98.

Here we used the usual metrics of classifiers, based on the following formulas [5]: Accuracy: (TP + TN) / (TP + FP + TN + FN) Precision: TP / (TP + FP) Recall: TP / (TP + FN) F1 score: 2* precision * recall / (precision + recall)

All of the above metrics must fall within the interval [0.0, 1.0]. The closer the value is to 1.0, the better the result. The total running time of the diagnosis prediction was 240 seconds for 1361 measurements, so the prediction takes an average of 0.1763 seconds per measurement. Comments:

1. The evaluation is somewhat distorted by the fact that we have fewer healthy subjects than asthmatics.

2. It is similarly distorted by the fact that we used all the measurements of all asthmatic and healthy tests from the database for teaching. This is due to the limited number of measurements. In the case of several measurements, we could use only a small part of the measurements during teaching, and test the neural network on the larger part. That way we would get more objective test results.

5. CONCLUSIONS

In this research we developed a neural network based application that uses capnography measurements to help the diagnosis of asthma. Possible future works are the followings:

1. Training the neural network with the raw measurement data as well, not only with the calculated parameters. This is expected to require more hardware resources and time. An advantage may be that the neural network can also learn useful information that is lost during the parameter calculation.
2. Training the neural network with the volumetric parameters or together with volumetric and time-based parameters. The disadvantage here may be that there are no volumetric parameters for purely time-based measurements without flow data.
3. Teaching the neural network for the different severities of asthma, and using the trained neural network to distinguish between them.
4. Teaching the neural network for other diseases, e.g. COPD (and its sub-conditions), ACOS (and its sub-conditions), COVID, etc. Distinguishing these diseases with the help of a trained neural network.

6. ACKNOWLEDGMENTS

This study was carried out in cooperation with PROFIT-EXPERT Ltd., University of Szeged, Bay Zoltán Nonprofit Ltd. for Applied Research, Optin Ltd. in the framework of the EU-funded Hungarian project "CAPNO - research on the application possibilities of capnography and development of an instrument for the diagnosis of asthma and other respiratory diseases (GINOP-2.2.1-15-2017-00046)."

7. REFERENCES

[1] Deeplearning4j Suite Overview. `https://deeplearning4j.konduit.ai/`. [Accessed 16-Jul-2022].

[2] K. Bhavani-Shankar, A. Y. Kumar, H. S. L. Moseley, and R. Ahyee-Hallsworth. Terminology and the current limitations of time capnography: A brief review. *Journal of Clinical Monitoring*, 11(3):175–182, May 1995.

[3] K. Bhavani-Shankar and J. H. Philip. Defining segments and phases of a time capnogram. *Anesthesia & Analgesia*, 91(4):973–977, Oct. 2000.

[4] J. B. Chambers, P. J. Kiff, W. N. Gardner, G. Jackson, and C. Bass. Value of measuring end tidal partial pressure of carbon dioxide as an adjunct to treadmill exercise testing. *BMJ*, 296(6632):1281–1285, may 1988.

[5] D. Chicco and G. Jurman. The advantages of the matthews correlation coefficient (MCC) over f1 score and accuracy in binary classification evaluation. *BMC Genomics*, 21(1), jan 2020.

[6] C. Chopin, P. Fesard, J. Mangalaboyi, P. Lestavel, M. C. Chambrin, F. Fourrier, and A. Rime. Use of capnography in diagnosis of pulmonary embolism during acute respiratory failure of chronic obstructive pulmonary disease. *Critical Care Medicine*, 18(4):353–357, apr 1990.

[7] N. Eipe and D. R. Doherty. A review of pediatric capnography. *Journal of Clinical Monitoring and Computing*, 24(4):261–268, jul 2010.

[8] J. S. Gravenstein, M. B. Jaffe, N. Gravenstein, and D. A. Paulus, editors. *Capnography*. Cambridge University Press, Mar. 2011.

[9] B. D. Guthrie, M. D. Adler, and E. C. Powell. End-tidal carbon dioxide measurements in children with acute asthma. *Academic Emergency Medicine*, 14(12):1135–1140, dec 2007.

[10] S. Kunkov, V. Pinedo, E. J. Silver, and E. F. Crain. Predicting the need for hospitalization in acute childhood asthma using end-tidal capnography. *Pediatric Emergency Care*, 21(9):574–577, sep 2005.

[11] G. Tusman, A. Scandurra, S. H. Böhm, F. Suarez-Sipmann, and F. Clara. Model fitting of volumetric capnograms improves calculations of airway dead space and slope of phase III. *Journal of Clinical Monitoring and Computing*, 23(4):197–206, June 2009.

[12] B. You, R. Peslin, C. Duvivier, V. D. Vu, and J. Grilliat. Expiratory capnography in asthma:. *European Respiratory Journal*, 7(2):318–323, Feb. 1994.

Online Bin Covering with Exact Advice*

Andrej Brodnik
University of Ljubljana
Slovenia
andrej.brodnik@upr.si

Bengt J. Nilsson
Malmö University
Sweden
bengt.nilsson.TS@mau.se

Gordana Vujovic
University of Ljubljana
Slovenia
gogili.vujovic@gmail.com

ABSTRACT

We show a 2/3-competitive strategy for the bin covering problem using $O(b + \log n)$ advice, where b is the number of bits used to encode a rational value and n is the length of the input sequence.

Categories and Subject Descriptors

500 [**Theory of computation**]: Online algorithms

1. INTRODUCTION

In the bin covering problem, we are given a set of items of different sizes in the range $]0, 1]$ and the goal is to find a maximum number of covered bins where a bin is covered if the sizes of items placed in it is at least 1. It has been shown that is NP-hard [1]. The covering problem has applications in various situations in business and in industry, from packing snack pieces into boxes so that each box contains at least its defined net weight, to such complex problems as redistribution tasks/items to a maximum number of factories/bins, all working at or beyond the minimal feasible level. The problem is, as mentioned, maximizing the number of covered bins, and is NP-hard [9]. The bin covering problem was studied in-depth in Assmann's Ph.D. thesis [2]. In the online version, items are delivered successively (one-by-one) and each item has to be packed, either in an existing bin or a new bin, before the next item arrives. The quality of online strategies is measured by their *competitive ratio*, the minimum ratio between the quality of the strategy's solution and that of an optimal one. The first known online strategy that has been proposed for the problem is *Dual Next Fit* (DNF), analogous to Next Fit for the bin packing problem. A competitive ratio of DNF is 1/2 proved by Assmann *et al.* [1]. A few years later, Csirik and Totik [8] prove that no online algorithm can achieve a competitive ratio better than 1/2. Further lower bounds ar given by Balogh *et al.* [3]. Thus, the only way to improve on the competitive ratio is to change the computational model. Boyar *et al.* [4] look at bin covering using extra advice provided by an oracle through an advice tape that the strategy can read. If the input sequence consists of n items, they show that with $o(\log \log n)$ bits of advice, no strategy can have better competitive ratio than 1/2. They also provide a strategy with $O(\log \log n)$ bits of advice having competitive ratio 8/15 and then show that a linear number of bits of advice is necessary to achieve competitive ratio greater than 15/16.

We show a 2/3-competitive strategy for the one-dimensional bin covering problem using $O(b + \log n)$ advice, where b is the number of bits used to encode a rational value in the input sequence and n is the length of the input sequence.

*This work is sponsored in part by the Slovenian Research Agency (research program P2-0359 and research projects J1-2481, J2-2504, and N2-0171).

2. PRELIMINARIES

The *online bin covering problem* we consider is, given an input sequence $\sigma = (v_1, v_2, \ldots)$, of rational values $v_i \in [0, 1]$, find the *maximum* number of unit sized bins that can be covered online with items from the input sequence σ. The bin covering problem is a dual version of the *bin packing problem*.

We define the *load* of a bin B to be

$$\mathrm{ld}(B) \stackrel{\text{def}}{=} \sum_{v \in B} v. \tag{1}$$

We can similarly define the load of a sequence σ to be $\mathrm{ld}(\sigma) \stackrel{\text{def}}{=} \sum_{v \in \sigma} v$.

A *covering* is a partitioning of the items into bins $B_1, B_2, \ldots$ such that for each bin B_j

$$\mathrm{ld}(B_j) \geq 1 \tag{2}$$

and our objective is to find the maximum number of bins that satisfy Inequality (2). In contrast to the bin packing problem, a strategy can open any number of bins at any time. However, only those that are filled to a load of at least 1 are counted in the solution.

We measure the quality of an online maximization strategy by its *competitive ratio*, the maximum bound R such that

$$\big|A(\sigma)\big| \geq R \cdot \big|\mathrm{OPT}(\sigma)\big| - C, \tag{3}$$

for every possible input sequence σ, where $A(\sigma)$ is the solution produced by the strategy A on σ, $\mathrm{OPT}(\sigma)$ is a solution on σ for which $|\mathrm{OPT}(\sigma)|$ is minimal, and C is some constant.

Of particular interest is the Dual Next Fit strategy (DNF), where DNF maintains one active bin B, and packs the items into B until it is covered. It then opens a new empty bin as

the active bin and continues the process. Assmann *et al.* [1] prove that DNF has a competitive ratio of 1/2 and Csirik and Totik [8] prove that no online algorithm can achieve a competitive ratio better than 1/2.

If we know some further structure of the input sequence, we can do slightly better as is shown in the next lemma that we will make extensive use of in the sequel.

LEMMA 1. *The online strategy* DNF *for the bin covering problem on an input sequence σ_α where the items have weights bounded by $\alpha < 1$ has cost*

$$\big|\mathsf{DNF}(\sigma_\alpha)\big| > \frac{1}{1+\alpha}\big|\mathrm{OPT}(\sigma_\alpha)\big| - \frac{1}{1+\alpha}.$$

PROOF. Assume that DNF opens $m+1$ bins when accessing the sequence σ_α, m of which are covered. Since every item has weight at most α, it means that each of the m covered bins are filled at most to a total weight of $1+\alpha$. A bin not obeying this limit would have been covered already before DNF places the last item in it, a contradiction. Thus the total load of the sequence σ is

$$(1+\alpha)m + 1 > \mathrm{ld}(\sigma_\alpha) \geq \big\lfloor \mathrm{ld}(\sigma_\alpha) \big\rfloor \geq \big|\mathrm{OPT}(\sigma_\alpha)\big|,$$

whereby $|\mathsf{DNF}(\sigma_\alpha)| = m > |\mathrm{OPT}(\sigma_\alpha)|/(1+\alpha) - 1/(1+\alpha)$ as claimed. □

Another strategy of interest is Dual Harmonic (DH_k), where the strategy subdivides the items by sizes into k groups,

$$]0, 1/k[,\ [1/k, 1/(k-1)[, \ldots, [1/3, 1/2[,\ [1/2, 1[,$$

and packs items in each group, maintaining k groups, according to DNF. Evidently, DH_k is at best 1/2-competitive using the same argument as in Csirik and Totik [8].

In certain situations, the complete lack of information about future input is too restrictive. In a sense, the online strategy plays a game against an all-powerful adversary who can construct the input sequence in the worst possible manner. To alleviate the adversary's advantage, we consider the following *advice-on-tape* model [6]. An *oracle* has knowledge about both the strategy and the full input sequence from the adversary, it writes information on an *advice tape* of unbounded length. The strategy can read bits from the advice tape at any time, before or while the requests are released by the adversary. The *advice complexity* is the number of bits read from the advice tape by the strategy. Since the length of the advice bit string is not explicitly given, the oracle is unable to encode information into the length of the string, thereby requiring some mechanism to infer how many bits of advice the strategy should read at each step. This can be done with a self-delimiting encoding that extends the length of the bit string only by an additive lower order term [5].

A bit string s is encoded as $e(s) = u(s) \circ b(s) \circ s$ ($\circ$ denotes concatenation), where $b(s)$ is a binary encoding of the length of the string s and $u(s)$ consists of $\big|b(s)\big|$ ones followed by a single zero, thus indicating how many bits the strategy needs to read in order to obtain the length of the string s. The encoding has length at most $\big|e(s)\big| = |s| + 2\lceil \log(|s|+1)\rceil + 1$. We henceforth assume that all advice information is encoded in this way. An integer m can thus be encoded exactly using $O(\log m)$ bits and a rational value m_e/m_d, where m_e and m_d are integers can be encoded using $O(\log m_e + \log m_d)$ bits. If the rational value lies in the interval $[0,1]$, then $m_e \leq m_d$ and the encoding can be made using $O(\log m_d)$ bits.

We will base our strategy on DH_k with added advice to improve on the competitive ratio, as do Boyar *et al.* [4].

3. AN EXACT ADVICE STRATEGY FOR BIN COVERING

Each item v corresponds to a rational value $0 < v < 1$, since any v above or equal to 1 will cover a bin and the optimum solution can be assumed to place v alone in a bin to cover it. Also, values of size 0 could be placed in the first covered bin without loss of generality.

Fix an integer $k \geq 2$. We will subdivide the set of items into k subsets, such that $1/t \leq v < 1/(t-1)$ for each integer $2 \leq t \leq k$, the *t-items*, and items $v < 1/k$, the *small items*.

Consider a fixed optimal covering $\mathrm{OPT}(\sigma)$ for the input sequence σ. We can partition the solution $\mathrm{OPT}(\sigma)$ into groups, $\mathcal{G}_{t_1 t_2 \cdots t_j}$, where the index $t_1 t_2 \cdots t_j$, with $2 \leq t_1 \leq t_2 \leq \cdots \leq t_j \leq k$, denotes that each bin in group $\mathcal{G}_{t_1 t_2 \cdots t_j}$ contains one t_1-item, one t_2-item, etc, multiplicity denoting the number of times each item type occurs in the bin in addition to the small items needed to fill it. We say that a bin in group $\mathcal{G}_{t_1 t_2 \cdots t_j}$ is *easy*, if $\sum_{t \in \{t_1, t_2, \ldots, t_j\}} 1/t \geq 1$ and we can assume without loss of generality that easy bins contain no small items. Furthermore, we assume that if the bins in $\mathcal{G}_{t_1 t_2 \cdots t_j}$ are easy, then any bin group $\mathcal{G}_{t_1 t_2 \cdots t_j t_{j+1}}$ is empty, if $t_1 t_2 \cdots t_j$ is a subsequence of $t_1 t_2 \cdots t_{j+1}$, as the t_{j+1}-item in a bin in $\mathcal{G}_{t_1 t_2 \cdots t_j t_{j+1}}$ can be moved to other bins while we still maintain coverage in the bin. We also say that a bin in $\mathcal{G}_{t_1 t_2 \cdots t_j}$ is a *gap bin*, if $\sum_{t \in \{t_1, t_2, \ldots, t_j\}} 1/(t-1) < 1$, as each of these bins must contain small items to the amount of more than $1 - \sum_{t \in \{t_1, t_2, \ldots, t_j\}} 1/(t-1)$ to be covered. Lastly, we denote the group of bins that are only covered by small items by $\mathcal{G}_S$.

As an example, $\mathcal{G}_{22}$ are those bins that each contain two 2-items (bins in $\mathcal{G}_{22}$ are easy), $\mathcal{G}_2$ are those bins that each contain one 2-item and some small items, and $\mathcal{G}_3$ are those bins that each contain one 3-item and some small items (bins in $\mathcal{G}_3$ are gap bins since they require small items to the amount of more than 1/2 to be covered).

The size of the optimal solution is given by

$$|\mathrm{OPT}(\sigma)| = \sum_{\forall t_1 t_2 \cdots t_j} |\mathcal{G}_{t_1 t_2 \cdots t_j}| + |\mathcal{G}_S|, \tag{4}$$

for all valid index combinations $t_1 t_2 \cdots t_j$.

We modify the Dual Harmonic strategy to operate on advice and describe this strategy, denoted DH_k^a, dependent on the parameter k, the number of item types used to partition the items into. The superscript a indicates that the strategy admits advice. Let $x_1, \ldots, x_n$, $n = |\sigma|$, be an ordering of the items in σ, such that $x_i \geq x_{i+1}$, for $1 \leq i \leq |\sigma|$. The oracle provides the strategy with an integer m and the value x_m through a self-delimiting encoding.

The strategy DH_k^a initially reads the parameters m and x_m and opens m bins that we call *critical bins* and that will each be covered with one of the m largest items of the input sequence σ together with small items. Initially, each critical bin is assumed to have a *virtual load* of x_m. When an item of size $\geq x_m$ is placed in a critical bin, its virtual load

is increased to the actual value of the item. The strategy further opens a *t-bin* for every item type $t \in \{2, \ldots, k\}$, and a *small bin* for the small items. As the next item v of the input sequence arrives, it is handled as follows:

1. if $x_m \leq v$, place v in the next critical bin that does not yet contain a large item and update the virtual load of the critical bin,
2. if $1/k \leq v < x_m$ is a t-item, place v in the corresponding t-bin using DNF. If the bin becomes covered, close it and open a new t-bin,
3. if $v < 1/k$ is small, place v in the next critical bin that does not contain small items up to a virtual load of at least 1 and update the virtual load of this critical bin. If all critical bins are filled up to a virtual load of 1, place v in the small bin using DNF. If the small bin becomes covered, close it and open a new small bin.

LEMMA 2. *Assume that the strategy DH_4^a has access to the exact values of m and x_m, then it has competitive ratio*

$$\big|\mathsf{DH}_4^a(\sigma)\big| \geq \frac{2}{3}\big|\mathrm{OPT}(\sigma)\big| - \frac{173}{60}$$

for serving any sequence σ of size n.

PROOF. Note that the number of t-items, for $t = 2$, 3, and 4, in the instance is

$$\begin{aligned} T_2 &= |\mathcal{G}_2| + 2|\mathcal{G}_{22}| + |\mathcal{G}_{23}| + |\mathcal{G}_{24}| + |\mathcal{G}_{233}| + |\mathcal{G}_{234}| + \\ &\quad + |\mathcal{G}_{244}|, && (5)\\ T_3 &= |\mathcal{G}_3| + |\mathcal{G}_{23}| + 2|\mathcal{G}_{33}| + |\mathcal{G}_{34}| + 2|\mathcal{G}_{233}| + |\mathcal{G}_{234}| + \\ &\quad + 3|\mathcal{G}_{333}| + 2|\mathcal{G}_{334}| + |\mathcal{G}_{344}| + 2|\mathcal{G}_{3344}|, +|\mathcal{G}_{3444}|, && (6)\\ T_4 &= |\mathcal{G}_4| + |\mathcal{G}_{24}| + |\mathcal{G}_{34}| + 2|\mathcal{G}_{44}| + |\mathcal{G}_{234}| + 2|\mathcal{G}_{244}| + \\ &\quad + |\mathcal{G}_{334}| + 2|\mathcal{G}_{344}| + 3|\mathcal{G}_{444}| + 2|\mathcal{G}_{3344}| + 3|\mathcal{G}_{3444}| + \\ &\quad + 4|\mathcal{G}_{4444}|. && (7) \end{aligned}$$

For each non-easy bin group $\mathcal{G}_2, \ldots, \mathcal{G}_{444}$ (there are eleven of them), let $S_{t_1 \cdots t_4}$ denote the weight of the small items that the optimum solution packs in the bins of group $\mathcal{G}_{t_1 \cdots t_4}$. In addition, we denote by $S_S = \sum_{B \in \mathcal{G}_S} \mathrm{ld}(B)$ the total load of the small items covering the bins in $\mathcal{G}_S$.

We consider first some arbitrary set of covered bins $\mathcal{G}$, where each bin only contains small items. Assume that these bins have a total load of $S = \sum_{B \in \mathcal{G}} \mathrm{ld}(B) \geq |\mathcal{G}|$ and that the input sequence restricted to these small items is σ_S. From Lemma 1 we have that

$$\big|\mathsf{DNF}(\sigma_S)\big| > \frac{4}{5}S - \frac{4}{5} \geq \frac{4}{5}\big|\mathcal{G}\big| - \frac{4}{5} \qquad (8)$$

We can analyze the competitive ratio of the critical bins by first considering a decreasing ordering of the bins $B_1, \ldots, B_{|\mathcal{G}_2|}$ in $\mathcal{G}_2$ by the weight of their 2-item, w_i. We let $u_i = \mathrm{ld}(B_i) - w_i$ be the weight of the small items in B_i, whereby $S_2 \geq \big(|\mathcal{G}_2| - m\big) \cdot \big(1 - w_m\big)$ for arbitrary choice of $m \leq |\mathcal{G}_2|$ since $B_{m+1}, \ldots, B_{|\mathcal{G}_2|}$, each contains at least $1 - w_m$ amount of small items; see Figure 1. The critical bins, C_i, $1 \leq i \leq m$, each contains one 2-item of weight a_i, a small item of weight z_i that was the last small item placed in C_i by our strategy, and small items to the weight of $y_i = \mathrm{ld}(C_i) - a_i - z_i$. Again, by construction, $y_i \leq 1 - w_m$ for each $1 \leq i \leq m$.

Consider next the gap bins in the optimal solution. These are the bins in groups $\mathcal{G}_3$, $\mathcal{G}_4$, $\mathcal{G}_{34}$, and $\mathcal{G}_{44}$. Each bin in these groups is guaranteed to have small items to the amount of at least 1/2, 2/3, 1/6, and 1/3, respectively. Thus, for each of those groups we have $S_3 \geq |\mathcal{G}_3|/2$, $S_4 \geq 2|\mathcal{G}_4|/3$, $S_{34} \geq |\mathcal{G}_{34}|/6$, and $S_{44} \geq |\mathcal{G}_{44}|/3$.

For each group of non-easy bins $\mathcal{G}_2, \ldots, \mathcal{G}_{444}$, let $\mathcal{I}_{t_1 \cdots t_4}(m) \subseteq \{1, \ldots, m\}$ be the set of indices i such that the last small element (of weight z_i) that was placed in critical bin C_i was placed by the optimal solution in a bin from bin group $\mathcal{G}_{t_1 \cdots t_4}$. Easy bins are assumed, without loss of generality, to not contain any small items. Also, let $\mathcal{I}_S(m) = \big\{1, \ldots, m\big\} \setminus \big(\bigcup_{t_1 \cdots t_4 \notin Easy} \mathcal{I}_{t_1 \cdots t_4}(m)\big)$ be the set of remaining indices. The possible values of m range between $0 \leq m \leq m^+ = \big\lfloor (|\mathcal{G}_2| - |\mathcal{I}_2(m^+)|)/2 \big\rfloor$, where m^+ is the largest integer such that $2m^+ + |\mathcal{I}_2(m^+)| \leq |\mathcal{G}_2|$, since the strategy needs to guarantee that it can cover all the critical bins.

The oracle reveals $m = m^+ = \big\lfloor (|\mathcal{G}_2| - |\mathcal{I}_2(m^+)|)/2 \big\rfloor$ and $x_m = x_{m^+}$, the m^{th} largest item in the input sequence σ, so our strategy constructs m critical bins, $\big\lfloor (T_2 - m)/2 \big\rfloor$ 2-bins, $\big\lfloor T_3/3 \big\rfloor$ 3-bins, $\big\lfloor T_4/4 \big\rfloor$ 4-bins, and some bins corresponding to the amount of unused small items, giving us

$$\begin{aligned} \big|\mathsf{DH}_4^a(\sigma)\big| &> m + \left\lfloor \frac{T_2 - m}{2} \right\rfloor + \left\lfloor \frac{T_3}{3} \right\rfloor + \left\lfloor \frac{T_4}{4} \right\rfloor + \frac{4}{5}\Bigg(S_S \\ &\quad + S_2 + S_3 + S_4 + S_{34} + S_{44} - \left(\sum_{i=1}^{m} y_i + z_i\right)\Bigg) - \frac{4}{5} \\ &\geq \frac{m}{2} + \frac{T_2}{2} + \frac{T_3}{3} + \frac{T_4}{4} + \frac{4}{5}\left(S_S - \sum_{i \in \mathcal{I}_S(m)} z_i\right) + \frac{4}{5}\left(S_3 - \sum_{i \in \mathcal{I}_3(m)} z_i\right) \\ &\quad + \frac{4}{5}\left(S_4 - \sum_{i \in \mathcal{I}_4(m)} z_i\right) + \frac{4}{5}\left(S_{34} - \sum_{i \in \mathcal{I}_{34}(m)} z_i\right) + \frac{4}{5}\left(S_{44} - \sum_{i \in \mathcal{I}_{44}(m)} z_i\right) \\ &\quad + \underbrace{\frac{4}{5}\left(S_2 - \sum_{i=1}^{m}(1 - w_m) - \sum_{i \in \mathcal{I}_2(m)} z_i\right)}_{\geq 0} - \frac{163}{60} \\ &\geq \frac{m}{2} + \frac{T_2}{2} + \frac{T_3}{3} + \frac{T_4}{4} + \frac{2}{3}\left(S_S - \sum_{i \in \mathcal{I}_S(m)} z_i\right) \\ &\quad + \frac{2}{3}\left(S_3 - \sum_{i \in \mathcal{I}_3(m)} z_i\right) + \frac{5}{8}\left(S_4 - \sum_{i \in \mathcal{I}_4(m)} z_i\right) \\ &\quad + \frac{1}{2}\left(S_{34} - \sum_{i \in \mathcal{I}_{34}(m)} z_i\right) + \frac{1}{2}\left(S_{44} - \sum_{i \in \mathcal{I}_{44}(m)} z_i\right) - \frac{163}{60} \\ &\geq \frac{m}{2} + \frac{T_2}{2} + \frac{T_3}{3} + \frac{T_4}{4} + \frac{2S_S}{3} - \frac{|\mathcal{I}_S(m)|}{6} + \frac{2S_3}{3} - \frac{|\mathcal{I}_3(m)|}{6} \\ &\quad + \frac{5S_4}{8} - \frac{5|\mathcal{I}_4(m)|}{32} + \frac{S_{34}}{2} - \frac{|\mathcal{I}_{34}(m)|}{8} + \frac{S_{44}}{2} - \frac{|\mathcal{I}_{44}(m)|}{8} - \frac{163}{60} \\ &\geq \frac{m}{2} + \frac{T_2}{2} + \frac{T_3}{3} + \frac{T_4}{4} + \frac{2S_S}{3} - \frac{m}{6} + \frac{|\mathcal{I}_2(m)|}{6} + \frac{|\mathcal{I}_3(m)|}{6} \\ &\quad + \frac{|\mathcal{I}_4(m)|}{6} + \frac{|\mathcal{I}_{34}(m)|}{6} + \frac{|\mathcal{I}_{44}(m)|}{6} + \frac{2S_3}{3} - \frac{|\mathcal{I}_3(m)|}{6} \\ &\quad + \frac{5S_4}{8} - \frac{5|\mathcal{I}_4(m)|}{32} + \frac{S_{34}}{2} - \frac{|\mathcal{I}_{34}(m)|}{8} + \frac{S_{44}}{2} - \frac{|\mathcal{I}_{44}(m)|}{8} - \frac{163}{60} \\ &> \frac{m}{3} + \frac{T_2}{2} + \frac{T_3}{3} + \frac{T_4}{4} + \frac{|\mathcal{I}_2(m)|}{6} + \frac{2|\mathcal{G}_S|}{3} + \frac{|\mathcal{G}_3|}{3} + \frac{5|\mathcal{G}_4|}{12} \\ &\quad + \frac{|\mathcal{G}_{34}|}{12} + \frac{|\mathcal{G}_{44}|}{6} - \frac{163}{60} \end{aligned}$$

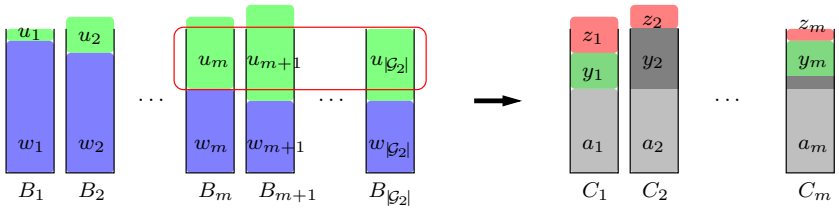

Figure 1: The critical bins and their relationship to the $\mathcal{G}_2$-bins in the optimal covering. In the $\mathcal{G}_2$-bins, blue are 2-items and light green are the small items. In the critical bins, red represents the last small items in the bin, dark green are the remaining small items, grey items are the 2-items, and dark grey represents the overlap between the virtual and actual load of the 2-item.

$$
\begin{aligned}
&= \frac{2|\mathcal{G}_2|}{3} + \frac{2|\mathcal{G}_3|}{3} + \frac{2|\mathcal{G}_4|}{3} + \frac{2|\mathcal{G}_{33}|}{3} + \frac{2|\mathcal{G}_{34}|}{3} + \frac{2|\mathcal{G}_{44}|}{3} + \frac{2|\mathcal{G}_S|}{3} \\
&\quad + \frac{3|\mathcal{G}_{24}|}{4} + \frac{3|\mathcal{G}_{444}|}{4} + \frac{5|\mathcal{G}_{23}|}{6} + \frac{5|\mathcal{G}_{344}|}{6} + \frac{11|\mathcal{G}_{334}|}{12} + |\mathcal{G}_{22}| \\
&\quad + |\mathcal{G}_{244}| + |\mathcal{G}_{333}| + |\mathcal{G}_{4444}| + \frac{13|\mathcal{G}_{234}|}{12} + \frac{13|\mathcal{G}_{3444}|}{12} + \frac{7|\mathcal{G}_{233}|}{6} \\
&\quad + \frac{7|\mathcal{G}_{3344}|}{6} - \frac{173}{60} \qquad \geq \qquad \frac{2}{3}|\mathrm{OPT}(\sigma)| - \frac{173}{60}
\end{aligned}
$$

bins, by applying Equalities (5)–(7) in the second to last step, while using that each $z_i < 1/4$, that $S_S \geq \sum_{i \in \mathcal{I}_S(m)} z_i$ and $S_{t_1 \cdots t_4} \geq \sum_{i \in \mathcal{I}_{t_1 \cdots t_4}(m)} z_i$, for each bin group $\mathcal{G}_{t_1 \cdots t_4}$, that critical bin C_i can be covered by a large item of size at least w_m plus the small items from a bin among the last bins $B_{m+1}, \ldots, B_{|\mathcal{G}_2|}$ in $\mathcal{G}_2$ and one extra small item from a non-easy bin in the optimal solution; see Figure 1, and that $m = |\mathcal{I}_S(m)| + \sum_{t_1 \cdots t_4} |\mathcal{I}_{t_1 \cdots t_4}(m)|$, for any m. The competitive ratio is the smallest coefficient of any of the terms corresponding to bin groups, since an adversary can ensure that the groups with larger coefficient contain no bins. This gives a competitive ratio of $2/3 \approx 0.6666\ldots$. □

For completeness sake we mention that using the same proof technique it is possible to show that $|\mathsf{DH}_2^a(\sigma)| \geq 3|\mathrm{OPT}(\sigma)|/5 - 19/15$, where $3/5 = 0.6$ and $|\mathsf{DH}_3^a(\sigma)| \geq 9|\mathrm{OPT}(\sigma)|/14 - 173/84$, where $9/14 \approx 0.64285\ldots$, if these strategies are given the exact values for m and x_m.

The two advice values $m \leq n$ and x_m can be represented by $O(\log n)$ bits and $O(b)$ bits respectively, where b is the number of bits required to represent the integer denominator of the rational value x_m, since $x_m < 1$. We have the following immediate theorem.

THEOREM 1. *The strategy DH_4^a receives $O(b + \log n)$ bits of advice and has competitive ratio*

$$|\mathsf{DH}_4^a(\sigma)| \geq \frac{2}{3}|\mathrm{OPT}(\sigma)| - \frac{173}{60}$$

for serving any sequence σ of size n, where b is the number of bits required to represent any rational value in σ.

One could venture to think that strategy DH_k^a, for $k > 4$, would give improved competitive ratio, or even that extending the strategy with more sets of critical bins could improve it further. However, this is not possible, since an adversary can simply provide an instance where all bin groups except $\mathcal{G}_2$ in an optimal solution are empty. Thus, the instance consists of only 2-items and small items. Any critical bin-based strategy must solve this instance and does so, even if the adversary provides all the small items first and the 2-items last, by choosing $m = \lfloor |\mathcal{G}_2|/3 \rfloor$, since the index set is $\mathcal{I}_2(m) = \{1, \ldots, m\}$, for all m, to guarantee that all critical bins are covered, thus opening m critical bins, and packing the remaining 2-items in pairs to cover $\lfloor (|\mathcal{G}_2| - m)/2 \rfloor$ bins. The strategy covers

$$
\begin{aligned}
m + \left\lfloor \frac{|\mathcal{G}_2| - m}{2} \right\rfloor &= \left\lfloor \frac{|\mathcal{G}_2|}{3} \right\rfloor + \left\lfloor \frac{|\mathcal{G}_2| - \lfloor |\mathcal{G}_2|/3 \rfloor}{2} \right\rfloor \\
&\leq \frac{|\mathcal{G}_2|}{3} + \frac{|\mathcal{G}_2|}{3} + \frac{1}{3} \leq \frac{2}{3}|\mathrm{OPT}(\sigma)| + \frac{1}{3} \qquad (9)
\end{aligned}
$$

bins, proving that our analysis in Lemma 2 is asymptotically tight.

4. REFERENCES

[1] Susan Fera Assmann, David S. Johnson, Daniel J. Kleitman, and Joseph Y-T. Leung. On a dual version of the one-dimensional bin packing problem. *Journal of Algorithms*, 5(4):502–525, 1984.

[2] Susan Fera Assmann. *Problems in discrete applied mathematics*. PhD thesis, Massachusetts Institute of Technology, 1983.

[3] János Balogh, Leah Epstein, and Asaf Levin. Lower bounds for online bin covering-type problems. *Journal of Scheduling*, 22(4):487–497, 2019.

[4] Joan Boyar, Lene M. Favrholdt, Shahin Kamali, and Kim S. Larsen. Online bin covering with advice. *Algorithmica*, 83(3):795–821, 2021.

[5] Joan Boyar, Shahin Kamali, Kim S. Larsen, Alejandro López-Ortiz. Online bin packing with advice. *Algorithmica*, 74(1):507–527, 2016.

[6] Hans-Joachim Böckenhauer, Dennis Komm, Rastislav Královič, Richard Královič, Tobias Mömke. On the advice complexity of online problems *Proc.* 20th *ISAAC*, LNCS 5878, pages 331–340, 2009.

[7] Edward G. Coffman Jr., Joseph Y-T. Leung, and D.W. Ting. Bin packing: Maximizing the number of pieces packed. *Acta Informatica*, 9(3):263–271, 1978.

[8] János Csirik and Vilmos Totik. Online algorithms for a dual version of bin packing. *Discrete Applied Mathematics*, 21(2):163–167, 1988.

[9] Michael R. Garey and David S. Johnson. *Computers and intractability*, volume 174. Freeman San Francisco, 1979.

Subsets without arithmetic subsequences: computational experiments and unsatisfiable cores

Uroš Čibej
University of Ljubljana
Faculty of Computer and Information Science
1000 Ljubljana, Slovenia
uros.cibej@fri.uni-lj.si

Ervin Győri
Alfréd Rényi Institute of Mathematics
Hungarian Academy of Sciences
H-1053 Budapest, Reáltanoda u. 13-15.
gyori@renyi.hu

ABSTRACT

A reduction to satisfiability of a combinatorial problem of minimal saturated subset without arithmetic subsequences is given in this paper. We conduct an empirical evaluation and present previously unknown optimal solutions for certain instances of the problem. The results also show where the limits for computing the optimal solutions are. Finally, we present a new possibility for solving such combinatorial problems, namely the unsatisfiable cores of the SAT expressions, which could give new insights to mathematicians and possibly new methods for solving the problem computationally.

Categories and Subject Descriptors

G.2.2 [**Mathematics of Computing**]: Discrete Mathematics—*combinatorics*

Keywords

satisfiability, solvers, modelling

1. INTRODUCTION

The problem of satisfiability has been at the center of computer science for more than half a century. It has been widely used to show some of the most important results in computational complexity and it is used as a showcase problem of a hard computational problem. But on the other hand, there has been a huge interest in developing better and better algorithms that can solve impressively large instances. The main driving force of this development is the annual SAT competition [5], spawning a huge research field and resulting in an enormous improvement in the speed and efficiency of these solvers in the past decade.

In this paper, we are exploring the possibilities of solving a set of instances of a hard combinatorial problem with the final goal of getting some new insights into the problem with such an empirical exploration. One way to implement this search could be to hand-tailor an efficient solver for this particular problem, but from our experience, we established that it takes a huge effort to construct a competitive solver which could outperform a state-of-the-art SAT solver. So instead of putting the effort into a custom-made solver, we construct a reduction to SAT and explore where are the limits of this approach.

The paper is structured as follows. The next section gives a short definition of the combinatorial problem and the reduction of this problem to SAT. Furthermore, we show what the reduced expressions look like for the instances that we are interested in. Section 3. describes the computational results, namely what are the optimal solutions for our problem and what are the times required to obtain them. This will give us some insight into what are the feasible sizes that can be solved with any kind of solver. Section 4. describes an interesting concept that can be obtained with SAT solvers and could be a new approach to solving this problem, either theoretically or empirically.

2. PROBLEM DEFINITION AND REDUCTION TO SAT

2.1 Problem definition

Definition 1 (Arithmetic triple)**.** *A set $\{a, b, c\}$ (assuming $a < b < c$) is an arithmetic triple if $b - a = c - b$.*

The set of all arithmetic triples of a set A will be denoted as

$$arith(A) = \{\{a, b, c\} \subseteq A | \{a, b, c\} \text{ is an arithmetic triple}\}$$

.

Definition 2 (Non-arithmetic set)**.** *A set A is said to be non-arithmetic if $arith(A) = \emptyset$.*

With these two definitions in place, we can define our optimization problems.

Definition 3 (MinNArith)**.** *Given a set $A = \{1 \dots n\}$, what is the smallest non-arithmetic subset $A' \subseteq A$, which is also maximal (or saturated), i.e. it is not possible to enlarge this set without creating an arithmetic triple, formally:*

$$\forall i \in A \setminus A' : arith(A' \cup \{i\}) \neq \emptyset$$

In our case, we are mostly interested in the size of such a set, given a particular value of n.

Example:. Let us say the set $A = \{1, \ldots, 16\}$, one saturated set that does not contain any arithmetic triple is $A' = \{6, 7, 10, 11\}$. It can be easily checked that by adding any other number from A, we create an arithmetic triple. We will see that this is also the smallest such subset of A.

What is known about this problem? [7, 6]
1. it seems to be a hard mathematical problem
2. for $n = 4^k$ there exists a construction of such a set of size $\sqrt{n}$,i.e., 2^k
3. it is conjectured that this is the smallest possible such set
4. It has been proven that this is indeed the case for $n = 4, 16,$ and 64.

The goal of this work is to push this boundary further, as much as possible, with the final goal to explore the feasibility of computing the size of the minimal set for $n = 256$. Since we already know such a set of size 16, the goal is to prove that there is no such set of size 15.

2.2 Reduction to SAT

In principle, this is a minimization problem, but we will use a decision version of this problem to reduce to SAT.

Definition 4 (MinNArithDEC). *Given a set $A = \{1, \ldots, n\}$ and a number k, does a saturated non-arithmetic subset $A' \subseteq A$ exist, such that $|A'| \leq k$.*

In what follows, we give a reduction $MinNArithDEC \rightarrow SAT$. The input to this problem are two numbers n and k, the reductions construct a logical expression (in CNF form) that is satisfiable if there exists a saturated non-arithmetic subset of $A = \{1 \ldots n\}$ with the size $\leq k$.

Each element in $i \in A$ has a corresponding logical variable x_i, which is true if $i \in A'$. To describe the problem $MinNArithDEC$, we introduce three types of constraints on this set of logical variables.

1. Cardinality constraint, i.e., at most k of the variables x_i can be true. We will denote this constraint as $CARD$.
2. In order for the set of true variables to describe a non-arithmetic subset, we have to check every arithmetic triplet $\{a, b, c\} \in arith(A)$ and assure that not all three elements are in A':
$$\overline{x_a \wedge x_b \wedge x_c}$$
. We will denote this type of constraint as $NARITH$.
3. And the third type of constraints assure the saturation of the set A'. For every element, $a \in A$, either a is already in the chosen subset, or there exists an arithmetic triplet where both of the other two elements are in the set. We will denote this type of constraint as $SATUR$.

The final expression is the conjunction of these three constraints:

$$E(n, k) = CARD \wedge NARITH \wedge SATUR$$

The $NARITH$ constraints have already been described (if we apply DeMorgan's rule it is already in CNF form), now we need to define more precisely how $CARD$ and $SATUR$ constraints are written as logical expressions.

Cardinality constraints. The problem of encoding cardinality constraints in SATs is a research topic on its own, and many different approaches are known. The main issue we are addressing in this problem is how to enforce that at most a certain number of variables are set to true. The most researched version of this problem is known as at-most-one (AMO) constraint [9]. Generalizations (at-most-k constraints) have been explored and are now part of most standardized SAT modeling toolboxes. It is also well-known [12] that different encodings can have varying effects on the execution times of different solvers.

In order to explore this impact, we tested 4 different standard encodings: Sequential counting [10], Sorting network [2], Cardinality network [1], and k-modulo totalizer [8].

Saturation constraints. For each element $i \in A$, we have to express that either it is in the set A', or that there exists an arithmetic triple $\{i, j, k\} \in arithA$ such that $j, k \in A'$.

This is straightforwardly transformed into a logical expression:

$$x_i \vee (\bigvee_{(i,j,k) \in arith(A)} (x_j \wedge x_k))$$

However, there is a technical difficulty with this expression. Namely, it is not in CNF, which is a typical requirement of SAT solvers. A straightforward transformation to CNF would result in exponentially large expressions, which would quickly make them practically unusable. Luckily, there exists a transformation, called the Tseitin transform [11], that transforms any logical expression into CNF, and the final expression is linear in the size of the original expression. The downside of this transformation is that also introduces a linear number of new variables.

2.3 Sizes of instances

To give an impression of the sizes of the reduced problem, the number of clauses and the number of variables for different n and k are given in Table 1.

Two different parameters describe the size of the expression, those are the number of variables and the number of clauses in the expression.

The table gives these two sizes for three different expressions, $E(16, 3), E(64, 7), E(256, 15)$. These are the expressions at $n = 4^k$ since we are testing the hypothesis that the solution at this n is 2^k and these expressions should all be unsatisfiable based on this hypothesis. We also show the difference in sizes if different cardinality encodings are used. We can see that the encoding does not greatly influence the entire size of the expression, since the largest difference is only a few percent.

Table 1: Sizes of reduced problems

n, k	CC	# of vars	# of clauses
(16,3)	seqcount	391	1168
	cardnet	440	1213
	sortnet	416	1265
	kmtotalizer	397	1182
(64,7)	seqcount	6415	19760
	cardnet	6600	19789
	sortnet	7102	20542
	kmtotalizer	6251	19598
(256,15)	seqcount	101407	316576
	cardnet	101248	314305
	sortnet	105470	320638
	kmtotalizer	98712	312147

But that does not describe the entire picture of the structure of these expressions. A more detailed view is given in Figure 1 which shows the percentage of clauses and new variables introduced by each type of constraint. The first graph shows how the ratio of the three types of clauses. It can be seen that the $SATUR$ constraint contributed to the vast majority of clauses. As n grows, the dominance of the $SATUR$ clauses gets even bigger. A similar situation can be seen on the right graph that shows the ratio of the newly introduced variables. First, notice that $NARITH$ constraints do not introduce any new variables and are thus omitted from this graph. Again, the vast majority of newly introduced variables originate from the $SATUR$ constraints.

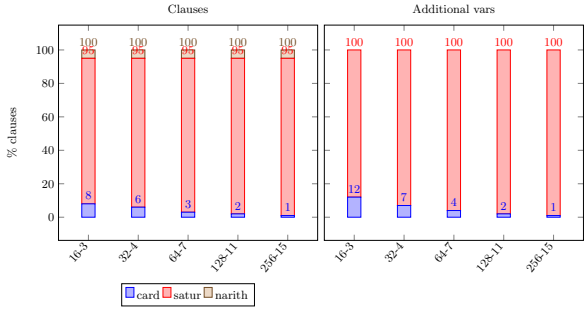

Figure 1: The structure of the SAT expression.

3. COMPUTATIONAL RESULTS

These generated SAT expressions have been given to the state-of-the-art solver. We used currently the fastest solver kissat [3], which won the 2020 and 2021 SAT competitions.

The minimal size of the set is obtained by finding a number k, such that $E(n, k)$ is satisfiable and $E(n, k-1)$ is unsatisfiable. In the evaluation description, we will be mostly describing the unsatisfiable expressions, since these expressions are usually much harder for the SAT solvers.

Figure 2 shows the sizes of minimal saturated subsets. The black plot is $\sqrt{n}$, i.e., the hypothesized lower bound. The results give further empirical evidence for this hypothesis to be true since the $\sqrt{n}$ lower-bound is reached only at points $n = 4^k$, which makes it more plausible that the proposed construction of such sets is also optimal.

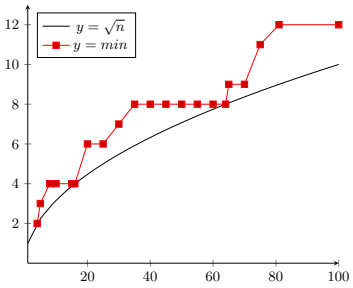

Figure 2: The plot of the minimal saturated set sizes. The black plot shows the lower bound which was reached only at the points $n = 4^k$.

An even more important aspect of our empirical test is the running times to prove the optimality for a certain n. Figure 3 shows the measurements for 4 different cardinality constraint encodings. The first chart gives the running times (y axis is log scale) and it can be seen that all four encodings follow the same trend. Nonetheless, some differences cannot be seen on this log-plot, so we show the results for $E(75, 10)$ and $E(81, 11)$ which shows that that k modulo totalizer has a significant advantage over the other three encodings.

4. UNSATISFIABLE CORES

Based on the results described above, it seems unlikely that we could solve the expression $E(256, 15)$ in a reasonable amount of time. This also gives little hope for solving the problem using some other reductions or even hand-crafted solvers for this particular problem.

However, there could be a different path, using the byproduct of SAT solvers. Namely, these solvers produce proof that their claim (satisfiable or unsatisfiable) is true. For satisfiable expressions, the proof is simply the variable assignment that yields a true value of the expression. But for unsatisfiable expressions, the proof is the trace of the execution of the solver.

The trace of the execution is very large, but a more compact proof can be extracted from it, and that is the unsatisfiable core [4] of the expression. Unsatisfiable cores are subexpressions that are unsatisfiable, but if we remove any clause, the expression becomes satisfiable. This can thus be viewed as the core "reason" why this expression is unsatisfiable.

As an example, let us examine the unsatisfiable core of the expression $E(16, 3)$. The entire expression has 397 variables and 1182 clauses (k modulo totalizer), but the unsatisfiable core has only 399 clauses. We can brake down the structure of the unsatisfiable core even further:

- original size of the $CARD$ expression is 102 and the core contains only 73,
- $NARITH$ has 56 clauses, but the core only contains 19 of these,
- and $SATUR$ has 1024 clauses, but the core only 307.

These results show that there is a more compact reason for

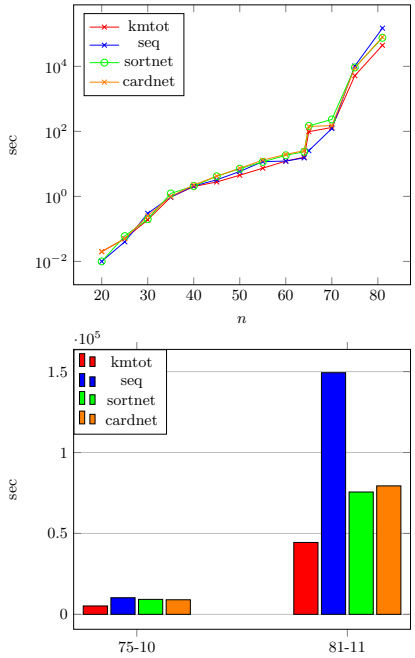

Figure 3: Run times for solving the problem $E(n, k)$, by varying n and k being the largest value where the expression is unsatisfiable. The left plot shows the log plot of the time for four different encodings of the cardinality constraint. The right plot shows only the times for $E(75, 10)$ and $E(81, 11)$ to show a more significant difference between different encodings.

the impossibility of the existence of a saturated set of size 3. There are two possible usages of these unsatisfiable cores:

- reverse engineering the unsatisfiable core to obtain the rules and find a general pattern and thus the proof for the lower bound as it is currently hypothesized.
- knowing the shape of the unsatisfiable cores, we could solve the problem by explicitly finding such a substructure in the entire expression, e.g. $E(256, 15)$. This would require a search for a known substructure, which is a computationally simpler problem than proving that a certain structure does not exist.

5. CONCLUSIONS

In this paper, we described a reduction of the problem of finding a minimal saturated non-arithmetic subset. This is a hard mathematical problem and empirical results could shed some new light on this problem, giving potential insights for future exploration of the problem.

The current hypothesis is, that the lower bound for the size of such a set is $\sqrt{n}$ and it can be reached for values $n = 4^k$. Our initial hope was to be able to reach $n = 256$, where a known set is of size 16 and we need to prove or disprove that this is the smallest possible one.

Using state-of-the-art solvers, we attacked this problem and obtained optimal solutions for values up to 100. All the results are in favor of the current hypothesis, but there is also a pessimistic result that it seems unlikely to obtain optimal results for a much larger n without a significant new insight into this problem.

But the reduction to SAT also has another potentially useful side-effect, namely the unsatisfiable cores. These subexpressions that can be obtained by SAT solvers carry some new insight into the problem and the study of their structure and reverse-engineering the rules for their creation might lead to discoveries in this and maybe other similar problems as well.

6. REFERENCES

[1] R. Asín, R. Nieuwenhuis, A. Oliveras, and E. Rodríguez-Carbonell. Cardinality networks and their applications. In O. Kullmann, editor, *Theory and Applications of Satisfiability Testing - SAT 2009*, pages 167–180, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[2] K. E. Batcher. Sorting networks and their applications. In *Proceedings of the April 30–May 2, 1968, Spring Joint Computer Conference*, AFIPS '68 (Spring), page 307–314, New York, NY, USA, 1968. Association for Computing Machinery.

[3] A. Biere, K. Fazekas, M. Fleury, and M. Heisinger. CaDiCaL, Kissat, Paracooba, Plingeling and Treengeling entering the SAT Competition 2020. In T. Balyo, N. Froleyks, M. Heule, M. Iser, M. Järvisalo, and M. Suda, editors, *Proc. of SAT Competition 2020 – Solver and Benchmark Descriptions*, volume B-2020-1 of *Department of Computer Science Report Series B*, pages 51–53. University of Helsinki, 2020.

[4] A. Cimatti, A. Griggio, and R. Sebastiani. A simple and flexible way of computing small unsatisfiable cores in sat modulo theories. In J. Marques-Silva and K. A. Sakallah, editors, *Theory and Applications of Satisfiability Testing – SAT 2007*, pages 334–339, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[5] N. Froleyks, M. Heule, M. Iser, M. Järvisalo, and M. Suda. Sat competition 2020. *Artificial Intelligence*, 301:103572, 2021.

[6] E. Győri. personal communication, 2022.

[7] S. J. Miller. Combinatorial and additive number theory problem sessions: '09–'19, 2014.

[8] A. Morgado, A. Ignatiev, and J. Marques-Silva. Mscg: Robust core-guided maxsat solving. *J. Satisf. Boolean Model. Comput.*, 9:129–134, 2014.

[9] V.-H. Nguyen, V.-Q. Nguyen, K. Kim, and P. Barahona. Empirical study on sat-encodings of the at-most-one constraint. In *The 9th International Conference on Smart Media and Applications*, pages 470–475, 2020.

[10] C. Sinz. Towards an optimal cnf encoding of boolean cardinality constraints. In P. van Beek, editor, *Principles and Practice of Constraint Programming - CP 2005*, pages 827–831, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[11] G. S. Tseitin. On the complexity of derivation in propositional calculus. In *Automation of reasoning*, pages 466–483. Springer, 1983.

[12] N.-F. Zhou. Yet another comparison of sat encodings for the at-most-k constraint, 2020.

Exact time measuring challenges

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

ABSTRACT

In this paper, we focus on implementations of the BubbleSort algorithm in three different programming languages: Java, C, and x86 assembler. Using the ALGator system we execute these implementations with different inputs and perform an empirical evaluation of the results. We discuss the importance of test repetition for achieving accurate timing results. We show that the Java and the C implementations achieve similar efficiency and that the quality order depends on the type of input data.

Categories and Subject Descriptors

F.2 [**Analysis of algorithms and problem complexity**]: Reliability and Testing; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

General Terms

Algorithm Engineering

Keywords

empirical algorithm analysis, time measuring, accuracy, reliability, comparing Java and C

`This work is sponsored in part by the Slovenian Research Agency (research project N2-0171).`

1. INTRODUCTION

The theoretical complexity analysis of algorithms is a very important part of the algorithm design process. This analysis usually estimates the amount of resources (e.g. time or memory storage) that is going to be utilized during the algorithm execution [4]. The result of the analysis depends on the selected computation model [3] which implies the execution environment and its limitations. Roughly, the results of such theoretical analysis are used to distinguish between fast (i.e. polynomial) and slow (exponential) algorithms. But in practice these results are of limited value especially if the two algorithms compared have the same (theoretical) time complexity. The model that is used in theoretical study does usually not take into account all the peculiarities of the real execution environment (like memory caching, paging, branch prediction, etc.), which are revealed only during the execution of the algorithm on a real computer. Therefore, for practical comparison of actual algorithms' capacity theoretical analysis has to be replaced with empirical measurements of resource usage during the algorithms' executions on various types of input data [5]. In order to provide quality results, these measurements have to be performed carefully since many factors impact the measured data. In this paper we focus on some of them and present results of our measurements that highlight the importance of each of them. Namely, we use three programming languages and present the impact of language selection on the speed of execution. Furthermore, we present the importance of repetition of tests, especially when the size of the input (and therefore the execution time) is small. In addition, we discuss how the type of input data can affect the algorithms' quality rankings.

2. TESTING ENVIRONMENT SETUP

For all our tests in this research we will use the BubbleSort [1] algorithm for sorting arrays of integers. Since this is a very well known and a simple algorithm we are able to perform a precise theoretical analysis and provide very accurate (theoretical) forecast for the time complexity of its implementations. The algorithm is so simple that we can count the number of operations performed during the execution for different inputs. Thus we will be able to compare theoretical predictions with the empirical results.

One of the goals of this research was to analyse the impact of the selected programming language on the efficiency of algorithm execution. Therefore we used three programming languages (namely the Java, the C and the x86 assembler) to implement BubbleSort. Due to the simplicity of the algorithm we managed to write the three implementations in such a way that they provide semantically identical code. For further reference, we named implementations `BubbleJ`, `BubbleC` and `BubbleA`, where the last letter denotes the programming language used (J for Java, C for C and A for x86 assembler). When executing these implementations on the same inputs they will perform the same number of each programming-language-dependant atomic operations. Any differences in the execution speed will thus reflect the differences in the execution speed of these operations in the selected programming language. The C implementation was compiled with the `gcc` compiler in two ways: without op-

timization (the `-O0` flag) and with full optimization (the `-O3` flag). In this way we got two distinct implementations (namely `BubbleC0` and `BubbleC3`). In the following we will analyze the impact of this optimization to the speed of execution.

To facilitate the empirical evaluation in our research we used the ALGator system [2]. We used its tools to configure the Sorting project, to provide the test sets of input data and implementations and to execute the algorithms' implementations in a controlled environment. For the execution machine we used the Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz computer with 32GB RAM and with the Linux Ubuntu operating system installed.

The inputs for our algorithms consist of arrays of integers prearranged in three different orders: random order (RND), sorted order (SOR) and inversely sorted order (INV). These three distributions of input data are well manageable from a theoretical point of view, since we know for all three the number of operations that will be performed during the sorting process. In all three case BubbleSort will perform exactly $n(n-1)/2$ comparisons, and $n^2/4$, 0, $n(n-1)/2$ swaps for RND, SOR, INV respectively. Note that all the numbers of operations are exact, except for the number of swaps in RND case - here we only have the expected (instead of exact) number of swaps, since the sequence is randomly mixed. Since BubbleSort performs only comparisons and swaps (and some auxiliary increments of indices to maintain the loops) we could expect that, for example, sorting RND array will be faster than sorting INV array of the same size. But as we will see in the following this in not the case.

In the ALGator project inputs (i.e. test cases) are grouped into test sets. Each test case has its own identifier (Test ID), so the results can also be compared on the test-basis. To provide accurate results each test case is executed several times (each execution of the test case has its identifier, Repetition ID). Besides a list of all execution times of a test case ALGator provides two information, the time of the first execution (Tfirst) and the time of the fastest execution (Tmin) of this test case. The first execution is usually much slower than other executions since (this is true especially for the java environment) the execution machine needs to warm up. As we will see in the following the Tfirst time can even be twice as big as the Tmin time.

To measure the time in Java we can only use the wall-clock (Java does not provide any processor usage information). To minimize the unreliability of the measured time (which is due to the fact that the process may spend time waiting for I/O or for other processes that are also using the CPU) we use a "clean" computer which is dedicated only for execution of the algorithms. Besides that, we usually take the Tmin time as a reference data, since this is a time in which the computer is capable of solving the problem (the number of disturbing factors is minimal). For he algorithms implemented in the C programming language we use the CPU time obtained by the `clock()` function (which returns the number of clock ticks used by the process). By calling this function before and after the algorithm execution and subtracting the returned values we get the total amount of time a process has actively used a CPU. The time measured this way is much more reliable and accurate quality indicator.

3. THE MEANING OF TEST REPETITION

In our first experiment we would like to find out the meaning of several repetitions of a given test case execution. For this we used a test set consisting of three groups of test cases: in each group there are 50 identical tests of sizes 500, 5000 and 20000. All the input arrays in these test cases were ordered in inverse order (to ensure the identical number of operations during the sort process). We executed each test case 50 times. By analysing the results we noticed that the (absolute) difference between Tfirst and Tmin is approximately the same for all three groups of test cases. The relative difference is therefore smaller for bigger measured times. We can conclude that the measurement of both Tfirst and Tmin is important for small inputs and that the importance of distinguishing between Tfirst and Tmin decreases with increasing input size. Measurements have shown that something similar to the Java's "Tfirst phenomenon" also happens with C, except that in this case "warming up the machine" adds significantly less to the overall time complexity, so the differences in speed between Tfirst and Tmin are noticeable only in experiments that take very little time. From Table 1, which shows the relationship between the average first and the minimum execution time of a test case, $f = \frac{\overline{T}_{first}}{T_{min}}$ it can be seen that for small n the ratio is similar in both implementations, but for larger n the difference between Tfirst and Tmin is almost negligible for the `BubbleC3`, while for the `BubbleJ` the value decreases significantly more slowly. At $n = 20000$ the difference is still more than 5%.

The difference in measured times of multiple executions of the `BubbleJ` and `BubbleC3` implementations is depicted in Fig. 1. Here we used 50 inversely ordered arrays of size 5000, each test case was repeated 50 times. On the graph, the time of the first execution of the test case is shown in gray (Tfirst, Repetition ID=0), the first 20 repetitions are shown in orange, and the next 30 in red. With `BubbleJ`, we see that the first times (Tfirst) deviate considerably from the other measured times; the Tfirst times are somewhere between 18k and 22k, and the other times are much smaller (between 12k and 14k), which corresponds to the factor of 1.4 from Table 1. Other measured times on this graph does not show much fluctuation, as the scale of the display is reduced due to the large Tfirst times; we see that some Tfirst times are almost 100% larger than the smallest measured times. With `BubbleC3`, all times are quite similar to each other; the graph shows some variations, but everything is between 14.6k and 15.6k; the differences between measured times are relatively small (approx. 6%).

For the conclusion: is it important to repeat the algorithm execution for several times to find the minimum time? As

N	*BubbleJ*	*BubbleC3*
500	1.98	1.80
5000	1.40	1.02
20000	1.05	1.01

Table 1: The ratio $f = \frac{\overline{T}_{first}}{T_{min}}$ between the average of the first and the minimal measured times

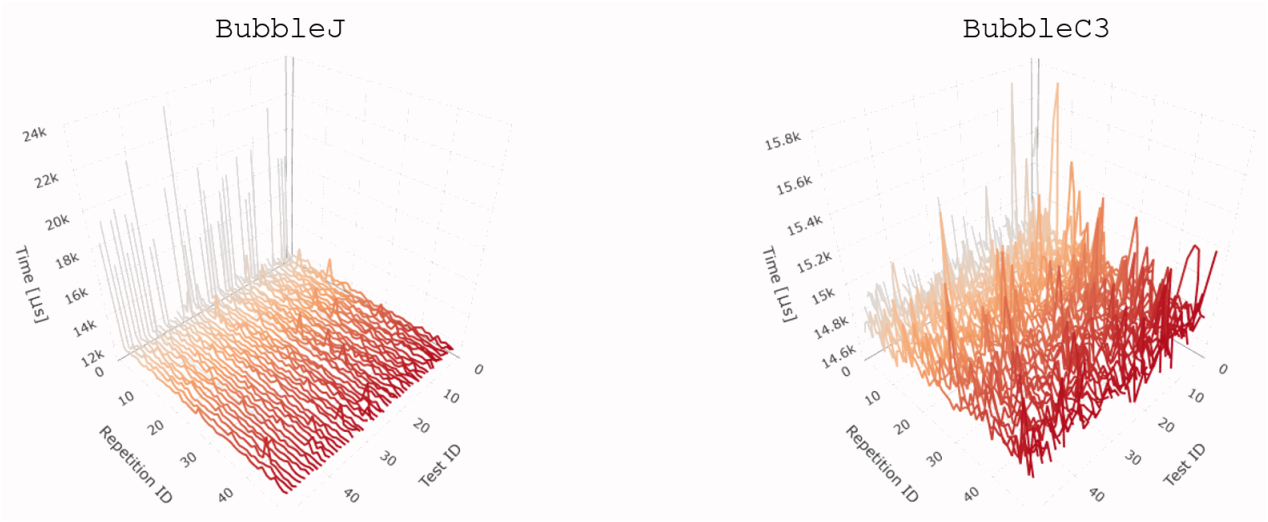

Figure 1: Times of execution of 50 identical test cases (50 repetitions of each test case) with `BubbleJ` and `BubbleC3`

the measurements show, the answer depends on the size of the input - the smaller the input, the more measurements are unreliable, so we need to take more measurements to get a good result.

Bar charts in Fig. 2 depict the proportion of measurements that differ from the smallest measurement by the given percentage range. The measurements on small inputs for the `BubbleJ` vary a lot. More than 36% of all measurements differ from the minimal time more than 10%. For the `BubbleC3` on the other hand only 17% of the measurements are that bad. When increasing the size of the input the results for both algorithms improve. For n=20000, for example, more than 73% (98%) of measurements differ from the minimal measurements for less than 1% for `BubbleJ` (`BubbleC3`) implementation.

The relative standard deviations of all measured times for `BubbleJ` are 21%, 7% and 1% for n=500, 5000 and 20.000 respectively. This confirms the claim that as the size of the input increases, the importance of multiple tests decreases. Since the relative standard deviations are even smaller for `BubbleC3` (namely 15%, 1%, 0.24%), the importance of a large number of measurements is even smaller here.

4. THE IMPACT OF THE PROGRAMMING LANGUAGE

We compared the times of execution of four implementations (`BubbleJ`, `BubbleA`, `BubbleC0` and `BubbleC3`) on randomly ordered sequences (RND) of length 500 to 50000 (step 500). Each test was executed 30 times. Fig. 3 shows the minimum measured times Tmin of all four algorithms.

We expected the `BubbleC3` to be the best, which was also proven with the measurements. The difference between `BubbleC0` and `BubbleC3` is somewhat surprising. Since BubbleSort is a simple algorithm, one would expect that the speedup resulting from the optimization would not be that great. But this is not the case, the difference is almost 2 times for large n. The relationship between `BubbleJ` and `BubbleA` is interesting. In a battle between fast implementations, Java turned out to be the slowest, although the differences in speed are not so great. Fitting all measurements with a quadratic functions results in the following:

`BubbleC0`:	$T_{min}(n) = 2.438n^2\ \mu s$
`BubbleJ`:	$T_{min}(n) = 1.372n^2\ \mu s$
`BubbleA`:	$T_{min}(n) = 1.311n^2\ \mu s$
`BubbleC3`:	$T_{min}(n) = 1.246n^2\ \mu s$

The ratio between the best (`BubbleC3`) and the worst (`BubbleC0`) implementation is 1 : 1.956, which we also noticed from the graph. More interesting is the ratio between the optimized C3 and Java implementation: `BubbleC3` : `BubbleJ` = 1 : 1,101. This means that for sorting random sequences Java is 10% slower than C. To find out, how good this conclusion is, lets calculate and depict the relative error

$$Error = \frac{|BubbleC3.T_{min} - 1.1 * BubbleJ.T_{min}|}{BubbleJ.T_{min}} * 100\%$$

Fig. 4 shows that for small inputs ($n < 5000$) the error is very big (as big as 1200%), but for larger inputs ($n > 10000$) the error is always less than 5% and it seems that it decreases when n increases.

As the last experiment we compared the times of execution of four implementations on inversely ordered sequences (INV) of length 500 to 50,000 (step 500). The results that are presented in Fig. 5 are somehow surprising.

The quality ranking of algorithms when sorting INV data changes comparing to the ranking on RND data. While `BubbleC0` remains the worst implementation, on the first place there is a swap - `BubbleC3` gives way to `BubbleA` and `BubbleJ`. Something similar happens with the sorted (SOR) data. This change in ranking is hard to explain, but according to our other research results which shown that the processor's branch predictor has a great impact to the execution time, we could speculate that the code generated by JVM is less suitable for branch prediction. With INV (and

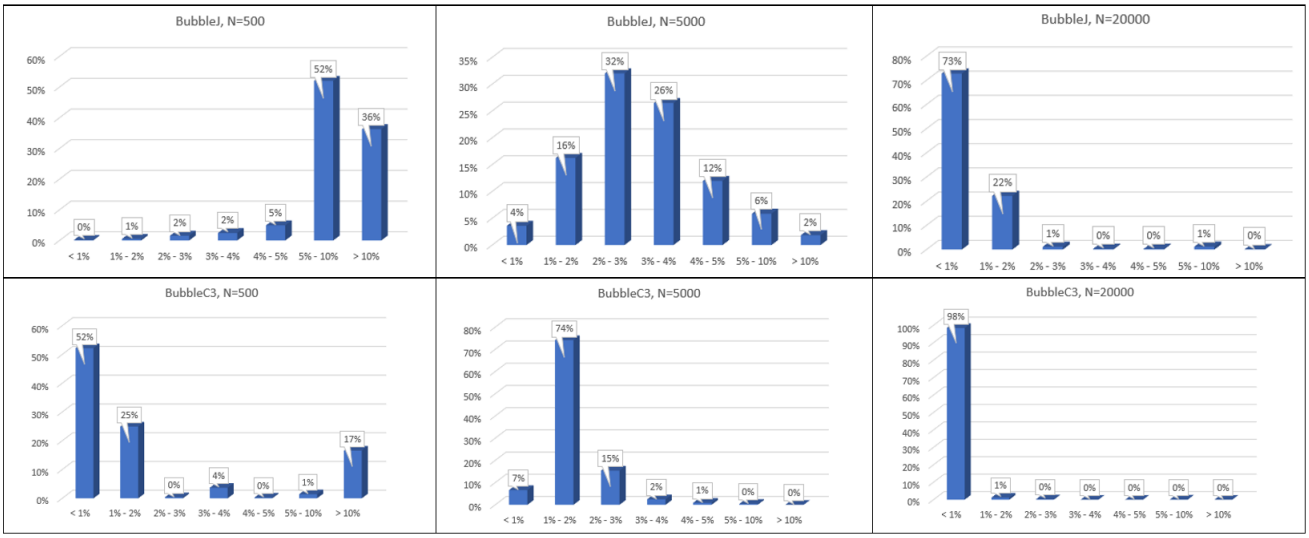

Figure 2: The proportion of measurements that differ from the smallest measurement by the given percentage range.

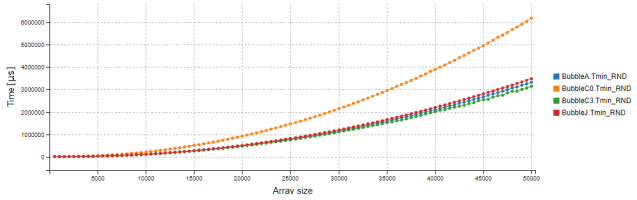

Figure 3: Tmin, RND data, $n = 500, ..., 50000$

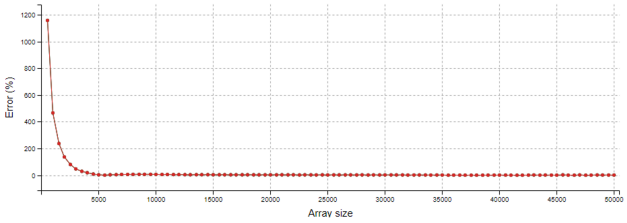

Figure 4: Relative error of estimating `BubbleJ`.Tmin with 1.1*`BubbleC3`.Tmin

SOR) data the code predictor is always correct, which could reflect in better performance. Anyway, the results unequivocally shows that the type of the input has a great impact on the quality of implementation. While with random data `BubbleC3` implementation was faster than `BubbleJ`, for inversely ordered data (and sorted, as we also found out) the Java implementation is the fastest.

5. CONCLUSIONS

The results presented in this paper show that there are many factors that have impact to the execution time of the algorithms. We have shown that despite a carefully controlled environment, deviations occur in measurements. The deviations are particularly pronounced in Java, since the way of measuring time here is significantly more sensitive to the influence of the environment than in C. We have seen that repeated execution of algorithms is especially important for

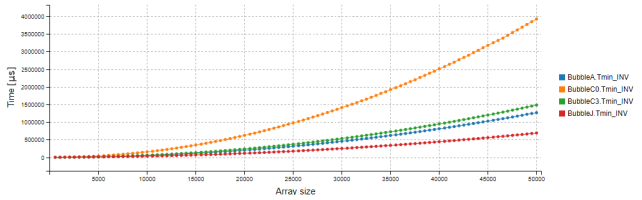

Figure 5: Tmin, INV data, $n = 500, ..., 50000$

small inputs. We also compared the differences between the programming languages. We showed that the difference between Java and C is not very big and that it depends on the type of input data - for randomly sorted arrays, the C implementation was faster, while for inversely ordered and already sorted data, Java took the first place in the ranking. In the future we could provide similar results for some other problems (to see if the results can be generalized), we could use another popular programming language (like Python) and we could investigate the real impact of the branch predictor the final results.

6. REFERENCES

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, 2nd edition, 2001.

[2] T. Dobravec. Algator — an automatic algorithm evaluation system. *Advances in Computers*, 116(1):65–131, 2019.

[3] M. Fernández. *Models of Computation, An Introduction to Computability Theory*. Springer, 2009.

[4] B. Swathi. A comparative study and analysis on the performance of the algorithms. *International Journal of Computer Science and Mobile Computing*, 5(1):91–95, Januar 2016.

[5] M. Tedre and N. Moisseinen. Experiments in computing: A survey. *The Scientific World Journal*, 2014(1):1–11, 2014.

Systematic generation of precedence based MILP models with P-graphs for multipurpose scheduling problems

Máté Hegyháti
Institute of Informatics and Mathematics
Bajcsy-Zs. u. 4.
Sopron, Hungary
hegyhati.mate@uni-sopron.hu

ABSTRACT

Scheduling of various processes is a widely researched topic in the literature. Different fields have their own specific constraints and parameters, thus, specialized approaches often emerge to tackle these needs efficiently. Solution methods have many flavors from general mathematical models such as mathematical programming, constraint programming; through general purpose heuristics, e.g., genetic algorithms, ant colony optimization; to problem specific tools like the S-graph framework.

A general aim of any newly developed method is to perform efficiently, and provide the optimal solution quickly, preferably faster than existing approaches. However, each tool has its strengths and weaknesses, and even for a well defined problem class, it is often not trivial to select the best approach for a problem instance in advance. This work focuses on a prerequisite of this dilemma: having the set of approaches to consider.

The aim of the paper is to present a modeling approach which enables the systematic generation of sound MILP models for a problem class. To illustrate this approach, a well known scheduling problem class from the batch process industry is considered, and the investigation is limited to only a specific type of MILP formulations, namely the general precedence models.

Keywords

scheduling, model generation, precedence, P-graph, MILP

1. INTRODUCTION AND LITERATURE

In general, the goal of scheduling is to time processes and often to allocate resources wisely in order to achieve a plan that meets all requirements or to find the best one among them. Due to its general nature, scheduling has been investigated in many different fields of science. In this work, the focus is on the production industry, where machines or equipment units are used to produce a certain set of goods. More specifically, makespan minimization for multipurpose batch processes is investigated, which has been addressed with various techniques over the last few decade, such as Mixed-Integer Linear Programming (MILP), S-graph framework, Linearly-Priced Timed Automata, and others[1, 7]. Even among MILP models, several different branches have emerged, the two main categories are precedence based and time discretization based formulations[2].

A lot of effort has been invested into the testing various models on case studies, benchmark examples, or a large number of randomly generated examples. When such a study is carried out, the examiner has to make a decision about the models to be included. This is not a straight forward decision even for only MILP models, as there is a huge number of them, often developed for a slightly different problem classes, and usually have several variants. Moreover, even with expert knowledge on the field, one can only include models that have already been published.

The aim of this work is to introduce a modeling approach, that results in a mathematical model enabling systematic generation of possible general precedence based models, including the ones published in the literature. Such a method would ensure for a comparative study, that it is not limited on manually hand-picked models, but consider all options within a given set of limitations. For the sake of simplicity and the illustration if this approach, this work focuses only on precedence based MILP models with general precedence variables for the most basic multipurpose batch scheduling problems without any common real life requirements such as changing- and setup times, storage limitations, etc. It would have been too ambitious for this study to include a a wider range of model types or a more complex scheduling problem class. This is left to future research. It is not the aim of this paper to provide any comparative results between the various MILP models, neither is to present the list of the numerous MILP formulations generated. The key messages of this study are the concepts and techniques used for building this model.

For the aforementioned purposes, the P-graph framework was utilized. P-graphs have been originally introduced as combinatorial models of process networks[3], that allows the developed algorithms to find feasible or optimal solution structures to synthesis problems[5, 4]. Since then, the models and the algorithms of the framework have been widely

applied in different areas of science from renewable supply chain optimization through polygeneration plant modeling to mobile workforce optimization[8, 6, 9].

The paper is structured as follows: Section 2 provides the definition of the scheduling problem class selected for illustration. Different modeling techniques among precedence based MILP formulations are briefly discussed in Section 3. Due to space limitations, completeness is not the aim of this section, only the showcasing of variations. To facilitate understanding, the utilized parts of the P-graph framework are briefly introduced in Section 4. The P-graph formulation of precedence based MILP models is illustrated in 5, again without the aim of completeness. Concluding remarks are shared in 6 along with possible directions of further research.

2. PROBLEM DEFINITION

The goal of the considered problem class is to minimize makespan of producing a set of products P with the set of equipment units: J. The production recipe of each product $p \in P$ is linear, and $n_p \in \mathbb{Z}^+$ denotes the number of steps. A derived set, $I = \{(p,n) \mid p \in P, n \in \{1,2,\ldots,n_p\}\}$ denotes the set of all tasks to be carried out. For simpler notation, a task is denoted by i instead of (p,n) when that level of detail is unnecessary. $I_j \subseteq I$ refers to the tasks that can be carried out by unit $j \in J$, and for all $i \in I_j$, the execution time is denoted by pt_{ij} in that unit. J_i is used to refer to units that can perform $i \in I$, i.e., $J_i = \{j \in J \mid i \in I_j\}$.

Each task is uninterruptible, must be assigned to exactly one unit, and a unit may not work on two tasks at the same time. Other common timing parameters for setups, changeovers, etc., are neglected, and intermediate materials can be stored at any amounts for any duration of time, i.e., Unlimited Intermediate Storage (UIS) Policy is considered. [1]

3. VARIATIONS FOR GENERAL PRECEDENCE BASED MILP MODELS

Precedence based models for the described problem class usually rely on decision variables about the assignment, sequencing, and starting time of tasks.

Assignment is generally modeled by a binary variable, Y_{ij} denoting whether unit $j \in J$ is assigned to task $i \in I$. Similarly, the exact starting time of task $i \in I$ is modeled by a continuous variable $S_i \in \mathbb{R}^{0+}$. The models also always define the makespan, MS as a target variable to be minimized.

This is the point, where models start to diverge. Some dedicate a continuous variable C_i for the completion time of each task, similar to S_i. The largest difference, however, appears in the sequencing variables, usually denoted by X. In some models, $X_{iji'}$ takes the value of 1 if and only if both tasks i and i' are assigned to unit j and the decision has been made to perform i before i'. With such a variable, the constraint to properly sequence tasks can be expressed by the following inequality, where M is a sufficiently large number:

$$S_{i'} \geq S_i + pt_{ij} - M \cdot (1 - X_{iji'}) \qquad \forall j \in J,\, i, i' \in I_j$$

This formulation, however, requires an unnecessarily large number of binary variables, thus models often employ another variable, $X_{ii'}$ that takes the value of 1 if tasks i and i' are assigned to the same unit, and the decision has been made that i is performed before i'. With such variables, the constraint above can be replaced with another:

$$S_{i'} \geq S_i + pt_{ij} - M \cdot (3 - X_{ii'} - Y_{ij} - Y_{i'j}) \qquad \forall j \in J,\, i, i' \in I_j$$

Note, that the value of $X_{ii'}$ is irrelevant if i and i' are assigned to different units. If that feature is ensured by another constraint, and a separate completion variable is defined, sequencing can be enforced via the following simple constraint:

$$S_{i'} \geq C_i - M \cdot (1 - X_{ii'}) \qquad \forall i, i' \in I,\, J_i \cap J_{i'} \neq \emptyset$$

Again, this constraint does not require $X_{ii'}$ to be 0 if the tasks are assigned to different units, it only needs them to be allowed to be 0.

Following similar thinking, other variants of the X variables and their corresponding constraints can be derived. Such further discussion is omitted here, however, there other ways how different model variants can be generated. For example, $X_{ii'}$ variables could either be defined for all i, i' pairs in both directions or only for $i < i'$ if a total ordering is defined. Or, when C_i is introduced, it may be required to be the exact time when i finishes, or just any time point after that, as discussed in a bit more detail in Section 5. Also, if the same requirement of the problem class can be expressed in various ways, only one can be selected or even several of them redundantly. The latter may seem unreasonable, however, testing experience often shows, that redundant constraints may affect the search space in such a way, that MILP solvers and their heuristics can find better solutions or bounds.

4. BRIEF INTRODUCTION TO P-GRAPHS

The P-graph framework was introduced to provide a rigorous and efficient framework to generate feasible process networks [3], or find the optimal one among them[5, 4]. This work relies only on the original P-graph model (without any extensions or additional parameters) and the SSG (Solution Structure Generator) algorithm, that generates all combinatorially feasible solution structures.

A P-graph is an (M, O) pair, where M denotes the set of materials, and O the set of operating units. Each operating unit $o \in O$ is a pair of material sets: its (mandatory) inputs and outputs. A Process Network Synthesis (PNS) problem is defined by a triple $(\mathsf{P}, \mathsf{R}, \mathsf{O})$ where

P is the set of product materials that must be produced by at least one operating unit.

R is the set of raw materials that can be consumed without being produced by an operating unit.

O is the set of possible operating unit to be included in the network.

[1] In this simplified form, the problem class is reduced to Flexible Job Shop Scheduling. However, as the research was motivated by the batch production industry, its terminology is kept along with its standard notation, that has less conflicts with that of the P-graph framework.

The SSG algorithms takes (P, R, O) as its input, and generates all of the feasible subsets of O that satisfy the 5 axioms for feasible solution structures.

5. SYSTEMATIC MODEL GENERATION

In many industrial applications of the P-graph framework, the original meaning of operating units and materials is extended to broader concepts. Materials often represent logical states, and operating units may model transportation, etc. This application of P-graphs follows the same idea. There are, however, terms like intermediates, product, unit, etc., that both appear in the underlying process to be scheduled and in the P-graph model, that is used to generate MILP models. To avoid confusion, the elements of the P-graph model will be referred to as M-, P-, I-,R- and O-nodes.

5.1 P- and R-nodes

The R-nodes of the proposed model are the variable declarations with their domain, such as:

- $S_i \in \mathbb{R}^{0+} \quad \forall i \in I$
- $C_i \in \mathbb{R}^{0+} \quad \forall i \in I$
- $Y_{ij} \in \{0,1\} \quad \forall j \in J, i \in I$
- $X_{ii'} \in \{0,1\} \quad \forall i, i' \in I,\ i < i', J_i \cap J_{i'} \neq \emptyset$
- $X_{ii'} \in \{0,1\} \quad \forall i, i' \in I,\ i > i', J_i \cap J_{i'} \neq \emptyset$
- $X_{iji'} \in \{0,1\} \quad \forall j \in J, i, i' \in I_j, i < i'$
- $X_{iji'} \in \{0,1\} \quad \forall j \in J, i, i' \in I_j, i > i'$

The P-nodes are the feasibility constraints of the scheduling problem:

P1 Each task must be carried out.

P2 Production steps of a product must be in order.

P3 Tasks assigned to the same unit can not overlap.

P4 The shutdown of the facility must happen after all of the products are produced.

These are the results that has to be produced (ensured) from the R-nodes. To do so, O-nodes are available.

5.2 O- and I-nodes

One type of O-nodes are the possible constraints in a precedence based MILP model.

As an example, the following constraint is an O-node, that consumes the R-node representing Y and generates P1:

$$\sum_{j \in J} Y_{ij} = 1 \quad \forall i \in I$$

As another example, the following constraint is another O-node that generates P2:

$$S_{(p,n+1)} \geq C_{(p,n)} \quad \forall p \in P, n \in \{1, 2, \ldots, n_p - 1\}$$

The inputs of this O-node are less obvious and worth a short discussion. Seemingly there should be two of them, the R-nodes representing the S_i and C_i variables. However, that is not correct, as the R-nodes only represent the availability of those variables, but don't ensure additional semantic features. Namely, the R-node for C_i only ensures that there is a variable named C but nothing about its timing. For the O-node above to produce P2, such a C variable is needed, whose value is ensured to be greater or equal than the completion time of the task. This semantically correct C variable is different from the R-node above, and can be represented by an I-node. In turn, this I-node can be produced by the following O-node:

$$C_i \geq S_i + \sum_{j \in J_i} Y_{ij} \cdot pt_{ij} \quad \forall i \in I$$

or by another O-node:

$$C_i \geq S_i + Y_{ij} \cdot pt_{ij} \quad \forall i \in I, j \in J_i$$

Both O-nodes require the R-nodes for C and S variables as inputs. Again, the R-node representing the Y variables is not an input, as it does not hold the semantic meaning, that one unit is assigned to a task. This is represented by P1.

An observant reader may point out that the first suggested O-node above should have had equality:

$$C_i = S_i + \sum_{j \in J_i} Y_{ij} \cdot pt_{ij} \quad \forall i \in I$$

This constraint, of course is a valid, and a different O-node in the proposed model. However, the output of this node is not the same, as in this case, it is ensured that C_i is exactly the finishing time of a task, not a time point at least that much. Some O-nodes may require this more specific version of C, others the previous one. On the other hand, if this new I-node for the C variables with exact timing is available, the other I-node is also implied. This implication is also represented by an O-node, that has no constraint assigned to it.

Thus, I-nodes in the proposed model are concepts similar to the P-nodes, albeit more specific to the variables of the model. O-nodes represent either constraints or logical implications.

5.3 Illustration

Figure 1 illustrates the part of the model that in the previous section.

It is easy to see, that there are 7 feasible solution structures (3 of them without redundancy) to just this small part of the P-graph model. Generating the model from a solution structure is straight forward, the constraints and variable declarations corresponding to included O-nodes and R-nodes compose the MILP model together with the objective function to minimize the makespan.

Note, however, that there could be other O-nodes producing P2, further increasing the number of possible models. For example:

$$S_{(p,n+1)} \geq S_{(p,n)} + \sum_{j \in J_{(p,n)}} Y_{(p,n)j} \cdot pt_{(p,n)j} \quad \forall (p,n) \in I$$

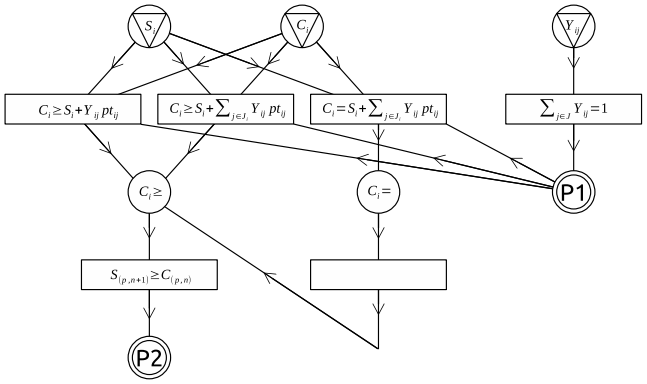

Figure 1: Part of the proposed P-graph model

5.4 Remarks on extendability

The rest of the model could be formulated in a similar fashion. Naturally, the presented model only considers constraints that are included as O-nodes. It will not discover new type of constraints, only find the sound combinations of proposed ones.

The modeling procedure, however, allows extension by addition. For example, if one were to consider immediate precedence variables as well, that would introduce an additional I-node for an X variable that satisfies the required conditions, that are more strict than the ones required by general precedence models. Moreover, O-nodes would be added to produce and consume this I-node. Some models also employ an integer variable, that denotes the position of a task in the assigned units production sequence. This could be implemented by adding R-, I- and O-nodes similarly.

Thus, this is a model that can continuously grow, and feasible solution structures of a previous version would still remain feasible.

There are still options, that were not discussed here due to space limitations, and to the early stage of this research. For example, in an extended model, some I-nodes could require a mutually exclusive relation. This is a feature, that is not supported by the P-graph tools introduced here. P-graphs with material balances and an underlying MILP model can, however, easily model this.

Also, the presented model only allows redundancy among constraints, but not among variables. Simplest example to this would be, if two sets of C_i variables were introduced, and some would be used by one subset of O-nodes, and the other one by others. This feature, again, could be modeled by introducing material balances and quantities into the model.

6. CONCLUSIONS & FUTURE RESEARCH

In this paper, an approach is presented to develop a model, that can generate MILP models for a specific problem class. This modeling technique - based on P-graphs - is illustrated on the simple class of multipurpose batch process scheduling problems and with the scope on general precedence based MILP models by P-graphs. Following this technique results in a large PNS problem, where raw materials are variable declarations and operating units are linear constraints on them. An MILP model can easily be derived for the original illustrative problem class for every solution structure generated by SSG for this PNS problem. The model is backwards-compatible with addition, i.e., the introduction of new variables and constraints does not render previous solution structures (MILP formulations) infeasible (invalid).

This research is in an early stage. There are numerous additions and extensions of the proposed approach that could be considered for future research. After the inclusion of intermediate precedence variables, parts of time discretization models may be added, which may help uncover previously unknown hybrid models. Extensive performance testing of generated models, and observing the reduction between model variants by the preprocessor of various MILP solvers could be of interest.

7. ACKNOWLEDGEMENT

The author thanks Nikolett Sós, whose diploma project consultations sparked the idea behind this work.

8. REFERENCES

[1] State-of-the-art review of optimization methods for short-term scheduling of batch processes. *Computers & Chemical Engineering*, 30:913–946, 2006.

[2] C. A. Floudas and X. Lin. Continuous-time versus discrete-time approaches for scheduling of chemical processes: a review. *Computers & Chemical Engineering*, 28:2109–2129, 2004.

[3] F. Friedler, K. Tarján, Y. W. Huang, L. T. Fan, K. Tarjan, Y. W. Huang, and L. T. Fan. Graph-theoretic approach to process synthesis: Axioms and theorems. *Chem. Engng Sci.*, 47:1973–1988, 1992.

[4] F. Friedler, B. J. Varga, E. Fehér, and L. T. Fan. *State of the Art in Global Optimization*, chapter Combinatorially Accelerated Branch-and-Bound Method for Solving the MIP Model of Process Network Synthesis, pages 609–626. Kluwer Academic Publishers, Dordrecht, 1996.

[5] F. Friedler, J. B. Varga, and L. T. Fan. Decision-mapping: A tool for consistent and complete decisions in process synthesis. *Chemical Engineering Science*, 50:1755–1768, 1995.

[6] F. Friedler, Ákos Orosz, and J. P. Losada. *P-graphs for Process Systems Engineering*. Springer, Cham, 1 edition, 2022.

[7] M. Hegyhati and F. Friedler. Overview of industrial batch process scheduling. *Chemical Engineering Transactions*, 21:895–900, 2010.

[8] J. Klemes and P. Varbanov. Spreading the message: P-graph enhancements: Implementations and applications. *Chemical Engineering Transactions*, 45:1333–1338, 10 2015.

[9] A. Éles, I. Heckl, and H. Cabezas. New general mixed-integer linear programming model for mobile workforce management. *Optimization and Engineering*, Feb. 2021. IF: 2.760.

On relations of Watson-Crick finite automata to other computational paradigms

[Extended Abstract]

Benedek Nagy
Department of Mathematics, Eastern Mediterranean University
Famagusta, North Cyprus, via Mersin 10, Turkey
Institute of Mathematics and Informatics, Eszterházy Károly Catholic University, Eger, Hungary
nbenedek.inf@gmail.com

ABSTRACT

We study language classes that are accepted by some variants of Watson-Crick finite automata, i.e., with a 2-head model of finite automata working on Watson-Crick tape modeling DNA molecules. We show a relation between sticker systems and stateless traditional Watson-Crick automata where the two heads scan the input in the same direction. We also establish a new connection between external contextual grammars with choice to the sensing $5' \to 3'$ Watson-Crick automata, i.e., to the 2-head model of finite automata where the two heads starting from the two extremes of the input and they move in opposite direction till they meet.

Categories and Subject Descriptors

F.4.3 [**Formal Languages**]

General Terms

Theory, Automata, Languages, Computing paradigms

Keywords

Watson-Crick automata, sticker systems, formal languages, finite state machines, stateless automata, external contextual grammars, linear languages

1. INTRODUCTION

On the one hand, contextual grammars are one of the formal methods developed to generate languages [2, 9, 21] initiated by Solomon Marcus in the end of 1960s. The generation of the words, i.e., the derivation process is going by inserting or attaching two subwords at the same time to the actual word. The inserted/attached subwords are called context and the condition how the process is done, the "choice", is handled by selection languages. The main motivation comes from linguistics, as there are various non-context-free structures that occur in natural languages, in general. Contextual grammars give a somewhat orthogonal classes of languages to the Chomsky hierarchy. However, they are not independent of each other, as in the former model, the class of selection languages is frequently chosen as one of the classes of the Chomsky hierarchy. There are two main types of contextual grammars, the internal and the external types, in the former the words of a context are inserted inside the actual word, while in the latter the words of the context are attached to the two ends of the actual word. In this paper, we are interested mostly in the external contextual grammars, thus we will only recall those in Section 2. On the other hand, DNA computing belongs to new computing paradigms emerged in the end of the last century. Two of the 'traditional' models of DNA computing, the filtering and the sticker systems are in close relation to the ground breaking experiments of Adleman and Lipton solving (some instants) of the Hamiltonian path and SAT problems by coding graphs in sets of DNA molecules, filtering out and detecting the result [1, 8]. DNA computing and its models can also be used to generate/accept formal languages, and not only to solve some (combinatorially hard) problems. The sticker systems can also be used for language generation based on the sticking operation: DNA molecules having single stranded, so-called 'sticky' ends, can stick together to form a larger molecule. For a good analogy, one may think about how a (long) line can be built by dominos. (A more formal description is given in Section 2.)

Another early formal DNA computing model, named by the Nobel price winners Watson and Crick, who discovered the structure of the DNA molecules, is a generalisation of finite state automata working on two-stranded DNA molecules (also called Watson-Crick tape) instead of the traditional tape [3]. These models of computation, namely Watson-Crick automata, are entirely defined to describe some formal languages, in a similar manner as models of traditional automata theory do. Already it is mentioned in [20, 22] that Watson-Crick finite automata are developed as a kind of accepting counterpart of the language generating sticker systems.

In this paper, our aim is twofold, first we investigate a formal connection between some sticker systems and some variants of Watson-Crick finite automata, and then, we also concentrate on connections of reverse Watson-Crick automata and external contextual grammars. Some variants of the reverse Watson-Crick automata are also called $5' \to 3'$ Watson-

Crick automata as the heads of these automata are scanning the input DNA in opposite physical directions, i.e., both strands from their $5'$ to $3'$ direction. We note here that there is an expansive literature on the sensing $5' \to 3'$ Watson-Crick automata [10, 11, 12, 13, 14, 16, 17, 18, 19] in which the process on the input finishes when the heads meet if the automaton does not get stuck earlier. The structure of this paper is as follows. In the next section we recall some formal definitions from the literature mentioned earlier. In Section 3, the first part of the main results are presented, by establishing a connection between sticker systems and Watson-Crick automata. Then, in Section 4, we present connections between external contextual grammars and sensing $5' \to 3'$ Watson-Crick automata.

2. NOTATIONS AND DEFINITIONS

Here, we recall some important definitions and fix our notations. We assume that the reader already knows the basic concepts of finite automata, formal languages, generative grammars and computing. We denote the empty word by λ.

Let us start with external contextual grammars with choice (also called selection). Formally, an external contextual grammar EC is a triplet (V, A, P) where V is a finite alphabet, A is a finite set of words over V, the set of axioms and P is a finite set of pairs. The elements of P are of the form (C, S) where $C \subset V^* \times V^*$ contains the contexts and S is a language over V, the selection language for context C. The direct derivation relation is defined as follows: for any word $x \in V^*$, $x \Rightarrow uxv$ if there is a context $C = (u, v)$ such that $x \in S$, for a pair $(C, S) \in P$. As usual, the direct derivations (or also called derivation steps) can be extended to the derivation relation, denoted by $\Rightarrow^*$ by taking their reflexive and transitive closure. The generated language is then defined as

$$L(EC) = \{w \in V^* | \text{ there exists } x \in A \text{ such that } x \Rightarrow^* w\}.$$

We say that an external contextual grammar is without choice if $S = V^*$ for every element of P. If all choice languages are regular/linear etc., then we say that this is an external contextual grammar with regular/linear, etc. choice.

Now we describe some basic facts about DNA. DNA molecules are built up by 4 types of nucleotides which are usually abbreviated as A,C,G,T by their name. A DNA strand can be seen as a sequence of nucleotides. The sequence has two ends, one of them is denoted by $5'$ and the other by $3'$. These ends can easily be distinguished chemically. The nucleotides have a so-called Watson-Crick complementarity relation meaning that A and T are pairs of each other as well as C and G are pairs of each other. Two DNA strands could form a DNA molecule that is double stranded, if they are complement of each other as follows. At each position of a strand the nucleotide is the Watson-Crick pair of the nucleotide of the other strand at that position. Moreover, the direction of the two strands are opposite, i.e., the nucleotide of the $5'$ end of a strand is paired to the nucleotide of the $3'$ end of the other. A full double stranded DNA molecule is denoted by $\left[\begin{smallmatrix} u \\ v \end{smallmatrix}\right]$ with u on the upper and v on the lower strand. With the notation $\binom{u}{v}$ we may denote a DNA which has u on the upper and v on the lower strand, but it may not be a full double stranded molecule, e.g., their lengths may not be identical. When one of the strands has some extra nucleotides on one of the ends, than this is called a sticky end, as the molecule can be expanded by sticking there an appropriate other molecule...

Instead of the original nucleotides, in formal models we may use letters of any alphabet. Moreover, as [6] has proven, we may use the identity relation in the role of the complementarity relation without loss of generality, both at sticker systems and at traditional Watson-Crick automata. Thus, for simplicity, from now on we use always the identity relation.

Now, we are ready to see Watson-Crick automata (WK automata shortly). These automata work on (full double stranded) DNA molecules; they have two reading heads one for each of the two DNA strands. Formally, a Watson-Crick automaton is a 6-tuple $M = (V, \rho, Q, q_0, F, \delta)$, where V is the (input) alphabet, $\rho \subseteq V \times V$ denotes a complementarity relation, (in this paper, we use the identity), Q represents a finite set of states, $q_0 \in Q$ is the initial state, $F \subseteq Q$ is the set of final (accepting) states and δ is called transition mapping and it is of the form $\delta : Q \times \begin{pmatrix} V^* \\ V^* \end{pmatrix} \to 2^Q$, such that it is non empty only for finitely many triplets of q, u and v, i.e., $(q, \binom{u}{v})$, where $q \in Q, u, v \in V^*$. In case $Q = F = \{q_0\}$, we say that the automaton is stateless.

The description so far has not distinguished the traditional and the sensing $5' \to 3'$ models. Thus, the configuration and the computation steps, i.e., the transitions between configurations play crucial role to specify the type of the model. In a traditional WK automaton, both heads of the automaton starts from the same place, from one of the ends of the input DNA molecule, and both of them read the entire strand in an accepting computation. The input molecule is accepted, if the automaton could reach an accepting state when both strands are fully scanned. Formally, a configuration of a traditional WK automaton is a triplet $\left(\binom{u}{v}, q\right)$ in which $u, v \in V^*$ are the upper and lower strand parts of the input that have not processed yet, and $q \in Q$ is the actual state. The initial configuration is $\left(\binom{w}{w}, q_0\right)$ with the input word w in both strands, as at that stage of the computation the whole input DNA $\left[\begin{smallmatrix} w \\ w \end{smallmatrix}\right]$ is to be processed. From the configuration $\left(\binom{bu}{cv}, q\right)$ the configuration $\left(\binom{u}{v}, p\right)$ directly computed (or computed in one step), denoted by $\left(\binom{bu}{cv}, q\right) \Rightarrow \left(\binom{u}{v}, p\right)$), if $p \in \delta\left(q, \binom{b}{c}\right)$. The reflexive and transitive closure of $\Rightarrow$ is denoted by $\Rightarrow^*$ and called computation. A DNA molecule $\left[\begin{smallmatrix} w \\ w \end{smallmatrix}\right]$ or, let us say, the word w, is accepted if $\left(\binom{w}{w}, q_0\right) \Rightarrow^* \left(\binom{\lambda}{\lambda}, p\right)$ for some $p \in F$. The set of accepted words is the accepted language.

However, as we have already mentioned, the two strands of the DNA molecule have opposite $5' \to 3'$ orientations, thus it is worth to take into account such variants of Watson-Crick automata that parse the two strands of the Watson-Crick tape in opposite directions. These automata are also called reverse WK automata. Obviously, if each head starts to read its strand from its $5'$ end, as usually enzymes also do, then they start to scan the input DNA from physically different ends [7, 22]. On the other hand, as the two strands of a DNA molecule are determined by each other via the Watson-Crick complementarity relation, we may also think about the sensing $5' \to 3'$ WK automata, in which the two heads sense

if they arrived into the same position. Since, actually, by this point of the computation, exactly one of the letters of each position has been already read, this model finishes the work on the input at this point and decides on the acceptance. It is easy to see, that the complementarity relation can also be substituted by the identity in case of these automata without changing their computational power, (i.e., the class of languages that can be formed by the words on the upper strands of the accepted molecules).

Formally, a configuration of a sensing $5' \to 3'$ WK automaton is a pair (q, w) where q is the current state of the automaton and w is the (double stranded) part of the input word which has not been processed (read) yet by any of the heads. For $w', x, y \in V^*, q, q' \in Q$, we write a transition, a step of a computation, between two configurations as: $(q, xw'y) \Rightarrow (q', w')$ if and only if $q' \in \delta(q, \binom{x}{y})$. We denote the reflexive and transitive closure of the relation $\Rightarrow$ by $\Rightarrow^*$ also in this case. Therefore, for a given $w \in V^*$, an accepting computation is a sequence of transitions $(q_0, w) \Rightarrow^* (p, \lambda)$, starting from the initial state and ending in a final state $p \in F$. The language accepted by a sensing $5' \to 3'$ WK automaton M is:

$$L(M) = \{w \in V^* \mid (q_0, w) \Rightarrow^* (p, \lambda), p \in F\}.$$

Finally, let us recall the sticker systems. We note here that sticker systems, as computing model, can be used to provide efficient solutions to computationally hard, e.g., NP-complete problems (at least theoretically) [1, 4, 8, 15, 23]. However, in this paper we are concentrating in their language generating feature, thus we present them in the sequel from this point of view [5].

Single stranded DNA molecules of the form $\binom{u}{\lambda}$ are called upper dominos, while ones of the form $\binom{\lambda}{v}$ are called lower dominos. The set of double stranded DNA molecules with sticky end on the right side are denoted by $R(V)$: they are either of the form $\left[\begin{smallmatrix} w \\ w \end{smallmatrix}\right]\binom{u}{\lambda}$ or $\left[\begin{smallmatrix} w \\ w \end{smallmatrix}\right]\binom{\lambda}{v}$.

A simple regular sticker system is a pentuple $(V, \varrho, D_u, D_l, A)$, where
- V is an alphabet,
- ϱ is the complementarity relation (we use identity in this paper),
- D_u and D_l are sets of dominos that are associated to the upper and lower strand, respectively,
- $A \subset R(V)$ is a finite set of axioms.

The generation starts with an axiom and goes step by step. Depending on the position of the sticky end of the actual molecule, only dominos of the other strand may stick to it by filling that strand without gaps and maybe further expanding it. The generation has finished when a full double stranded DNA is produced. The generated language is defined consequently [5, 15].

Those special cases are called sticker systems without axioms, when $A = \left\{\binom{\lambda}{\lambda}\right\}$ to infer the fact that the molecules are built up only by dominos and the generation could start from any dominos in this case.

Let $(V, \varrho, D_u, D_l, A)$ be a simple regular sticker system. Let T be a finite alphabet and $u : D_u \to T$, $\ell : D_l \to T$ be two mappings to T. For a finished computation, let us assign the string $w_u \in T^*$, by using u, to the sequence of dominos used in the upper strand. Similarly, let us assign $w_l \in T^*$, by using ℓ, for the sequence of dominos used in the lower strand.

Further, a computation is coherent, if $w_u = w_l$. Based on that, the language generated by a simple regular sticker system R in coherent way is

$$\left\{w \mid \begin{bmatrix} w \\ w \end{bmatrix} \text{ can be generated in coherent way in } R\right\}.$$

3. STICKER SYSTEMS AND WATSON-CRICK AUTOMATA

In this section we present relations between two paradigms dealing with formal languages in DNA computing.

It is known that if a language L can be obtained by a simple regular sticker system in a coherent way, then there is a WK automaton that accepts L [22].

Now, we give a kind of extension of this result.

THEOREM 1. *If the language L can be obtained by a simple regular sticker system without axiom in a coherent way, then L is also accepted by a stateless WK automaton.*

PROOF. Let $(V, \varrho, D_u, D_l, A)$ be a simple regular sticker system that generates L in coherent way with the mappings $u : D_u \to T$ and $\ell : D_l \to T$ with some alphabet T. Let us construct a stateless WK automaton M that accepts L as follows. Let $M = (V, \varrho, \{q\}, q, \{q\}, \delta)$, where δ is defined in the following way: $q \in \delta\left(q, \binom{x}{y}\right)$, for all pairs $\binom{x}{\lambda} \in D_u$, $\binom{\lambda}{y} \in D_l$, if $u(x) = \ell(y)$.

By the construction, the application of the sticking operation in a coherent way is matched with the computing steps of M, and thus, exactly those molecules can be generated by the given sticker system as the ones accepted by M. □

4. $5' \to 3'$ WK AUTOMATA AND EXTERNAL CONTEXTUAL GRAMMARS

In this section we present two new links between Watson-Crick automata and external contextual grammars. On the first hand, sensing $5' \to 3'$ WK automata accept exactly the class of linear languages of the Chomsky hierarchy [10, 11, 13, 14, 17, 18]. It is also known (see, e.g., [2]) that external contextual grammars without choice generate exactly those languages which can be generated by linear grammar with exactly one nonterminal symbol. First, we complement these results as follows.

THEOREM 2. *If the language L is generated by external contextual grammars without choice such that the only axiom is the empty word λ, then L is accepted by a stateless sensing $5' \to 3'$ WK automaton.*

PROOF. We prove by construction. Let $EC = (V, \{\lambda\}, P)$ be an external contextual grammars without choice, where

P contains pairs (C_i, V^*) with $C_i = (u_i, v_i)$. Let $M = (V, \rho, \{q\}, q, \{q\}, \delta)$ be a stateless $5' \to 3'$ WK automaton with the identity relation on V as ρ and let δ be defined as follows based on P: $q \in \delta(q, \binom{u_i}{v_i})$ for each C_i. It is easy to see that derivations, i.e., generations of words in EC correspond step by step in an opposite order to accepting computations in M and vice versa. □

Further, we can establish the following new connection between these models.

THEOREM 3. *If the language L is accepted by sensing $5' \to 3'$ WK automaton, then it can be generated by external contextual grammars with linear choice languages.*

PROOF. Let $M = (V, \rho, Q, q_0, F, \delta)$ be a $5' \to 3'$ WK automaton that accepts L, where ρ is the identity relation on V. For each state $q \in Q$, let L_q denote the linear language that can be accepted from q, i.e., the language accepted by $M_q = (V, \rho, Q, q, F, \delta)$. Obviously, $L = L_{q_0}$. Now, we give an external contextual grammar EC with linear choice. Obviously the same alphabet V is used. Further, let the set of axioms be $A = \{\lambda\}$. Now, for each transition $p \in \delta(q, \binom{u}{v})$ of M (where $p, q \in Q, u, v \in V^*$), let the pair (C, S) $C = (u, v)$ and the choice L_p be in P. Thus, the system $EC = (V, A, P)$ is able to produce exactly L by a backward stepwise simulation of the accepting computations. □

5. CONCLUSIONS

Languages that can be obtained by a simple regular sticker system without axiom in a coherent way, are shown to be accepted by WK automaton with a sole state. On the other hand, languages generated by external contextual grammar without choice and with linear choice are related to languages of sensing $5' \to 3'$ WK automata.

6. REFERENCES

[1] L. M. Adleman. Molecular computation of solutions to combinatorial problems. *Science*, 226(5187):1021–1024, 1994.

[2] R. Ceterchi. Marcus contextual grammars. In *Formal languages and applications* (C. Martín-Vide, V. Mitrana and Gh. Păun, eds.), Springer, pp. 335–366, 2004.

[3] R. Freund, Gh. Păun, G. Rozenberg and A. Salomaa. Watson-Crick finite automata. In *3rd DIMACS Sympozium On DNA Based Computers, Philadelphia*, pp. 305–317, 1997.

[4] Z. Ignatova, I. Martínez-Pérez and K.-H. Zimmermann. *DNA computing models.* Springer, New York, 2008.

[5] L. Kari, Gh. Păun, G. Rozenberg, A. Salomaa and S. Yu. DNA computing, sticker systems, and universality. *Acta Informatica* 35: 401–420, 1998.

[6] D. Kuske and P. Weigel. The role of the complementarity relation in Watson-Crick automata and sticker systems. In *Developments in Language Theory, DLT 2004, LNCS* 3340, Springer, Berlin, Heidelberg, pp. 272—283, 2004.

[7] P. Leupold and B. Nagy. $5' \to 3'$ Watson-Crick automata with several runs. *Fundamenta Informaticae* 104, pp. 71-–91, 2010.

[8] R. J. Lipton. DNA solution of hard computational problems. *Science*, 268(5210), pp. 542–545, 1995.

[9] S. Marcus. Contextual grammars. *Revue Roum. Math. Pures Appl.*, 14, pp. 1525—1534, 1969.

[10] B. Nagy. On $5' \to 3'$ sensing Watson-Crick finite automata. In: *Preliminary proceedings of DNA13: The 13th International Meeting on DNA computing, Memphis, Tennessee, USA*, pp. 327–336, 2007.

[11] B. Nagy. On $5' \to 3'$ sensing Watson-Crick finite automata. In: *DNA Computing. DNA 2007: Selected revised papers, LNCS* 4848, Springer, Berlin, Heidelberg, pp. 256—262, 2008.

[12] B. Nagy. On a hierarchy of $5' \to 3'$ sensing WK finite automata languages. In: *Computaility in Europe, CiE 2009: Mathematical Theory and Computational Practice, Abstract Booklet*, Heidelberg, pp. 266—275, 2009.

[13] B. Nagy. $5' \to 3'$ Sensing Watson-Crick Finite Automata. In: G. Fung, ed.: *Sequence and Genome Analysis II - Methods and Applications*, iConcept Press, pp. 39—56, 2010.

[14] B. Nagy. On a hierarchy of $5' \to 3'$ sensing Watson-Crick finite automata languages. *Journal of Logic and Computation* 23(4), pp. 855—872, 2013.

[15] B. Nagy. *DNS számítógépek és formális modelljeik (in Hungarian).* Typotex, Budapest, Hungary, 2014.

[16] B. Nagy and S. Parchami. On deterministic sensing $5' \to 3'$ Watson–Crick finite automata: a full hierarchy in 2detLIN. *Acta Informatica* 58 pp. 153–175, 2021.

[17] B. Nagy and S. Parchami. $5' \to 3'$ Watson-Crick automata languages – without the sensing parameter. *Natural Computing*, online first, 2022. doi: 10.1007/s11047-021-09869-9

[18] B. Nagy, S. Parchami and H. M. M. Sadeghi. A new sensing $5' \to 3'$ Watson-Crick automata concept. In: *Proc. 15th Int. Conf. Automata and Formal Languages, AFL 2017, EPTCS* 252, pp. 195–204, 2017.

[19] S. Parchami and B. Nagy. Deterministic Sensing $5' \to 3'$ Watson-Crick Automata Without Sensing Parameter. In: *Unconventional Computation and Natural Computation, UCNC 2018, LNCS* 10867, Springer, pp. 173–187, 2018.

[20] Gh. Păun. DNA Computing by Matching: Sticker Systems and Watson-Crick Automata, In *Pattern Formation in Biology, Vision and Dynamics* (A. Carbone, M. Gromov, P. Prusinkiewicz, eds.), World Scientific, Singapore, pp. 336—362, 2000.

[21] Gh. Păun. Marcus Contextual Grammars. Kluwer Publ. House, Doordrecht, 1998.

[22] Gh. Păun, G. Rozenberg and A. Salomaa. *DNA Computing: New Computing Paradigms.* Springer-Verlag, Berlin, Heidelberg, 2002.

[23] S. Roweis, E. Winfree, R. Burgoyne, N. V. Chelyapov, M. F. Goodman, P. W. K. Rothemund, and L. M. Adleman. Sticker-based model for DNA computation. *Journal of Computational Biology* 5, pp. 615–629, 1998.

Surrogate Component Approach for a Synchronization Problem

Alejandro Olivas González
Labex IMOBS3
LIMOS Lab, UCA/CNRS
Clermont-Ferrand, France

Alain Quilliot
Labex IMOBS3
LIMOS Lab, UCA/CNRS
Clermont-Ferrand, France
alain.quilliot@uca.fr

Hélène Toussaint
LIMOS UCA/CNRS
Clermont-Ferrand, France

ABSTRACT

We deal here with electric vehicles, provided in energy by a local photovoltaic micro-plant, with limited storage and time-dependent production capacities. Our goal is to synchronize energy production and consumption. Because of the complexity of resulting bi-level model, we handle it by short-cutting the production level through surrogate estimators, whose values are computed with the help of flexible pricing and machine learning devices.

Keywords

Operational Research, Combinatorial Optimization, Machine Learning

1. INTRODUCTION

Multi-level decisional [1] models usually involve several players, tied together by some hierarchical or collaborative links. They aim at providing scenario which would be the best in case all the players accept a common rule (centralized paradigm), or at searching for a compromise (collaborative paradigm). Standard approaches involve decomposition schemes, hierarchical (Benders, Stackelberg,...) or transversal (Lagrangean). Main difficulties are related to the retrieval of information from the different levels in order to make them interact, and to the collaborative issue, which may impose the players to deal with incomplete information. A trend, boosted by the rise of machine learning technology [5], is to bypass some levels and replace them by surrogate constraints or estimators. We follow this trend here while dealing with the joint management of local photovoltaic energy [3] production and its consumption by a fleet of electric vehicles [2, 4]. This problem arose in the context of the activities of IMOBS3 (*Innovative Mobility*) Labex in Clermont-Ferrand, which conducts research on both autonomous electric vehicles and solar energy, and of the national PGMO program promoted by power company EDF.

So we consider here a fleet of K small identical electric vehicles, initially located at a depot $Depot = 0$, and required to perform VRP: *Vehicle Routing Problem* tours, that means to visit a set of stations $J = \{1, \ldots, M\}$ within a time horizon $[0, TMax]$. Moving from station j to station k requires $\Delta_{j,k}$ time units and an amount $E_{j,k}$ of energy. Recharge *transactions* take place at the depot. An *Elementary Trip* is any VRP sub-tour that a vehicle may perform without recharging at the depot. The fleet relies on a set B of identical batteries, with capacity C and charge speed C^S, initially located at *Depot*, and vehicles switch their battery every time they come back to *Depot*. This plug *out/in* operation is instantaneous and avoids that the vehicle waits for recharging. It comes that while the vehicles are running with *active* batteries, *idle* batteries are recharged at *Depot* before being used again by the vehicles. For any battery b in B, V_b denotes the energy load of b at time 0.

In order to implement a *self-consumption* policy, *Depot* is provided with a *PV-Plant*, that means a photovoltaic facility which assigns the batteries to the vehicles and produces energy that it distributes between the currently idle batteries or that it sells to the market. In case this energy is not enough, the *PV-Plant* can also buy energy to the market. The time space $[0, TMax]$ is divided into small periods $i = 1, ..., N$, all with same length p. We denote by C^R, the recharge per period capacity, that means the quantity $p \cdot C^S$ of energy which may be loaded into a battery during a period. We also denote by R_i the expected production of the *PV-Plant* at period i, by A_i the energy unit purchase price at period i, and by B_i the energy unit sale price. For technical reasons, a battery switch takes place only at the junction between 2 periods, that means at a time $t = p \cdot i, i = 0, \ldots, N$. So resulting *PV_Prod_VRP* decision problem, represented in the Figure 1, comes as follows:

PV_Prod_VRP: Simultaneously schedule the vehicles and the PV-Plant, in such a way that:

- Every station is visited at least once by the fleet;
- Every time a vehicle k comes back to *Depot*, it is assigned a battery charged in such a way that it will make possible its next elementary trip;
- The global energy load of the batteries does not to decrease between the beginning and the end of the process.
- Some global cost is minimized, which combines standard VRP cost with the PV-Plant cost of energy self-

consumption.

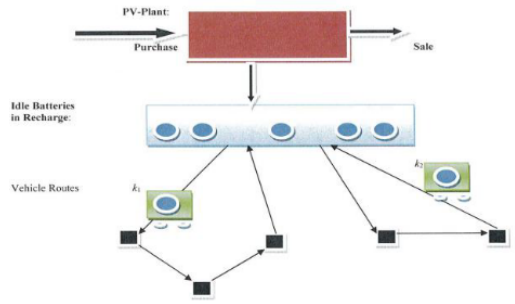

Figure 1: The *PV-Plant*, the batteries and the vehicle fleet.

In order to formalize, let us suppose that a collection Π_0 of *elementary trips* π has been computed and that every such a trip π has been scheduled inside a set of consecutive periods $I(\pi)$. We denote by $\Sigma_0 = \{(\pi, I(\pi)), \pi \in \Pi_0\}$ the resulting set of *scheduled trips* $\sigma = (\pi, I(\pi)$. For any trip π we denote by $E(\pi)$ the energy that its requires, by $T(\pi)$ its duration, and by $S(\pi)$ the set of stations that it visits. We extend those notations to scheduled trips σ and set $E^{Mean}(\sigma) = E(\sigma)/Card(I(\sigma))$. Then, *PV_Prod* resulting sub-problem is about the way the PV-Plant loads the batteries and assign them to *scheduled trips* σ.

PV_Prod(Σ_0) MILP (*Mixed Integer Linear Program*) *Model:*
Compute:

- $X^A = (X^A_i, i = 1, ..., N)$, $X^B = (X^B_i, i = 1, ..., N)$ and $X^D = (X^D_{b,i}, b \in B, i = 1, ..., N)$: respectively the energy amount bought on the market, sold to the market, and distributed to battery b by PV-Plant;
- $W = (W_{b,i}, b \in B, i = 0, ..., N)$: the amount of energy inside the battery b at the period i;
- $U = U_{\sigma,b}, \sigma \in \Sigma_0, b \in B : U_{\sigma,b} = 1$ if the battery b is assigned to the process σ;
- $\delta = \delta_{b,i}, b \in B, i = 1, ..., N : \delta_{b,i} = 1$ if the battery b is idle at period i.

Objective function: $\sum_i (A_i \cdot X^A_i - B_i \cdot X^B_i)$.
Constraints:

- For any $b, i : W_{b,i} \leq C$ and $X^D_{b,i} \leq C^R \cdot \delta_{b,i}$; (R1)
- For any $b : W_{b,0} = V_b$; (R2)
- $\sum_b W_{b,N} \geq \sum_b V_b$; (R3)
- For any $i : R_i + X^A_i = X^B_i + \sum_b X^D_{b,i}$; (R4)
- For any $b, i : (1 - \delta_{b,i}) = \sum_{\sigma \ s.t. \ i \in I(\sigma)} U_{\sigma,b} \leq 1$; (P1)
- For any $\sigma \in \Sigma_0 : \sum_b U_{\sigma,b} = 1$. (P2)
- For any $b, i : W_{b,i} = W_{b,i-1} + X^D_{b,i} - \sum_{\sigma \ s.t. \ i \in I(\sigma)} E^{mean}(\sigma) \cdot U_{\sigma,b}$; (P3)

Explanation: (R1): We charge a battery b only if it is idle. (R3): The batteries must be globally loaded with at least as much energy at the end of the whole process as at the beginning. (R4) tells the way energy is distributed between sale, purchase and battery loading. (P1): b is active at period i only if has been assigned to a unique *scheduled trip* σ active at period i. (P2): any *scheduled trip* σ is assigned a unique battery b. (P3) describes the evolution of a battery b from a period $i-1$ to next period i.

Any VRP decision means a collection Σ_0 of *scheduled trips* $\sigma = (\pi, I(\pi))$ such that:
- For any period i, $Card(\{\sigma \in \Sigma_0 \ s.t. \ i \in I(\sigma)\}) \leq K$; (S1)
- For any station j, $Card(\{\sigma \in \Sigma_0 \ s.t \ j \in S(\sigma)\}) \geq 1$; (S2)

If we consider as standard VRP cost of Σ_0 the global riding time $\sum_\sigma T(\sigma)$ *(Driver Cost)* then, a *time versus money* coefficient α being given, a bi-level setting of our *PV_Prod_VRP* problem comes as follows:

PV_Prod_VRP Problem: Compute a collection Σ_0 of *scheduled trips*, such that (S1, S2) hold and which minimizes the sum $\alpha \sum_\sigma T(\sigma) + Val\_PV\_Prod(\Sigma_0)$ where Val_PV_Prod is the optimal $PV\_Prod(\Sigma_0)$ value.

2. HANDLING PV_PROD_VRP WITH SURROGATE COMPONENTS

MILP formulation of *PV_PROD_VRP* is hardly practicable and does not fit uncertain or collaborative contexts. So we try:

- First approach: It aims at benchmarking. We solve *PV_Prod_VRP* through the MILP model restricted to a set Σ_0 of *scheduled trips*, pre-computed through successive applications of a randomized simple greedy procedure.
- Second Approach: We short-cut the *PV_Prod* level through the introduction of surrogate constraints and estimators.

2.1 Solving the PV_Prod_VRP bi-level model, while partially short-cutting the slave PV_Prod level.

Our purpose here is to compute the scheduled trip set Σ_0 without involving the PV-Plant. But, while the schedule $\sigma \rightarrow I(\sigma)$ must take into account prices A_i, B_i and production rates R_i, we can only say that a well-fitted collection Σ_0 requires small amounts of both time and energy. This suggests that we should distinguish between the design of the trip collection Π_0 and the way we turn it into a collection Σ_0 of *scheduled trips*. This leads us to the following parametric *VRP_Surrogate* process:

***VRP_Surrogate* Parametric Algorithm:**
Initialize flexible scaling parameter γ; Not Stop; current best solution *Best_Sol* is undefined;
While Not *Stop* do
1st step: Compute an *ad hoc* elementary trip collection Π_0 which minimizes $\alpha \cdot \sum_{\pi \in \Pi_0} T(\pi) + \gamma \cdot \sum_{\pi \in \Pi_0} E(\pi)$;
2nd step: Turn Π_0 into a *scheduled trip* collection Σ_0, that means compute the period sets $I(\pi), \pi \in \Pi_0$ in such a way that some surrogate constraints (SURR1) be satisfied and that some surrogate cost $\Phi(\pi \rightarrow I(\pi)$ be minimized;
3rd step: Update γ and *Stop*; Solve $PV\_Prod(\Sigma_0)$ and update *Best_Sol*.

2.2 Dealing with Step 1: A Branch and Cut Algorithm based upon Strong No Subtour constraints

For any subset A of the station set $J = \{1, \ldots, M\}$, we set: $\delta^+(A) = \{$arcs (j, k) such that $j \notin A$ and $k \in A\}$ and

$Cl(A) = \{$ arcs $e = (j,k)$ s. t. at least j or k is in $A\}$. Then the auxiliary *Elementary_Trip* ILP model comes as follows:

Elementary_Trip ILP model:
Compute a (0, 1)-valued vector $Z = (Z_{j,k}, j, k = 0, ..., M)$ in such a way that :

- For any j in $\{1, ...M\} : \sum_k Z_{j,k} = \sum_j Z_{j,k} = 1$; (S2)
- For any subset A of $\{1, ...M\}$:
  $C \cdot \sum_{(j,k)\in\delta^+(A)} Z_{j,k} \geq \sum_{(j,k)\in Cl(A)} E_{j,k} \cdot Z_{j,k}$; (SNS)
- Minimize $\alpha \cdot (\sum_{j,k} Z_{j,k} \cdot T_{j,k}) + \gamma \cdot (\sum_{j,k} Z_{j,k} \cdot E_{j,k})$.

Explanation: Above SNS: *Strong No Sub-Tour* constraints not only forbid sub-tours in the usual sense, but also ensure us that vector Z represents a collection of *elementary trips*, that means of routes π from *Depot* to *Depot* such that $E(\pi) \leq C$.

Theorem 1: *{0, 1} vector Z satisfies (S2, SNS) if and only if arcs (j, k) such that $Z_{j,k} = 1$ define a collection Π_0 of sub-tours $\pi_1, ..., \pi_S$ with $S = \sum_k Z_{0,k}$ such that:*

- *For every $s = 1, ..., S, \pi_s$ starts from Depot = 0 and ends into Depot, and spends less than C energy;*
- *Every station is visited exactly once by collection Π_0.*

Constraints SNS may be separated in polynomial time through application of a max flow (min cut) procedure.

Sketch of the Proof: (S2, SNS) imply that Z gives rise to a collection τ of sub-tours $\tau_0, ..., \tau_S$, and that if some tour τ_S spends more energy than capacity C, then a subset A of $\{0, \ldots, M+1\}$ exists which makes Z violate (SNS). We get our first statement. As for the second, we see that, Z (integral or rational) being given, separating (SNS) means searching for $A \subseteq \{1, ..., M\}$ and $B = \{1, ..., M\} - A$, such that (*):
$\sum_{j,k\in B} Z_{j\cdot k} \cdot E_{j,k} + C \cdot \sum_{(j,k)\in\delta^+(B)} Z_{j,k} < \Delta = \sum_{j,k} \cdot E_{j,k}$
So we construct a network G^{Aux}, whose node set is $\{0, 1, ...,$ $M+1\}$ and whose arc set may be written as $U^{Aux} = U \cup Copy(U)$ with:

- $U = \{$all pairs $(j,k), j, k = 0, ..., M\}$ s.t $Z_{j,k} \neq 0$: such an arc $u = (j,k)$ is provided with a capacity $w_u = Z_{j,k} \cdot (C - E_{j,k})$;
- With any arc $e = (j,k)$ in U, we associate an arc $u = Copy(e) = (j, M+1)$: such an arc $u = Copy(e) = (j, M+1)$ is provided with a capacity $w_u = Z_{j,k} \cdot E_{j,k}$. Then arc set *Copy(U)* is the set of all arcs *Copy(e)*, $e \in U$.

We conclude, by checking that searching for B such that (*) holds is equivalent to solving a Max Flow problem in G^{Aux}. □

Theorem 1 and related proof provide us with an efficient *separation* procedure which opens the way to the implementation of a *Branch and Cut* process. Still, because such a *Branch and Cut* approach remains difficult to use in case of large size instances and in case of uncertainty, we also implement a heuristic *Insertion/Removal (Build/Destroy)* algorithm.

2.3 Dealing with Step 2: Surrogate Components

In order to enhance $PV\_Prod(\Sigma_0)$ feasibility, we impose the following surrogate necessary (but not sufficient) constraints:

- For any period i: $Card(\{\sigma \in \Sigma_0$ such that $i \in I(\sigma)\}) \leq K$; (S1)
- For any subset $i_0 = 1, ..., N : C^R \cdot \sum_{i\leq i_0-1} n(\Sigma_0, i) + \Sigma_b V_b \geq \sum_{\sigma \ s.t.\ Start(\sigma)\leq i_0} E(\sigma)$, where $n(\Sigma_0, i)$ is the number of *scheduled trips* σ idle at period i, and $Start(\sigma)$ is the starting period of σ. (SURR1)

The constraint (SURR1) means that we must be able to feed the batteries in such a way that trips becomes possible. Then, in order to make possible the use of any surrogate estimator $\Phi(\pi \to I(\pi))$, we implement Step 2 while relying on a non deterministic local search heuristic $Scheduled\_Trip(\Pi_0, \Phi)$. So, what remains to be done is to discuss estimator Φ.

Defining $\Phi(\Sigma_0)$ according to a Pricing Mechanism. The idea here is that the cost of a schedule $(\pi \to I(\pi))$ is determined by the distribution of above defined values $n(\Sigma_0, i)$. Let $E = \sum_{\pi\in\Pi_0} E(\pi)$ be the global charge loaded into the batteries and $I = \sum_{\pi\in\Pi_0} \lceil T(\pi)/p \rceil$ be the number of periods required by the trips of Σ_0. If all batteries receive a same charge $E^{Mean} = E/I$ at every period when they are idle, then the cost of the production process is $I \cdot Q^{Stand}_{i,n(\Sigma_0,i)}$, where $Q^{Stand}_{i,n} = A_i \cdot (n \cdot E^{Mean} - R_i)$ if $n \cdot E^{Mean} \geq R_i$ and $Q^{Stand}_{i,n} = B_i \cdot (n \cdot E^{Mean} - R_i)$ else. This suggests us to express the surrogate cost $\Phi(\pi \to I(\pi))$ involved into the *VRP_Surrogate* algorithm as a sum $\sum_i Q_{i,n(\Sigma_0,i)}$, where $Q_{i,n}$ is the estimation of the cost induced by n batteries in recharge (idle) at period i. We notice that if $n(\Sigma_0, i) \cdot E^{Mean} \geq R_i$, then $Q_{i,n}$ should increase with A_i and that if $n(\Sigma_0, i) \cdot E^{Mean} < R_i$, then $Q_{i,n}$ should decrease as B_i increases. This suggests to set:

- A^{Mean} = mean value $A_i, i = 1, ..., N$; B^{Mean} = mean value $B_i, i = 1, ..., N$;
- $Q_{i,n} = Q^{Stand}_{i,n} \cdot (1 + \rho_1(A_i - A^{Mean}))$ if $n(\Sigma_0, i) \cdot E^{Mean} \geq R_i$, and $Q_{i,n} = Q^{Stand}_{i,n} \cdot (1 + \rho_2(B_i - B^{Mean}))$ else, where ρ_1 and ρ_2 are 2 non negative flexible parameters.

Computing $\Phi(\Sigma_0)$ through a Neural Network *N_Energy*. Instead of relying on energy price coefficients $Q_{i,n}$, we use a neural network *N_Energy* in order to provide us with the quality of a *scheduled trip* collection Σ_0. *N_Energy* is implemented with the help of the TensorFlow open software and trained with a large number (4000) of $PV\_Prod(\Sigma_0)$ instances. It is designed as a *convolutional* neural network. Such a network, whose main purpose is to be adaptable to inputs with flexible sizes, usually works in 2 (or more) steps: In the first step, a same standard perceptron called *convolutional mask* is applied to fixed size neighbours of the components of the input vector $IN = (IN_m, m \in M)$, and yields an output vector $OUT = (OUT_m, m \in M)$. In the next step, a pooling mechanism is applied to OUT, in order to compact it into the fixed size input of another perceptron which computes the final output. In the present case this final output is a number Θ between 0 and 1, such that the optimal value $VAL\_PV\_Prod(\Sigma_0)$ of $PV\_Prod(\Sigma_0)$ may be written $VAL\_PV\_Prod(\Sigma_0) = Val\_Min + \Theta \cdot (Val\_Max - Val\_Min)$, where *Val_Max* and *Val_Min* are respectively

lower and upper easy to compute bounds of $VAL\_PV\_Prod(\Sigma_0)$. More precisely we homogenize any input Σ_0, A, B, R, V of PV_Prod as a vector IN, with $IN[i] = (A_i^*, B_i^*, R_i^*, \mu_i^*, Q_i, C^*, C^{R*})$ as follows:

- $A_i^* = A_i/A^{Mean}$; $B_i^* = B_i/A^{mean}$;
- R^{Mean} = Mean values of coefficients R_i;
- $\mu_i = \sum_{\sigma \ s.t. \ i \in I(\sigma)} E^{Mean}(\sigma)$; $\mu_i^* = \mu_i/R^{Mean}$;
- $R_I^* = R_i/R^{Mean}$; $Q_i = n(\Sigma_0, i)/Card(B)$;
- $C^* = C/R^{Mean}$; $C^{R*} = C^R/R^{Mean}$.

The *convolutional mask CM* works on any sub-vector $IN_i^* = (IN[i], \ldots, IN[i+4])$, which means an input with 35 input arcs. It contains 3 inner layers, respectively with sizes 8, 4 and 2, and ends into an output layer, with 1 input value OUT_i. This network is complete in the sense that all 322 synaptic arcs are allowed, together with standard biased sigmoid activation functions whose derivative value in 0 is equal to 1/2. The *pooling* mechanism works by merging consecutive values OUT_i into a single value, in such a way it yields an intermediate vector AUX, with 13 entries, all with values between 0 and 1, which we handle with a perceptron *N_Pool*, with intermediate layers with size 6 and 3, and a final layer with size 1. At the very end, we must learn 421 synaptic coefficients.

3. NUMERICAL EXPERIMENTS

Technical Context: We use libraries CPLEX12 (for ILP models) and TensorFlow/Keras (for Machine Learning).

Instances: The main characteristics of an instance are: N = Number of periods, M = Number of stations, S = Expected number of *elementary trips* involved into a VRP solution, Q = number of macro-periods, which are associated with general trends in the production rate and in the market prices, L = expected length of an *elementary trip*, μ = *Battery Stress* coefficient, β = *Recharge Stress* coefficient, H = *Production Stress* coefficient and α = *time versus money* coefficient. *Stress* is related to the difficulty that one may have in computing the solution of the instance, in such a way that decreasing the *Stress* parameters increases the difficulty. The Table 1 shows the characteristics of the instances used during the experimentation.

Table 1: Characteristics of the instances

Instance	N	M	S	Q	L	μ	β	H	α
1	40	50	20	4	5	2	1.5	1	1.2
2	40	75	25	5	5	1.5	2	1.5	0.8
3	40	100	32	4	3	2.5	1.5	0.8	1.2
4	60	150	42	3	8	3	3	0.5	1.5
5	60	200	55	5	10	2	1.5	0.75	1.2

Outputs: For every instance, we apply the *VRP_Surrogate* resolution scheme while relying on both the pricing mechanism and the machine learning mechanism, performing the first *VRP_Surrogate* step through the *Insertion / Removal* algorithm. We denote by *W_Price* the value obtained with the pricing device and we denote by *W_ML* the value obtained while involving Machine Learning. *UB_G* denote an upper bound obtained through the first approach described in Section 2. The Table 2 shows these results. When the size of the instances increases, it tends to outperform the results produced by the global MILP model. Notice that the cost value of *PV_Prod_VRP* may be negative, due to the fact that the cost may become a profit. Also, we try

Table 2: Behavior of the Price and Machine Learning based *VRP_Surrogate* Algorithms

Inst.	UB_G	W_ML	W_Price
1	-235.91	-193.38	-116.28
2	-1311.60	-835.27	-1165.42
3	248.43	409.91	631.02
4	1760.33	1861.24	2138.38
5	3742.12	1602.39	1900.69

the *Branch and Cut* algorithm and the *Insertion / Removal* heuristic algorithm of Section 2.2 on the *Elementary_Trip* model, and retrieve in less that 1 CPU h:

- The lower and upper bounds *LB_BC* and *UB_BC* computed by the algorithm;
- The value *Relax* induced by the relaxation of the integrality constraint;
- The number *CUT* of *SNS* cuts generated during the Branch and Cut process;
- The value *W_Heur* computed by the heuristic algorithm.
- Related CPU time *Time* of the heuristic algorithm. The *Branch and Cut* algorithm was executed during 1 hour.

Related results are contained into the Table 3.

Table 3: Behavior of the *Branch and Cut* Algorithm

Inst.	LB_BC	UB_BC	Relax	CUT	W_Heur	Time
1	366.258	424.268	362.979	2917	428.08	1.19
2	362.481	470.942	359.901	904	470.50	5.41
3	421.34	528.725	420.884	712	534.73	10.38
4	1514.61	1809.3	1514.6	583	1776.16	28.30
5	2425.26	3372.26	2425.26	367	3025.93	41.79

Comments: Taken as a whole, solving *PV_Prod_VRP* while relying on surrogate component happens to be rather efficient. We also see that, even with *Strong No Sub-Tour* constraints, the *Elementary_Trip* model is difficult for large instances. The heuristic algorithm allows us to get close upper bounds to the *Branch and Cut* in reasonable time.

4. REFERENCES

[1] S. Dempe, V. Kalashnikov, Pérez-Valdés, and N. Kalashnykova. Bilevel Programming Problems Theory. *Springer*, 2015.

[2] T. Erdelić and T. Carić. A survey on the electric vehicle routing problem: variants and solution approaches. *Journal of Advanced Transportation*, 2019, 2019.

[3] S. Irani and K. R. Pruhs. Algorithmic problems in power management. *ACM Sigact News*, 36(2):63–76, 2005.

[4] G. Macrina, L. D. P. Pugliese, and F. Guerriero. The green-vehicle routing problem: a survey. In *Modeling and Optimization in Green Logistics*, pages 1–26. Springer, 2020.

[5] J. Wojtusiak, T. Warden, and O. Herzog. Machine learning in agent-based stochastic simulation: Inferential theory and evaluation in transportation logistics. *Computers & Mathematics with Applications*, 64(12):3658–3665, 2012.

Local reflection symmetry detection in Earth observation data

David Podgorelec
University of Maribor
Faculty of Electrical Engineering and Computer Science
Maribor, Slovenia
david.podgorelec@um.si

Luka Lukač
University of Maribor
Faculty of Electrical Engineering and Computer Science
Maribor, Slovenia
luka.lukac@student.um.si

Borut Žalik
University of Maribor
Faculty of Electrical Engineering and Computer Science
Maribor, Slovenia
borut.zalik@um.si

ABSTRACT

We propose a new algorithm which detects patterns with reflection symmetry in Earth observation data. It must consider approximate symmetries, as the acquisition of input datasets is not able to provide exact pairs of symmetric elements. Therefore, we look for symmetries between voxels, not between the input points. Furthermore, the nature of such data implies that the symmetric patterns in the top view are the most interesting and, thus, it suffices to detect symmetries with vertical symmetry planes. The symmetry detection may thus be split into horizontal voxel slices and the results with the same symmetry plane are then merged. At the end, the resulting symmetries are ranked with respect to the number of voxels involved. Early results obtained for some voxelisations of two LiDAR datasets of different sizes are promising both in terms of the detection speed and quality of solutions.

Categories and Subject Descriptors

I.3.5 [**Computing Methodologies**]: Computer Graphics—*Computational Geometry and Object Modeling*

General Terms

Algorithms, Performance, Theory

Keywords

Approximate symmetry, voxelization, line segment, merging

1. INTRODUCTION

An object is symmetric if there is a transformation (such as translation, scaling, rotation, reflection, etc.) that maps it onto itself [6]. The symmetry perception has been studied and demonstrated in humans, but also in birds, dolphins, apes and even honey bees [3]. It is assumed that symmetry perception has become an integral part of the individual's perceptual organization process during the evolution of visual systems where individual regularities have been selected on the basis of their relevance [9]. This explains why symmetry has always inspired people in different fields, including arts, architecture, biology, medicine, mathematics, and various engineering disciplines [1]. However, in contrast to natural, almost self-evident symmetry perception processes in living beings, computer-aided symmetry detection is anything but simple [1]. Machine learning approaches are increasingly popular as everywhere else [8], but they rely heavily on training datasets, which are usually incomplete. Traditional constructive approaches for individual types of symmetries, particularly the reflection and rotational one, still prevail. Symmetry can be global when it concerns the whole object, or local when only parts of the scene are incorporated. A local symmetry containing a single connected component is called partial symmetry. Furthermore, symmetry can be perfect (strong) or approximate (weak) [5]. Approaches of Žalik et al. [1] and Hruda et al. [4] are representatives of global reflection symmetry detection algorithms. Local (partial) symmetry is sometimes handled by decomposing the scene into individual parts and then detecting global symmetry separately on each of them [7]. Cailliere et al. [2] presented a true local reflection symmetry detection operating on triangular meshes. In 2006 already, Mitra et al. [5] presented a powerful general algorithm that detects different types of global and local symmetry.

In this paper, a new algorithm is presented, predominantly designed to detect local reflection symmetries in Earth observation (EO) data. Section 2 illustrates the overall idea of the proposed approach, while section 3 experimentally confirms its usability. Section 4 briefly summarizes the presented work and discusses future research challenges.

2. METHODOLOGY

The presented algorithm is predominantly designed to detect local reflection symmetries in EO and complementary geographic information systems (GIS) data. Current implementation reads LiDAR (light detection and ranging) point clouds stored in LAS files only, but these data are immediately voxelized and, thus, 3D or 2D raster grids may also be considered valid inputs. The algorithm is fully scalable, as the resolution of the voxelisation can vary from a few centimetres or even lower up to tens or hundreds of metres, depending on the data acquisition technology and the intended use. Furthermore, grayscale and colour raster im-

```
Algorithm: Local Reflection Symmetry Detection.
  Voxelization.
  For each horizontal slice of the voxel grid
    Identify LSs through interesting voxels.
    Cluster the LSs with respect to their lengths.
    For each cluster of LSs
      Establish basic symmetries among pairs of LSs.
Merge symmetries from different clusters and slices.
Postprocessing.
End.
```

Figure 1: Concept of the new algorithm.

ages may be processed if colours are interpreted as altitudes. Whether these input datasets are 2D or 3D, they are mostly acquired downwards from satellites, airplanes, drones, etc. This means that much more data are collected from the visible top surfaces than from the side and bottom surfaces. Furthermore, width and length of a considered geographic area are usually much greater than the range of altitudes, i.e. bigger geographic areas are relatively flat. All these findings have the following implications on symmetry detection:

- Due to sampling, points of an „original“ part and mirrored part rarely match exactly.
- Due to higher density of acquired data on visible top sides, it is more likely to detect symmetric parts there.
- Due to „flatness“ of acquired areas, it is more likely to explore symmetries from above than from side.

For these reasons, the algorithm is designed to detect locally symmetric patterns with approximate (and not the ideal) reflection symmetries with regard to vertical symmetry planes only. The latter implies that it is sufficient to detect symmetries in horizontal slices and then combine them on the basis of the common detected symmetry planes. This constraint crucially contributes to the affordable time complexity of the algorithm in the EO data domain, although the majority of the algorithm's steps can be easily generalized when we can afford a slower execution.

Bottom-up approach is another crucial feature of the proposed algorithm. The idea is to find basic symmetries first and then construct larger ones by merging. In the context of the reflection symmetry, the basic symmetry is the symmetry of two geometric primitives, while the construction means merging the symmetric pairs which share the same symmetry plane. Primitives may be points (voxels), line segments, or more complex structures, and our choice are line segments. Core idea is that each line segment which appears in some symmetry should have a symmetric pair (copy) or more of them with the same length somewhere (in the slice).

A rough outline of the algorithm is presented in Figure 1, and a more detailed explanation of the individual steps is given in the subsections that follow.

2.1 Voxelization

A user enters the total number of voxels, and the program computes subdivisions in each coordinate direction, where the ratios between the three sides of the bounding box are tried to be kept and, furthermore, the number of voxels in each coordinate direction must be above some user-defined threshold. Consequently, the actual number of voxels may be significantly lower than the value entered by the user.

The material voxels are then identified in the grid. Each material voxel contains at least one point from the input point cloud. Theoretically, the symmetries should be identified among these voxels, but this would result in numerous trivial solutions (e.g. “infinitely” many symmetries could be found on a flat surface), so we further reduce the set of candidates for the symmetry detection by extracting the so-called interesting voxels.

A voxel is considered interesting for further processing if its surroundings is not flat. We therefore test each material voxel against the patterns of 26 adjacent voxels and filter out the interesting ones which are not in the middle of a vertical, horizontal or diagonally slanted local surface. Such interesting voxels represent the input for the next step.

2.2 Clustering

Our algorithm is designed to determine the basic symmetries between pairs of LSs. Such a pair may be symmetric only if both LSs have the same length. Furthermore, only the LSs which mainly pass through the interesting voxels are considered in this step. It is worthless to compare two LSs of the same length where one lies on the objects' surfaces and the other penetrates the air. The default threshold for amount of material voxels in a regular LS is set to 80. Therefore, the “regular” LSs are extracted in this step and arranged into clusters due to their lengths.

2.3 Basic Symmetry Detection

As this step runs separately for each horizontal voxel slice, it appears quite straightforward. However, the distance between two voxels differs from the distance between two points inside these voxels, and the angle between two LSs defined by four points differs from the angle between two LSs defined by the centres of four voxels. Some user defined tolerances must be considered in this step. Smaller voxels decrease relevance of these tolerances, but they significantly increase the time complexity. Note that this is computationally most demanding task with the theoretical time complexity $O(n^4)$, where n is the number of voxels. The previous steps of extracting interesting voxels and grouping them into clusters not only prevent the calculation of trivial (meaningless) symmetries, but above all reduce the number of pairs of LSs that need to be compared here.

2.4 Merging

The previous step determines all symmetry planes between pairs of “regular” LSs. Next, we join all pairs that share a common plane of symmetry. This step simultaneously considers both the clusters in a single slice and the results in different slices, as seen in the concept from Fig. 1.

2.5 Postprocessing

After determining all symmetry planes and arranging all interesting voxels into individual symmetries, the method concludes its work by coupling the rest of the material (non-

Table 1: Results for two voxelizations of the Maribor Cathedral and two of the Slomšek Square

Measure	Cathedral	Cathedral	Square	Square
Points	11779	11779	35985	35985
Input voxels	500	1500	1000	2000
Voxels	384	1089	960	1680
Material	78	193	64	428
Interesting	37	64	129	132
Symmetries	45	152	1490	2340
Time [s]	0.38	1.86	48.73	55.93

interesting) voxels with respect to each symmetry plane. Furthermore, input points within the voxels participating in an individual symmetry may be checked, whether they have a mirrored sibling close enough on the other side of the symmetry plane. The same effect can be reached by increasing the number of voxels, which requires less effort for implementation but importantly slows down the performance. The classification of points on the two sides of the symmetry plane is based on a simple vector product test.

3. RESULTS

Two point clouds read from LiDAR LAS files were used to demonstrate the performance of the algorithm. The first one represents Slomšek Square with the Cathedral of Saint John the Baptist to the east, the Rectorate of the University of Maribor to the west, the Slovene National Theatre to the north, and the building of the main Maribor Post office to the south. The second file contains the Cathedral extracted from the first one. Experiments were carried out on a PC computer with Intel Core i7-5820K and 32GB DDR4 RAM. The algorithm was programmed in C++ in QtCreator 7.0.1. Two different voxelizations of the Maribor Cathedral and two voxelizations of the Slomšek Square point cloud were used in the measurements.

Fig. 2 shows the results for the strongest symmetry detected on the Maribor Cathedral point cloud, voxelized with 384 voxels (the input was set to 500). Red and blue points are those inside the detected symmetric pairs of voxels on both side of the symmetry plane (in black), while gray points are in material voxels not participating in this strongest symmetry. Fig. 3 shows only the points in the bottommost horizontal slice of voxels. By merging this slice and other 7 slices above it, the results from Fig. 2b are obtained. In another voxelization with 1089 voxels (the input se to 1500), the symmetry from Fig. 4 was detected as the strongest. Nearly the same symmetry was at the sixth place (out of 45) in the previous voxelization, while the strongest symmetry from Fig. 2 and Fig. 3 is the 11th strongest (out of 152) in this different voxelization. In the top part of Table 1, a process of decreasing the number of entities from the LiDAR points to interesting voxels is demonstrated. Instead of 11779 points, only 37 or 64 interesting voxels were used in the basic symmetry detection step in both voxelizations of the Cathedral.

Fig. 5 shows two different local reflection symmetries detected in the point cloud representing the Slomšek Square in Maribor. The first one (Fig. 5a) is the strongest one obtained from the voxelization with 960 voxels (the input set

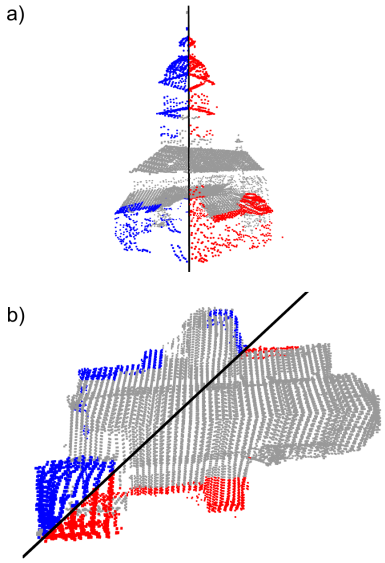

Figure 2: The Maribor Cathedral – the strongest symmetry in the 384 voxels grid: a) side view, b) top view.

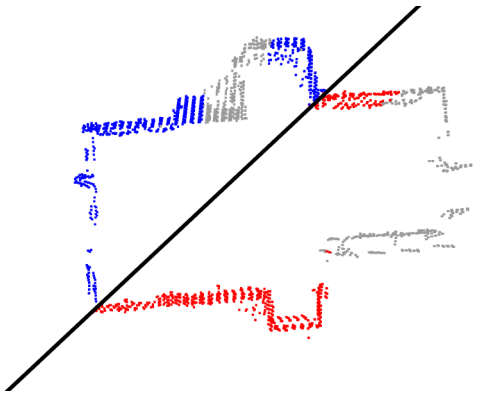

Figure 3: The Maribor Cathedral – the bottommost slice of 8 in the strongest symmetry in the grid of 384 voxels.

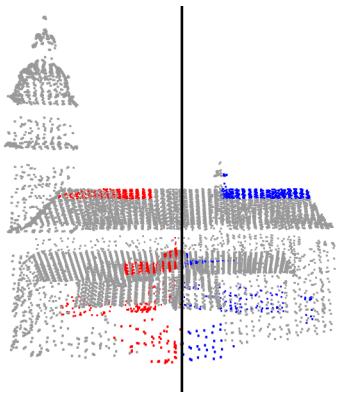

Figure 4: The Maribor Cathedral – the side view in the strongest symmetry in the grid of 1089 voxels.

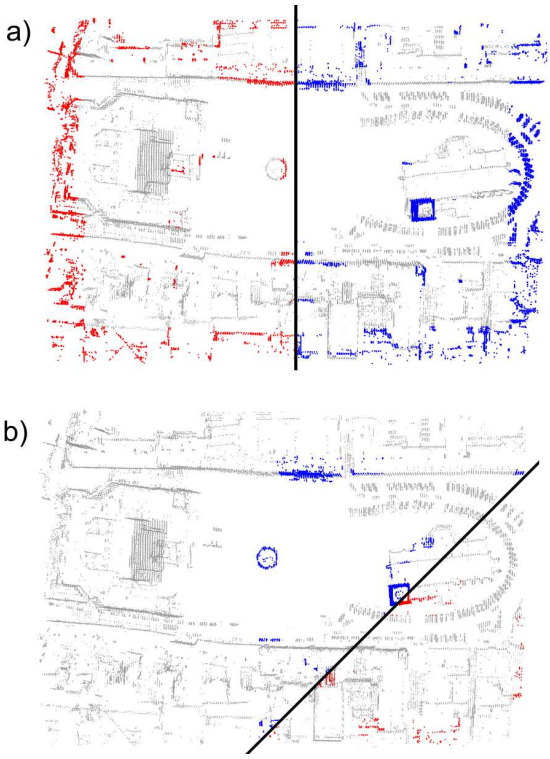

Figure 5: Slomšek Square in Maribor: a) the strongest symmetry out of 1490 in the grid of 960 voxels and b) the 68th strongest symmetry out of 2340 in the grid of 1680 voxels.

to 1000), and the bottom one (Fig. 5b) was obtained from the voxelization with 1680 voxels (the input set to 2000). Relatively low number of red in blue points indicates that this symmetry (Fig. 5b) is not among the strongest – it is indeed at the 68th place out 2340. Note that this example was not chosen at random. Namely, the symmetry plane here coincides with that of the Cathedral in Fig. 2.

In each voxelization, the algorithm detects as many symmetries as possible. They are ranked with respect to the number of voxels in a symmetry. Table 2 shows indicators of the three strongest and the weakest symmetries for the Cathedral with 384 voxels and Slomšek Square with 1680 voxels. The former is described wit 11,779 LiDAR points and with 78 material voxels, while the latter has 35,985 points and 428 material voxels.

We have also carried out experiments with bigger point clouds. Of course, the execution time and the quality of detected symmetries depend on voxelization. For an urban area about 20 times the size of Slomšek Square, described by half a million points and voxelised with 5000 voxels, the algorithm took just over 1 hour.

4. CONCLUSIONS

In this paper, we introduce a new algorithm for local reflection symmetry detection. It was predominantly designed for EO data processing, where it typically suffices to detect symmetries with vertical symmetry planes. The algorithm

Table 2: Number of points and voxels in individual symmetries and the proportion of the latter (%) among the material voxels

	The Cathedral			Slomšek Square		
	Points	Voxels	%	Points	Voxels	%
Best	2493	26	33.33	7489	133	31.07
2nd	2129	18	23.08	5715	116	27.10
3rd	779	10	12.82	7428	106	24.77
Last	174	4	5.13	131	4	0.93

first voxelizes the point cloud, extracts the so-called interesting voxels, and then finds basic symmetries between pairs of line segments of the same length, separately in each horizontal voxel slice. Basic symmetries sharing the symmetry plane are then merged into larger ones. The first results are promising, but there is a plenty of work left in order to ultimately affirm the method. The code and particularly some data structures must be optimized for faster performance. Individual voxel slices could be easily processed in parallel. Besides this, the inputs and the results must be filtered to eliminate connected parts with the number of voxels below some threshold. Finally, the algorithm must be evaluated in comparison to state-of-the-art methods.

5. ACKNOWLEDGMENTS

This research was funded by Slovene Research Agency under Research Project N2-0181 and Research Programme P2-0041.

6. REFERENCES

[1] B. Žalik, D. Strnad, S. Kohek, I. Kolingerová, A. Nerat, N. Lukač, and D. Podgorelec. A hierarchical universal algorithm for geometric objects's reflection symmetry detection. *Symmetry*, 14(5), 2022.

[2] D. Cailliere, F. Denis, D. Pele, and A. Baskurt. 3d mirror symmetry detection using hough transform. In *2008 15th IEEE International Conference on Image Processing*, pages 1772–1775. IEEE, 2008.

[3] M. Giurfa, B. Eichmann, and R. Menzel. Symmetry perception in an insect. *Nature*, 382(6590):458–461, 1996.

[4] L. Hruda, I. Kolingerová, and L. Váša. Robust, fast and flexible symmetry plane detection based on differentiable symmetry measure. *The Visual Computer*, 38(2):555–571, 2022.

[5] N. J. Mitra, L. J. Guibas, and M. Pauly. Partial and approximate symmetry detection for 3d geometry. *ACM Transactions on Graphics (TOG)*, 25(3):560–568, 2006.

[6] M. Petitjean. A definition of symmetry. *Symmetry: Culture and Science*, 18(2-3):99–119, 2007.

[7] I. Sipiran, R. Gregor, and T. Schreck. Approximate symmetry detection in partial 3d meshes. In *Computer Graphics Forum*, volume 33, pages 131–140. Wiley Online Library, 2014.

[8] S. Tsogkas and I. Kokkinos. Learning-based symmetry detection in natural images. In *European Conference on Computer Vision*, pages 41–54. Springer, 2012.

[9] C. W. Tyler. Human symmetry perception. *Human symmetry perception and its computational analysis*, pages 3–22, 1996.

Approximate Keys and Functional Dependencies in Incomplete Databases With Limited Domains–Algorithmic Perspective

[Extended Abstract]

Attila Sali*
Alfréd Rényi Institute of Mathematics
and Department of Computer Science,
Budapest University of Technology and
Economics
Budapest, Hungary
sali.attila@renyi.hu

Munqath Alattar
ITRDC, University of Kufa, Iraq
munqith.alattar@uokufa.edu.iq

ABSTRACT

A possible world of an incomplete database table is obtained by imputing values from the attributes (infinite) domain to the place of `NULL` s. A table satisfies a possible key or possible functional dependency constraint if there exists a possible world of the table that satisfies the given key or functional dependency constraint. A certain key or functional dependency is satisfied by a table if all of its possible worlds satisfy the constraint. Recently, an intermediate concept was introduced. A strongly possible key or functional dependency is satisfied by a table if there exists a strongly possible world that satisfies the key or functional dependency. A strongly possible world is obtained by imputing values from the active domain of the attributes, that is from the values appearing in the table. In the present paper, we study approximation measures of strongly possible keys and FDs. Measure g_3 is the ratio of the minimum number of tuples to be removed in order that the remaining table satisfies the constraint. We introduce a new measure g_5, the ratio of the minimum number of tuples to be added to the table so the result satisfies the constraint. g_5 is meaningful because the addition of tuples may extend the active domains. We prove that if g_5 can be defined for a table and a constraint, then the g_3 value is always an upper bound of the g_5 value. However, the two measures are independent of each other in the sense that for any rational number $0 \leq \frac{p}{q} < 1$ there are tables of an arbitrarily large number of rows and a constant number of columns that satisfy $g_3 - g_5 = \frac{p}{q}$. A possible world is obtained usually by adding many new values not occurring in the table before. The measure g_5 measures the smallest possible distortion of the active domains. We study complexity of determining these approximate measures.

*Research of the second author was partially supported by the National Research, Development and Innovation Office (NKFIH) grants K–116769 and SNN-135643. This work was also supported by the BME- Artificial Intelligence FIKP grant of EMMI (BME FIKP-MI/SC) and by the Ministry of Innovation and Technology and the National Research, Development and Innovation Office within the Artificial Intelligence National Laboratory of Hungary.

Categories and Subject Descriptors

H.2 [**Database Management**]: Miscellaneous; F.2 [**Analysis of Algorithms and Problem Complexity**]: Miscellaneous

General Terms

Theory

Keywords

Strongly possible functional dependencies, Strongly possible keys, incomplete databases, approximate functional dependencies, approximate keys.

1. INTRODUCTION

The information in many industrial and research databases may usually be incomplete due to many reasons. For example, databases related to instrument maintenance, medical applications, and surveys [8]. This makes it necessary to handle the cases when some information missing from a database and are required by the user. Imputation (filling in) is one of the common ways to handle the missing values [13].

In the present paper the classical relational model is considered that is the underlying concept of practical SQL database systems. The database is considered as a table, where rows (tuples) represent individual records, while columns correspond to properties or attributes. Important properties of these tables are the integrity constraints they (must) satisfy. In particular, keys and functional dependencies are the most common ones of those. An attribute set is a key if it determines all other attribute values in individual records, while functional dependency $X \to Y$ means that the values in attributes of X determine the values in attributes of Y.

A new approach for imputing values in place of the missing information was introduced in [3], to achieve complete data tables, using only information already contained in the SQL table attributes (which are called the active domain of an attribute). Any total table obtained in this way is called a *strongly possible world*. We use only the data shown on the table to replace the missing information because in many cases there is no proper reason to consider any other attribute values than the ones that already exist in the table. Using this concept, new key and functional dependency constraints called strongly possible keys (spKeys) and strongly possible functional dependencies (spFDs) were defined in [5, 4] that are satisfied after replacing any missing value (`NULL`) with a value that is already shown in the corresponding attribute. In Section 2, we provide the formal definitions of spKeys and spFDs.

The present paper continues the work started in [5], where an approximation notion was introduced to calculate how close any given set of attributes can be considered as a key. A classical measure is the ration of tuples needed to be removed. Tuple removal may be necessary because the active domains do not contain enough values to be able to replace the `NULL` values so that the tuples are pairwise distinct on a candidate key set of attributes K. In the present paper, we introduce approximation measures of spKeys and spFDs by adding tuples. Adding a tuple with new unique values will add more values to the attributes' active domains, thus some unsatisfied constraints may get satisfied. Adding tuples is only meaningful for strongly possible worlds. Earlier concept of possible worlds when any value of the attribute domain could be added is not appropriate, as adding tuples does not change the range of values usable for imputation. However, for strongly possible constraints the minimum ratio of tuples added to satisfy the constraint shows a smallest possible extension such that the constraint holds.

We denote by g_3 the minimum ratio of necessary tuple deletions and g_5 is the minimum ratio of necessary tuple additions. These two measures were shown to be basically independent of each other in [1]. In the present paper we review these results then turn our attention to interesting algorithmic and complexity problems involving the two approximation measures. The structure of the paper is as follows. Section 2 contains the necessary definitions, Section 3 discusses some related work. Section 4 reviews the definitions and main theorems about approximation measures. Subsection 4.2 contains the new results about complexity questions. Finally, Section5 includes some summary and concluding remarks.

2. DEFINITIONS

Let $R = \{A_1, A_2, \dots A_n\}$ be a relation schema. The set of all the possible values for each attribute $A_i \in R$ is called the domain of A_i and denoted as $D_i = dom(A_i)$ for $i = 1,2,\dots n$. Then, for $X \subseteq R$, then $D_X = \prod_{\forall A_i \in K} D_i$.

An instance $T = (t_1,t_2,\dots t_s)$ over R is a list of tuples such that each tuple is a function $t : R \to \bigcup_{A_i \in R} dom(A_i)$ and $t[A_i] \in dom(A_i)$ for all A_i in R. By taking a list of tuples we use the *bag semantics* that allows several occurrences of the same tuple. Usage of the bag semantics is justified by that SQL allows multiple occurrences of tuples. Of course, the order of the tuples in an instance is irrelevant, so mathematically speaking we consider a *multiset of tuples* as an instance. For a tuple $t_r \in T$ and $X \subset R$, let $t_r[X]$ be the restriction of t_r to X.

It is assumed that $\perp$ is an element of each attribute's domain that denotes missing information. t_r is called V-total for a set V of attributes if $\forall A \in V$, $t_r[A] \neq \perp$. Also, t_r is a total tuple if it is R-total. t_1 and t_2 are *weakly similar* on $X \subseteq R$ denoted as $t_1[X] \sim_w t_2[X]$ defined by Köhler et.al. [12] if

$$\forall A \in X \quad (t_1[A] = t_2[A] \text{ or } t_1[A] = \perp \text{ or } t_2[A] = \perp).$$

Furthermore, t_1 and t_2 are *strongly similar* on $X \subseteq R$ denoted by $t_1[X] \sim_s t_2[X]$ if

$$\forall A \in X \quad (t_1[A] = t_2[A] \neq \perp).$$

For the sake of convenience we write $t_1 \sim_w t_2$ if t_1 and t_2 are weakly similar on R and use the same convenience for strong similarity. Let $T = (t_1, t_2, \dots t_s)$ be a table instance over R. Then, $T' = (t'_1, t'_2, \dots t'_s)$ is a *possible world* of T, if $t_i \sim_w t'_i$ for all $i = 1, 2, \dots s$ and T' is completely `NULL` -free. That is, we replace the occurrences of $\perp$ with a value from the domain D_i different from $\perp$ for all tuples and all attributes. A active domain of an attribute is the set of all the distinct values shown under the attribute except the `NULL`. Note that this was called the *visible domain* of the attribute in papers [3, 4, 5, 2].

Definition 1. The *active domain* of an attribute A_i (VD_i^T) is the set of all distinct values except $\perp$ that are already used by tuples in T:

$$VD_i^T = \{t[A_i] : t \in T\} \setminus \{\perp\} \text{ for } A_i \in R.$$

To simplify notation, we omit the upper index T if it is clear from the context what instance is considered.

While a possible world is obtained by using the domain values instead of the occurrence of `NULL`, a strongly possible world is obtained by using the active domain values.

Definition 2. A possible world T' of T is called a *strongly possible world (spWorld)* if $t'[A_i] \in VD_i^T$ for all $t' \in T'$ and $A_i \in R$.

The concept of *strongly possible world* was introduced in [3]. A strongly possible worlds allow us to define *strongly possible keys (spKeys)* and *strongly possible functional dependencies (spFDs)*.

Definition 3. A strongly possible functional dependency, in notation $X \to_{sp} Y$, holds in table T over schema R if there exists a strongly possible world T' of T such that $T' \models X \to Y$. That is, for any $t'_1, t'_2 \in T'$ $t'_1[X] = t'_2[X]$ implies

$t'_1[Y] = t'_2[Y]$. The set of attributes X is a strongly possible key, if there exists a strongly possible world T' of T such that X is a key in T', in notation $sp\langle X\rangle$. That is, for any $t'_1, t'_2 \in T'$ $t'_1[X] = t'_2[X]$ implies $t'_1 = t'_2$.

If $T = \{t_1, t_2, \ldots, t_p\}$ and $T' = \{t'_1, t'_2, \ldots, t'_p\}$ is an spWorld of it with $t_i \sim_w t'_i$, then t'_i is called an *sp-extension* or in short an *extension* of t_i. Let $X \subseteq R$ be a set of attributes and let $t_i \sim_w t'_i$ such that for each $A \in R$: $t'_i[A] \in VD(A)$, then $t'_i[X]$ is an *strongly possible extension of* t_i *on* X *(sp-extension)*

3. RELATED WORK

Giannella et al. [9] measure the approximate degree of functional dependencies. They developed the IFD approximation measure and compared it with the other two measures: g_3 (minimum number of tuples need to be removed so that the dependency holds) and τ (the probability of a correct guess of an FD satisfaction) introduced in [11] and [10] respectively. They developed analytical bounds on the measure differences and compared these measures analysis on five datasets. The authors show that when measures are meant to define the knowledge degree of X determines Y (prediction or classification), then IFD and τ measures are more appropriate than g_3. On the other hand, when measures are meant to define the number of "violating" tuples in an FD, then, g_3 measure is more appropriate than IFD and τ. This paper extends the earlier work of [5] that utilized the g_3 measure for spKeys by calculating the minimum number of tuples to be removed from a table so that an spKey holds if it is not. The same paper proposed the g_4 measure that is derived from g_3 by emphasizing the effect of each connected component in the table's corresponding bipartite graph (where vertices of the first class of the graph represent the table's tuples and the second class represent all the possible combinations of the attributes' active domains). In this paper, we propose a new measure g_5 to approximate FDs by adding new tuples with unique values rather than deleting tuples as in g_3. In [14], Jef Wijsen summarizes and discusses some theoretical developments and concepts in Consistent query answering CQA (when a user queries a database that is inconsistent with respect to a set of constraints). Database repairing was modeled by an acyclic binary relation $\leq_{db}$ on the set of consistent database instances, where $r_1 \leq_{db} r_2$ means that r_1 is at least as close to db as r_2. One possible distance is the number of tuples to be added and/or removed. In addition to that, Bertossi studied the main concepts of database repairs and CQA in [6], and emphasis on tracing back the origin, motivation, and early developments. J. Biskup and L. Wiese present and analyze an algorithm called preCQE that is able to correctly compute a solution instance, for a given original database instance, that obeys the formal properties of inference-proofness and distortion minimality of a set of appropriately formed constraints in [7].

4. APPROXIMATION OF STRONGLY POSSIBLE INTEGRITY CONSTRAINTS

For examples of the following definitions see [1].

Definition 4. Attribute set K is an approximate strongly possible key of ratio a in table T, in notation $asp_a^- \langle K\rangle$, if there exists a subset S of the tuples T such that $T \setminus S$ satisfies $sp\langle K\rangle$, and $|S|/|T| \leq a$. The minimum a such that $asp_a^- \langle K\rangle$ holds is denoted by $g_3(K)$.

The g_3 approximation measure for spKeys was introduced in [5]. In this section, we introduce a new approximation measure for spKeys.

Definition 5. Attribute set K is an add-approximate strongly possible key of ratio b in table T, in notation $asp_b^+ \langle K\rangle$, if there exists a set of tuples S such that the table TS satisfies $sp\langle K\rangle$, and $|S|/|T| \leq b$. The minimum b such that $asp_b^+ \langle K\rangle$ holds is denoted by $g_5(K)$.

Definition 6. For the attribute sets X and Y, $\sigma : X \rightarrow_{sp} Y$ is a remove-approximate strongly possible functional dependency of ratio a in a table T, in notation $T \models\approx_a^- X \rightarrow_{sp} Y$, if there exists a set of tuples S such that the table $T \setminus S \models X \rightarrow_{sp} Y$, and $|S|/|T| \leq a$. Then, $g_3(\sigma)$ is the smallest a such that $T \models\approx_a^- \sigma$ holds.

Definition 7. For the attribute sets X and Y, $\sigma : X \rightarrow_{sp} Y$ is an add-approximate strongly possible functional dependency of ratio b in a table T, in notation $T \models\approx_b^+ X \rightarrow_{sp} Y$, if there exists a set of tuples S such that the table $T \cup S \models X \rightarrow_{sp} Y$, and $|S|/|T| \leq b$. Then, $g_5(\sigma)$ is the smallest b such that $T \models\approx_b^+ \sigma$ holds.

4.1 Relation between g_3 and g_5 measures

The following Proposition is used to prove Proposition 2.

PROPOSITION 1. *[1] Let T be an instance over schema R and let $K \subseteq R$. If the K-total part of the table T satisfies the key $sp\langle K\rangle$, then there exists a minimum set of tuples U to be removed that are all non-K-total so that $T \setminus U$ satisfies $sp\langle K\rangle$.*

PROPOSITION 2. *[1] For any $K \subseteq R$ with $|K| \geq 2$, we have $g_3(K) \geq g_5(K)$.*

Apart form the previous inequality, the two measures are totally independent for spKeys.

THEOREM 1. *[1] Let $0 \leq \frac{p}{q} < 1$ be a rational number. Then there exist tables over schema $\{A_1, A_2\}$ with arbitrarily large number of rows, such that $g_3(\{A_1, A_2\}) - g_5(\{A_1, A_2\}) = \frac{p}{q}$.*

Unfortunately, the analogue of Proposition 1 is not true for spFDs, so the proof of the following theorem is quiet involved.

THEOREM 2. *[1] Let T be a table over schema R, $\sigma : X \rightarrow_{sp} Y$ for some $X, Y \subseteq R$. Then $g_3(\sigma) \geq g_5(\sigma)$.*

Theorem 3 is proven by a construction [1] similar to the proof of Theorem 1.

THEOREM 3. *[1] For any rational number* $0 \leq \frac{p}{q} < 1$ *there exists tables with an arbitrarily large number of rows and bounded number of columns that satisfy* $g_3(\sigma) - g_5(\sigma) = \frac{p}{q}$ *for* $\sigma\colon X \to_{sp} Y$.

4.2 Complexity problems

Definition 8. The *SPKey problem* is the following.
Input Table T over schema R and $K \subseteq R$.
Question Is it true that $T \models sp\langle K\rangle$?
The *SPKeySystem problem* is the following.
Input Table T over schema R and $\mathcal{K} \subseteq 2^R$.
Question Is it true that $T \models sp\langle\mathcal{K}\rangle$?
The *SPFD problem* is the following.
Input Table T over schema R and $X, Y \subseteq R$.
Question Is it true that $T \models X \to_{sp} Y$?

The following was shown in [4].

THEOREM 4. *SPKey∈P, SPkeySystem and SPFD are NP-complete*

However, the approximation measures raise new, interesting algorithmic questions.

Definition 9. The *SpKey-g3* problem is the following.
Input Table T over schema R, $K \subseteq R$ and $0 \leq q < 1$.
Question Is it true that $g_3(K) \leq q$ in table T?
The *SpKey-g5* problem is the following.
Input Table T over schema R, $K \subseteq R$ and $0 \leq q < 1$.
Question Is it true that $g_5(K) \leq q$ in table T?

THEOREM 5. *Both, SpKey-g3 and SpKey-g5 are in P.*

PROOF. *SpKey-g3:* Let bipartite graph $G = (T, T^\star; E)$ be defined as in [4]. $T^\star$ is the set of sp-extensions of tuples in T on K, $\{t, t^\star\} \in E \iff t^\star$ is an extension of t. Then $T \setminus X \models sp\langle K\rangle$ iff there exists a matching covering $T \setminus X$ in G. Thus, X is a minimum set of tuples to be removed iff $|T \setminus X| = \nu(G)$, the matching number of G. This gives us $g_3(K) = \frac{|T| - \nu(G)}{|T|}$, so determination of $g_3(K)$ is equivalent with finding $\nu(G)$. The only problem is that $|T^\star|$ is usually of exponential size. In order to avoid exponential sized bipartite graph we only generate as many extensions as needed.For the sake of simplicity assume that $K = \{A_1, A_2, \ldots, A_b\}$. Let $T = \{t_1, t_2 \ldots t_m\}$ and $\ell(t_i) = |\{t^\star \in VD_1 \times VD_2 \times \ldots \times VD_b\colon t^\star \sim_w t_i[K]\}|$. Note that $\ell(t_i) = \prod_{j\colon t_i[A_j]=\perp} |VD_j|$, hence these values can be calculated by scanning T once and using appropriate search tree data structures to hold values of visible domains of each attribute. Sort tuples of T in non-decreasing $\ell(t_i)$ order, i.e. assume that $\ell(t_1) \leq \ell(t_2) \leq \ldots \leq \ell(t_p)$. Let $j = \max\{i\colon \ell(t_i) < i\}$ and $T_j = \{t_1, t_2, \ldots t_j\}$, furthermore $T_j^\star = \{t^\star : \exists t \in T_j : t^\star \sim_w t[K]\} \subseteq VD_1 \times VD_2 \times \ldots \times VD_b$. Note that $|T_j^\star| \leq \frac{1}{2}j(j-1)$. If $\forall i = 1, 2, \ldots, m\colon \ell(t_i) \geq i$, then define $j = 0$ and $T_j^\star = \emptyset$. Let $G^\star = (T_j, T_j^\star; E^\star)$ be the induced subgraph of G. Clearly $\nu(G) \leq \nu(G^\star) + |T \setminus T_j|$. On the other hand, a matching of G of size $\nu(G^\star) + |T \setminus T_j|$ can be created by extending a maximum matching of $G^\star$ greedily to the vertices (tuples) in $T \setminus T_j$.

SpKey-g5: To check whether $g_5(K) \leq q$ it is enough to add $\lfloor q|T|\rfloor$ pairwise distinct tuples with pairwise distinct new values and then check the satisfaction of $sp\langle K\rangle$ in polynomial time in the extended table. □

Example. Let $R = \{A_1, A_2, A_3\}$, $K_1 = \{A_1, A_2\}, K_2 = \{A_2, A_3\}$.

$T =$

	A_1	A_2	A_3
t_1	1	$\perp$	1
t_2	1	2	2
t_3	2	1	1
t_4	2	1	1

$T \setminus \{t_4\} \models sp\langle K_1\rangle$ and $T \setminus \{t_4\} \models sp\langle K_2\rangle$, but the spWorlds are different. In particular, this implies that for $\mathcal{K}$ we have $g_3(\mathcal{K}) > \max\{g_3(K)\colon K \in \mathcal{K}\}$ On the other hand, trivially $g_3(\mathcal{K}) \geq \max\{g_3(K)\colon K \in \mathcal{K}\}$ holds. This motivates the following theorem.

THEOREM 6. *Let Table T over schema R and $\mathcal{K} \subseteq 2^R$. The problem* Max-g3 *defined as*
Is $g_3(\mathcal{K}) = \max\{g_3(K)\colon K \in \mathcal{K}\}$?
is NP-complete.

PROOF. The problem is in NP, a witness consists of a set of tuples U to be removed, an index $j\colon \frac{|U|}{|T|} = g_3(K_j)$, also an spWorld T' of $T \setminus U$ such that each K_i is a key in T'. Verifying the witness can be done in three steps.

1. $g_3(K_j) \not\leq \frac{|U|-1}{|T|}$ is checked in polynomial time using Theorem 5.
2. For all $i \neq j$ check that $g_3(K_i) \leq \frac{|U|}{|T|}$ using again Theorem 5.
3. Using standard database algorithms check that $\forall i\colon K_i$ is a key in T'.

On the other hand, the SPKeySystem problem can be Karp-reduced to the present question as follows. First check for each $K_i \in \mathcal{K}$ separately whether $sp\langle K_i\rangle$ holds, this can be done in polynomial time. If $\forall i\colon T \models sp\langle K_i\rangle$ then give $\mathcal{K}$ and T as input for Max-g3. It will answer Yes iff $T \models sp\langle\mathcal{K}\rangle$. However, if $\exists i\colon T \not\models sp\langle K_i\rangle$, then give the example above as input for Max-g3. Clearly both problems have No answer. □

5. CONCLUSIONS

In the present paper we treat approximation measures of keys and functional dependencies in SQL database tables with null values. The strongly possible world semantics is used, that is only values from the active domains of attributes are allowed to be imputed in place of the null values. This semantics avoids unnecessary distortions of domains, since the active domain of an attribute is the set of values

that actually occur in the table. A classical approximation measure of an integrity constraint is that what percentage of the tuples must be deleted in order to that the remaining table satisfies the constraint. This is usually denoted by g_3. The strongly possible world semantics allows to introduce a new approximation measure, namely by adding new tuples the active domains can be extended and so the integrity constraints may be made valid in that way, as well. The minimum set of tuples to be added is a minimum extension of the current strongly possible world(s) to one, which satisfies the given constraint. The percentage of the necessary new tuples is denoted by g_5. It was shown earlier that for keys and functional dependencies $g_3 \geq g_5$, but otherwise they are independent. The approximation measures give rise to new algorithmic problems and complexity questions. The new results of the present note are about the complexity to determine $g_3(K)$ and $g_5(K)$ for a key constraint K. Also a natural decision problem about system of keys is investigated and it is shown to be NP-complete.

6. REFERENCES

[1] M. Al-Atar and A. Sali. Approximate keys and functional dependencies in incomplete databases with limited domains. In *Foundations of Information and Knowledge Systems 12th International Symposium, FoIKS 2022 Helsinki, Finland, June 20–23, 2022 Proceedings*, volume 13388 of *LNCS*, pages 147–167. Springer Nature Switzerland AG, 2022.

[2] M. Al-Atar and A. Sali. Strongly possible functional dependencies for sql. *Acta Cybernetica*, 2022.

[3] M. Alattar and A. Sali. Keys in relational databases with nulls and bounded domains. In *European Conference on Advances in Databases and Information Systems*, pages 33–50. Springer, 2019.

[4] M. Alattar and A. Sali. Functional dependencies in incomplete databases with limited domains. In *International Symposium on Foundations of Information and Knowledge Systems*, pages 1–21. Springer, 2020.

[5] M. Alattar and A. Sali. Strongly possible keys for sql. *Journal on Data Semantics*, 9(2):85–99, 2020.

[6] L. Bertossi. Database repairs and consistent query answering: Origins and further developments. In *Proceedings of the 38th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems*, pages 48–58, 2019.

[7] J. Biskup and L. Wiese. A sound and complete model-generation procedure for consistent and confidentiality-preserving databases. *Theoretical Computer Science*, 412(31):4044–4072, 2011.

[8] A. Farhangfar, L. A. Kurgan, and W. Pedrycz. A novel framework for imputation of missing values in databases. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, 37(5):692–709, 2007.

[9] C. Giannella and E. Robertson. On approximation measures for functional dependencies. *Information Systems*, 29(6):483–507, 2004.

[10] L. A. Goodman and W. H. Kruskal. Measures of association for cross classifications. *Measures of association for cross classifications*, pages 2–34, 1979.

[11] J. Kivinen and H. Mannila. Approximate inference of functional dependencies from relations. *Theoretical Computer Science*, 149(1):129–149, 1995.

[12] H. Köhler, U. Leck, S. Link, and X. Zhou. Possible and certain keys for sql. *The VLDB Journal*, 25(4):571–596, 2016.

[13] W. Lipski Jr. On databases with incomplete information. *Journal of the ACM (JACM)*, 28(1):41–70, 1981.

[14] J. Wijsen. Foundations of query answering on inconsistent databases. *ACM SIGMOD Record*, 48(3):6–16, 2019.

Building energy demand regression

Tamás Storcz
Department of Systems and Software Technologies, University of Pécs, Boszorkány u. 2, 7624 Pécs, Hungary,
storcz.tamas@mik.pte.hu

István Kistelegdi
University of Pécs, János Szentágothai Research Centre, Energy Design Research Group, Ifjúság u. 20, 7624 Pécs, Hungary,
kistelegdisoma@mik.pte.hu

Zsolt Ercsey
Department of Systems and Software Technologies, University of Pécs, Boszorkány u. 2, 7624 Pécs, Hungary,
ercsey.zsolt@mik.pte.hu

ABSTRACT

In the paper the applicability of regression models for building heating energy estimation is examined. During the experiment, regression models were created to estimate annual heating energy demand of generic family houses. Non-linearity of regression models was enhanced by creating non-linearly correlated new input variables. Then performance of generated models was measured and compared. As a result, multilayer dense neural net model with original input parameters was proposed. Its performance was almost equal to linear regression with extended input variables, but its structural and functional flexibility makes the neural network applicable in wider range of tasks.

Keywords

Heating energy, Regression, Neural network, Regression tree

INTRODUCTION

Heating energy demand determination by simulation for a family house by a special energetic simulation software requires time consuming setup and calculation per each building configuration. For energy optimization, a lot of simulations must be done, therefore speeding up the process would be very helpful.

Investigated simulations apply the same, generic engineering setup, since finding optimal building configurations and construction parameters are in the focus of the experiment series. Therefore, thermal insulation is, but artificial heating type and organization is not taken into consideration. Based on priorities specified by architect experts, from the collection of optimization aims, annual heating energy demand minimization comes first.

Complex and computation-intensive simulations could be replaced by multivariate linear regression. First test resulted bad regression accuracy, because as expected, the estimated energy demand function is not linear. But when increasing non-linearity of the model by enhance it with max. 3rd power of multiplicative combination of input variables, the accuracy of multivariate linear regression model grew above 0.95 of R^2 points, which is high enough to be accepted by architect experts.

But increasing the number of input variables in a non-linear extent, makes the model much more complex meanwhile the new input features are not interpretable by experts.

The proposed dense neutral network-based regression model is generated in 346 seconds. The created model calculates 3500 (train) estimations in 0.12 second and 1500 (test) estimations in 0.05 seconds. Its performance in accuracy, 0.96 R^2 is also acceptable by architect experts. The model operates on the initial input variables, no need for non-linearity addition by increasing number and complexity of input. Required non-linear features were extracted by the network in the training process. Besides, its structural flexibility opens possibilities of extended applications.

ENERGY REGRESSION

The main aim of regression models is to approximate unknow or known but complex correlation of descriptor and response variables. Such procedures are applied in all fields of science; thus, their application is not new in architectural energetics.

Peña-Guzmán and Rey [1] applied several types of linear regression models to estimate future development of residential electric power consumption with higher than 0.93 R^2 accuracy.

Mehedintu et al. [2] also used R^2 score for efficiency measurement of polynomial and auto regression methods. They applied these regression methods estimate the rate of total energy consumption and its part from renewable sources with higher than 0.91 score.

Mohammed et al. [3] estimated energy demand of school facilities. For the creation of linear regression model, 350 samples were used for training and 35 for testing. The accuracy of generated model was higher than 90%.

REGRESSION MODELS

Descriptive data
Building configurations can be used as input variables only if an appropriate representation is found for the blueprints. To help creation of this representation, the building is created from equal size boxes joined on sides conforming predefined architectural rules.
Figure 1. shows a valid building configuration.

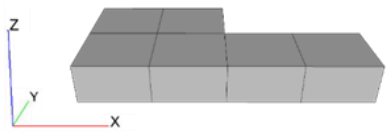

Figure 1: Building configuration example

Multivariate regression models cannot handle close relation of input variables (like coordinate triplets) therefore instead of using 3D coordinates of building components, count of details with special properties (convex edges and vertices, side walls, floors, roofs, etc.) were used as independent input variables.

Regression is a widely used statistical method [4], in which the aim is to model the relation between the descriptors and the dependent variable.

Linear regression
Linear regression [5] is a special case of generic regression, in which dependent variable is generated as linear combination of descriptor variables, as first-order Taylor series. For parameter determination in model generation, most commonly the least squares method is used.
Heating energy demand of family houses is a non-linear function. Definition of linear regression method enables dependency between its input variables, therefore multiplicative combination of inputs can be added to increase input complexity. Difficulty of this is the determination of maximum power of these combinations. As stated in the results, application of maximum 3rd power provides more than 0.95 of R^2 score.

Decision tree
Regression trees are special decision trees [6] for regression tasks. A decision tree is generated by a recursive binary partitioning process, which results internal nodes of the tree as decision nodes. These contain a test of the value of a specific input variable. The terminal nodes are the predicted output variable values.
Keys of regression tree generation are first the selection of the input variable for the decision – what to test. Next the selection of the separator value to test for. The idea behind the selection is the minimalization of variance of subspaces resulted by the split of decision.

For measuring the variances, the following methods are used:

- Mean Absolute Error based on L1 distance
- Mean Squared Error based on L2 distance
- Poisson method

Finally for the terminal nodes the predicted response value must be selected. This is done by averaging the group of samples covered by the terminal node.

Benefits of regression trees:

- greedy algorithm
- results good approximation
- short execution time – low resource needs
- well understandable decisions

Drawbacks of regression trees:

- not robust – sensitive to training data changes
- creation of optimal tree is NP-complete
- high chance for overfitting

Dense neural net

Neural net regression model [7] is inspired by nerve system and based on universal approximation theorem.

In terms of its structure, consists of independent analogue processing units which are organized into connected layers. The first layer of the network is called input, the last is the output, and layer between them (if any) is called hidden layer. Figure 2. shows the schematic model of a network with one hidden layer and l, m, n processors in input, hidden and output layers.

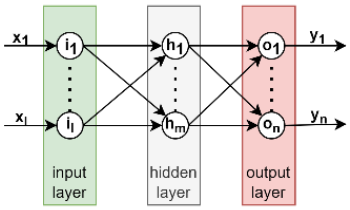

Figure 2: Neural net with 3 layers

Processors of consecutive layers are fully connected, but there are no connections between processors of same or non-consecutive layers.
The individual elements first calculate weighted sum of their inputs, then generate their output using an activation function. For providing the non-linearity and to support the learning procedure, special activations could be chosen.

MODEL GENERATION

For the model generation, 5010 samples were created by IDA ICE energy demand simulation software. Through simulations, architect experts designed 167 configuration of valid family houses of the same size. Then had to equip those with architectural (wall window ratio and orientation) and engineering (thermal insulation, heating system) components Then using weather data from local statistics of many years annual energy demand simulation must be done. Simulation outputs are generated by summarization of daily demands.
Using 70% of these samples (3507) 5 different type of models were created:

- linear regression
- decision tree with L1 metric
- decision tree with L2 metric
- decision tree with Poisson metric
- dense neural net with 1 hidden layer

3 versions were generated from all 5 model types, using different inputs:

- default inputs
- inputs extended with max. 2nd power of multiplicative combination
- inputs extended with max. 3rd power of multiplicative combination

That result 15 different model classes. To get rid of performance differences caused by randomization, 25 instances were generated from all 15 model classes. All 375 model instances were evaluated, then individual evaluations were averaged on classes.

MODEL EVALUATION

For better evaluation of model instance performances, a calculation of a single value is required for all instances. This single value must represent the approximation errors of each individual tests. R^2 metric is widely used in statistics and regression analyses was applied. As stated in equation 1, it conforms aforementioned requirement to represent individual approximation errors.

$$R^2 = 1 - \frac{\sum(y - \tilde{y})^2}{\sum(y - \bar{y})^2} \qquad \textbf{Equation 1}$$

where y is the value to approximate, $\bar{y}$ is its average and $\tilde{y}$ is the approximation.

In addition to R^2 score absolute and relative

(relative to the value to approximate) error average and their standard deviation also were measured.

RESULTS

After analysing averages and accuracy metrics of data groups, it can be stated that linear regression with 3rd and 2nd power of input variables and dense neural network with 1st (default) input had almost same performance, around 0.96 R^2 score. Score of all other model classes were below 0.75, therefore they were rejected.
Averages of absolute errors of estimations for all models are negligible compared to the annual demands, but only for the previously selected 3 model classes did the standard deviance of estimation error stayed below 500kWh/year.

CONCLUSION

As a final conclusion, it is stated that linear regression with higher power of input and dense neural net with default input perform the same.
But for model selection, structural flexibility is also taken into consideration. And linear regression model cannot be enhanced in the future. In the other hand, neural network structure can be extended or with more processors of the same type or different type of processor layers (convolutional, recurrent).
Therefore, the proposed model is the dense neural network with 1 hidden layer.

REFERENCES

[1] Peña-Guzmán C., Rey J. Forecasting residential electric power consumption for Bogotá Colombia using regression models, Energy Reports, (2020) vol. 6 sup. 1., pp. 561-566

[2] Mehedintu, A., Sterpu, M., Soava, G Estimation and Forecasts for the Share of Renewable Energy Consumption in Final Energy Consumption by 2020 in the European Union. Sustainability 2018, 10(5), 1515

[3] Mohammed, A., Alshibani, A., Alshamrani, O., Hassanain, M., A regression-based model for estimating the energy consumption of school facilities in Saudi Arabia. Energy & Buildings. 2021 vol. 237, 110809

[4] Sarstedt, M., Mooi, E. Regression Analysis. A Concise Guide to Market Research. pp 193-233. Springer Texts in Business and Economics 2014 ISBN: 978-3-642-53964-0

[5] Heiberger R., M., Holland B. Linear Regression by Least Squares in book Statistical Analysis and Data Display 2015 pp. 235-262, Springer

[6] Loh, W.Y. Classification and regression trees 2011. Wiley interdisciplinary reviews: Data Mining and KnowledgeDiscovery, 1(1), 14-23.

[7] Malte, J. Artificial neural network regression models: Predicting GDP growth. 2018. HWWI Research Paper, No. 185, Hamburgisches WeltWirtschaftsInstitut (HWWI), Hamburg

Clique relaxations of zero-one linear programs

Sándor Szabó Institute of Mathematics and Informatics University of Pécs
sszabo7@hotmail.com

Bogdán Zaválnij Alfréd Rényi Institute of Mathematics
bogdan@renyi.hu

ABSTRACT

In an earlier work a so-called conflict graph was associated to a given zero-one linear program basically to accumulate information to construct cuts to speed up the solution of the program. Later it was noticed that the conflict graph can be used in fixing values of variables and fathoming partial solutions in enumerative type solvers. In this paper we will show that the conflict graph helps in dividing dividing a zero-one linear program into independent smaller instances and so it opens a way for a parallel solution. Further the conflict graph suggests certain possibilities for preprocessing and simplifying the given zero-one linear program.

Keywords

discrete optimization, clique, independent set, weighted clique, zero-one program, parallel computing, preprocessing

1. INTRODUCTION

Given a zero-one linear program P. We assume that objective function of P is to be maximized, that is, we are dealing with a maximization problem. A zero-one variable sometimes called a binary or Boolean variable. The fact that a zero-one variables takes on the value zero sometimes expressed saying that variable is on level zero. Similarly, we can say that the variable is on level one.

Following [1] using the linear program P we construct a so-called conflict graph H and we assign this graph H to the program P. Here is the construction of H. Let $x_1, \ldots, x_n$ be the variables of the given zero-one linear program P and let $y_1 = 1 - x_1, \ldots, y_n = 1 - x_n$. Finally, for the sake of a uniform notation let

$$u_1 = x_1, \ldots, u_n = x_n, u_{n+1} = y_1, \ldots, u_{2n} = y_n.$$

The nodes of the conflict graph H are the variables

$$u_1, \ldots, u_n, u_{n+1}, \ldots, u_{2n}.$$

The nodes u_i, u_j, $1 \leq i < j \leq 2n$ are connected by an edge in H if the inequality $u_i + u_j \leq 1$ holds.

We distinguish two types of conflict graphs such as totally computed conflict graphs and partially computed conflict graphs. In other words a partially computed conflict graph can be viewed as a relaxed version of the totally computed conflict graph. Typically one works with partially computed conflict graphs. The reason of this is the following. Deciding whether the pair $\{u_i, u_j\}$ is an edge of the conflict graph H amounts to deciding the linear program P has a feasible solution with the extra constraints $u_i = u_j = 1$.

If the zero-one linear program P with the extra constraints $u_i = u_j = 1$ does not have any feasible solution then, the unordered pair $\{u_i, u_j\}$ is an edge of the conflict graph H. Carrying out these computations for each $1 \leq i < j \leq 2n$ can be computationally prohibitive. So one accepts this limitation and introduces edges into H whose existence can be verified easily. In this way we end up with a partially constructed conflict graph.

In [1] a number of properties of the conflict graph were established and were used to construct cuts. The next three of the above results were used to aid an enumerative solutions in [9].

LEMMA 1. *(Extension rule) If $\{x_i, u_j\}$ and $\{y_i, u_k\}$ are edges of H, then $\{u_j, u_k\}$ is an edge of H.*

LEMMA 2. *(Fixing rule) If $\{u_i, u_i\}$ is an edge (loop) of H, then $u_i = 0$ must hold.*

LEMMA 3. *(Fathoming rule) A partial solution in which x_i is fixed on level 0 and y_i is fixed on level 0 cannot be a feasible solution.*

We apply the extension rule repeatedly as long as the extension rule is applicable. This leads to the so-called closure of the conflict graph. If we are lucky we may fix the value of some variable in the linear program or we may fathom a partial solution.

The set of neighbors of a node v of the graph G consists of all the nodes of G that are adjacent to v. The set of neighbors of v is denoted by $N(v)$. In practice we apply the extension rule to the nodes x_i and y_i for each i, $1 \leq i \leq n$. Namely, the edges $\{x, y\}$, $x \in [N(x_i) \setminus \{y_i\}]$, $y \in [N(y_i) \setminus \{x_i\}]$ are added to the conflict graph. Of course some of these edges

Table 1: The adjacency matrix of a conflict graph.

		x	x	x	x	x	y	y	y	y	y
		1	2	3	4	5	1	2	3	4	5
x	1		•	•	•		•				•
x	2	•						•			
x	3	•							•		
x	4	•								•	
x	5										•
y	1	•						•	•		
y	2		•				•				
y	3			•			•				
y	4				•						
y	5	•				•					

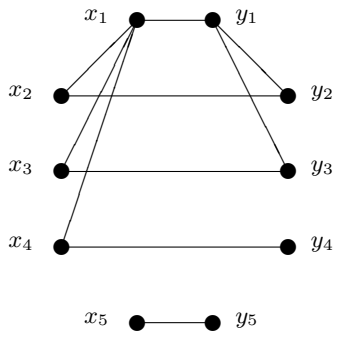

Figure 1: A graphical representation of the conflict graph

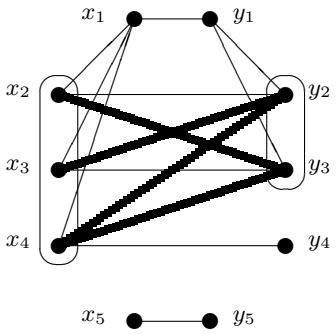

Figure 2: Extending the conflict graph. The newly added edges are bold.

may already be an edge of the conflict graph in which case we do not add any new edge to the conflict graph. Note that $y_i \in N(x_i)$, $x_i \in N(y_i)$ and so if $|N(x_i)| = 1$ or $|N(y_i)| = 1$, then we do not add any new edge to the conflict graph. If $[N(x_i) \setminus \{y_i\}] \cap [N(y_i) \setminus \{x_i\}] \neq \emptyset$, then there are variables whose values can be fixed. Namely, the variables appearing in the intersection can be fixed.

To see why let us assume that u_j is an element of the intersection. In this situation by the extension rule the unordered pairs $\{x_i, u_j\}$ and $\{y_i, u_j\}$ are edges of the conflict graph. Again, by the fixing rule $\{u_j, u_j\}$ is an edge of the conflict graph.

As an illustration we included a small example. In Table 1 the reader can see the adjacency matrix of a conflict graph. Figure 1 depicts a possible geometric representation of this conflict graph. Figure 2 shows the new edges we get applying the extension rule.

Of course the computations are happening on the adjacency matrices. During a computations only Tables 1 and 2 appear.

Table 2: The adjacency matrix of the extended conflict graph. The new edges are indicated by "○" signs.

		x	x	x	x	x	y	y	y	y	y
		1	2	3	4	5	1	2	3	4	5
x	1		•	•	•		•				•
x	2	•						•	○		
x	3	•						○	•		
x	4	•						○	○	•	
x	5										•
y	1	•						•	•		
y	2		•	○	○		•				
y	3		○	•	○		•				
y	4				•						
y	5	•				•					

2. THE WEIGHTED AGREEMENT GRAPH

In this section we define a new agreement graph G. This G can be used to preprocessing or precondition the original linear program and to divide it into smaller independent instances.

Let G be the complement of the subgraph of H induced by the set of nodes $\{x_1, \ldots, x_n\}$. To the node x_i of G we assign the coefficient of x_i in the objective function as a weight.

A subset C of the vertices $\{x_1, \ldots, x_n\}$ of the weighted agreement graph G is called a clique if each two distinct vertices in C are adjacent in G. The sum of the weights assigned to the elements of C is called the weight of the clique. The vector $[\alpha_1, \ldots, \alpha_n]$ is referred to as the characteristic vector of the set C if $\alpha_i = 1$ whenever $x_i \in C$ and $\alpha_i = 0$ whenever $x_i \notin C$.

The observation we will use is stated formally as a lemma.

LEMMA 4. *The set of characteristic vectors of the cliques of G contains each feasible solution of the zero-one linear program P.*

We spell out explicitly Lemma 5 as a corollary to Lemma 4.

LEMMA 5. *The value of each optimal solution of the zero-one linear program P is at most the weight of a maximum weight clique in the graph G.*

Any of the maximum weight clique solvers in [2], [4], [5], [6], [7] can be deployed to locate a maximum weight clique in the agreement graph G. This clique does not necessarily provide a feasible solution of the original zero-one linear program P. What is certain that we can establish upper bound for the optimal solution of the zero-one linear program P.

The Carraghan-Pardalos algorithm [3] is capable of listing all maximum weight cliques in the weighted agreement graph G. The algorithm maintains a partially constructed clique C and a list consisting of nodes that can be added to C to get a larger clique. With a slight modification of the procedure we may also check if C can be part of a feasible solution of the given zero-one program P. Thus, the modified Carraghan-Pardalos algorithm could locate a clique in G, which has maximum weight among the feasible solutions of P. In short, a clique problem solver can solve moderate size zero-one linear programs.

3. KERNELIZATION

A node of the graph G is referred to as a full degree node if it is adjacent to each other node of G. Let v be a full degree node of G and let G' be the subgraph of G induced by the nodes distinct from v. In plain English we get G' from G by deleting the node v.

LEMMA 6. *(Full degree rule) If C' is a maximum weight clique in G', then $C = C' \cup \{v\}$ is a maximum weight clique in G.*

Lemma 6 suggests to remove full degree nodes from the weighted conflict graph, then after locating a maximum weight clique in the reduced graph we can construct a maximum weight clique in the original weighted agreement graph.

We say that node v dominates node u of the weighted agreement graph G if u, v are distinct, $N(u) \subseteq N(v)$ and the weight of u is not larger than the weight of v. (Remember that $N(v)$ is the set of neighbors of the node v in G.)

We say that the edge $e = \{u, v\}$ dominates edge $f = \{v, w\}$ in the weighted agreement graph G if the unordered pair $\{u, w\}$ is not an edge of G, $[N(v) \cap N(w)] \subseteq [N(u) \cap N(v)]$ and the weight of w is not larger than the weight of u.

More generally, we say that the edge $e = \{u, v\}$ dominates edge $f = \{x, y\}$ in the weighted agreement graph G if at least one of the unordered pairs

$$\{u, x\}, \{u, y\}, \{v, x\}, \{v, y\}$$

is not an edge of G, $[N(x) \cap N(y)] \subseteq [N(u) \cap N(v)]$ and the sum of weights of x and y is not larger than the sum of weights of u and v.

The following result is proved in [8].

LEMMA 7. *If node v dominates node u, then node u can safely deleted from G when we are looking for a maximum weight clique in G.*

LEMMA 8. *If edge e dominates edge f, then edge f can safely deleted from G when we are looking for a maximum weight clique in G. (We do not delete any of the endpoints of the edge f.)*

Deleting a node from G means that we may fix the value of the corresponding variable of P on level zero. Deleting an edge from the weighted agreement graph G means that we may enter a new edge into the original unweighted conflict graph H. Applying the extension rule in H may result fixing variables or fathoming.

3.1 Coloring the vertices

In this section we will show how coloring of the nodes of the weighted agreement graph G can be used for preprocessing the zero-one linear program P.

We say that a coloring of the vertices of the graph G is a proper coloring if each node is colored exactly one color and the two end points of each edge of G receive distinct colors. Usually we use the numbers $1, \ldots, k$ as colors. The nodes of G receiving colors i give the elements of the color i-th color class C_i. A coloring of the nodes of G can be given by the color classes $C_1, \ldots, C_k$. From the color class C_i we pick a node with a maximum weight. Summing up these weights for each i, $1 \leq i \leq k$ we get a number which we call the weight of G with respect to the given coloring of the nodes of G.

Let v be a vertex of G. We consider the subgraph L of G induced by the set $N(v)$. Using a greedy algorithm we

properly color the vertices of L. To the vertex v we assign the weight of the coloring of L and call this number the color index of v.

Let $e = \{u, v\}$ be an edge of G. We consider the subgraph M of G induced by the set $N(u) \cap N(v)$. Using a greedy algorithm we properly color the vertices of M. To the edge e we assign the weight of the coloring of M and call this number the color index of e.

Using a greedy procedure we locate a clique C in G. The larger of the weight T of C is the better. Clearly, T is a lower bound of the weight of any maximum weight clique in G.

The basic observation we use for preprocessing is stated as a lemma.

LEMMA 9. *Let v be a vertex of G. If* $\text{weight}(v)+\text{index}(v) < T$*, then v can be deleted safely from G when we are looking for a maximum weight clique in G.*

Let $e = \{u, v\}$ be an edge of G. If $\text{weight}(u) + \text{weight}(v) + \text{index}(e) < T$*, then e can be deleted safely from G when we are looking for a maximum weight clique in G. (We do not delete any of the endpoints of the edge e.)*

4. PARALLELIZATION

Let us turn to the parallelization result. Let W_1, W_2, W_3 be subsets of the nodes of the finite simple graph G and assume that V is the set of nodes of G. Suppose W_1, W_2, W_3 are pair-wise disjoint and $V = W_1 \cup W_2 \cup W_3$. If in addition there is no edge of G is going from W_1 to W_3, then we say that the triple (W_1, W_2, W_3) is a splitting partition of G.

Let G_1 be the subgraph of G induced by the set of nodes $W_1 \cup W_2$ and let G_3 be the subgraph of G induced by the set of nodes $W_2 \cup W_3$.

The next result is proved in [8].

LEMMA 10. *If C is a maximum weight clique in the graph G, then C is a maximum weight clique in either G_1 or in G_3.*

In the particular case when both of the sets W_1, W_3 has more than one elements Lemma 10 offers a way to replace the original zero-one linear program P by two smaller programs P_1 and P_3. The program P_1 is constructed from program P by deleting variables corresponding the elements of the set W_3. Similarly, the program P_3 is constructed from program P by deleting variables corresponding the elements of the set W_1.

Sytematic ways to construct splitting partitions are presented and tested in [10] and [11].

5. ACKNOWLEDGMENTS

The project has been supported by National Research, Development and Innovation Office – NKFIH Fund No. SNN-135643.

6. REFERENCES

[1] A. Atamtürk and M. W. P. Savelsbergh, Conflict graphs in solving integer programming problems, *European Journal of Operation Research* **121** (1994), 40–45.

[2] E. Balas, J. Xue, Weighted and unweighted maximum clique algorithms with upper bounds from fractional coloring, *Algorithmica* **15** (1996), 397–412.

[3] R. Carraghan, P. M. Pardalos, An exact algorithm for the maximum clique problem, *Operation Research Letters* **9** (1990), 375–382.

[4] S. W. Cai and J. K. Lin, Fast solving maximum weight clique problem in massive graphs. In *Proceedings of 25th International Joint Conference on Artificial Intelligence*, IJCAI, (2016) 568???574.

[5] D. Kumlander, A new exact algorithm for the maximum weight clique problem based on a heuristic vertex-coloring and a backtrack search. In *Proceedings of Modeling, Computation and Optimization in Information Systems and Management Sciences, MCO, 202????208.* Hermes Science Publishing 2004.

[6] P. R. J. Östergård, A new algorithm for the maximum-weight clique problem, *Nordic Journal of Computing* **8** (2001), 424–436.

[7] P. Prosser, Exact algorithms for maximum clique: A computational study, *Algorithms* **5** (2012), 545–587.

[8] S. Szabó, Parallel algorithms for finding cliques in a graph, *Journal of Physics: Conference Series* **268** (2011)

[9] S. Szabó, Conflict graphs in implicit enumeration, *Pollack Periodica* **7** (2012), 145–156.

[10] S. Szabó, Metric space method for constructing spiltting partitions of graphs, *Acta Univ. Sapientiae, Informatica* **11** (2019), 131–141.

[11] S. Szabó and B. Zavalnij, Splitting partitions and clique search algorithms, *Middle-European Conference on Applied Theoretical Computer Science 2019*, 75–78.

Indeks avtorjev / Author index

Srednjeevropska konferenca o uporabnem teoretičnem računalništvu

Middle-European Conference on Applied Theoretical Computer Science

Uredniki • Editors:
Andrej Brodnik, Gábor Galambos, Branko Kavšek