6.–10. oktober 2025 | 6–10 October 2025
Koper, Slovenia

INFORMACIJSKA DRUŽBA

INFORMATION SOCIETY

Srednjeevropska konferenca o uporabnem teoretičnem računalništvu in informatiki (MATCOS)

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

Uredniki | Editors:
Andrej Brodnik, Gábor Galambos, Rok Požar

Zbornik 28. mednarodne multikonference

Zvezek N

Proceedings of the 28th International Multiconference

Volume N

Zbornik 28. mednarodne multikonference

INFORMACIJSKA DRUŽBA – IS 2025

Zvezek N

Proceedings of the 28th International Multiconference

INFORMATION SOCIETY – IS 2025

Volume N

Srednjeevropska konferenca o uporabnem teoretičnem računalništvu in informatiki (MATCOS)
Middle-European Conference on Applied Theoretical Computer Science (MATCOS)

Uredniki / Editors

Andrej Brodnik, Gábor Galambos, Rok Požar

http://is.ijs.si

9.–10. oktober 2025 / 9–10 October 2025
Koper, Slovenia

Uredniki:

Andrej Brodnik
Univerza na Primorskem, Koper in Univerza v Ljubljani, Ljubljana

Gábor Galambos
Univerza v Szegedu, Szeged

Rok Požar
Univerza na Primorskem, Koper

Založnik: Institut »Jožef Stefan«, Ljubljana
Priprava zbornika: Mitja Lasič, Vesna Lasič, Lana Zemljak
Oblikovanje naslovnice: Vesna Lasič, uporabljena slika iz Pixabay

Dostop do e-publikacije:
http://library.ijs.si/Stacks/Proceedings/InformationSociety

Ljubljana, oktober 2025

Informacijska družba
ISSN 2630-371X
DOI: https://doi.org/10.70314/is.2025.matcos

Kataložni zapis o publikaciji (CIP) pripravili v Narodni in univerzitetni knjižnici v Ljubljani
COBISS.SI-ID 255599107
ISBN 978-961-264-332-4 (PDF)

PREDGOVOR MULTIKONFERENCI INFORMACIJSKA DRUŽBA 2025

28. mednarodna multikonferenca *Informacijska družba* se odvija v času izjemne rasti umetne inteligence, njenih aplikacij in vplivov na človeštvo. Vsako leto vstopamo v novo dobo, v kateri generativna umetna inteligenca ter drugi inovativni pristopi oblikujejo poti k superinteligenci in singularnosti, ki bosta krojili prihodnost človeške civilizacije. Naša konferenca je tako hkrati tradicionalna znanstvena in akademsko odprta, pa tudi inkubator novih, pogumnih idej in pogledov.

Letošnja konferenca poleg umetne inteligence vključuje tudi razprave o perečih temah današnjega časa: ohranjanje okolja, demografski izzivi, zdravstvo in preobrazba družbenih struktur. Razvoj UI ponuja rešitve za številne sodobne izzive, kar poudarja pomen sodelovanja med raziskovalci, strokovnjaki in odločevalci pri oblikovanju trajnostnih strategij. Zavedamo se, da živimo v obdobju velikih sprememb, kjer je ključno, da z inovativnimi pristopi in poglobljenim znanjem ustvarimo informacijsko družbo, ki bo varna, vključujoča in trajnostna.

V okviru multikonference smo letos združili dvanajst vsebinsko raznolikih srečanj, ki odražajo širino in globino informacijskih ved: od umetne inteligence v zdravstvu, demografskih in družinskih analiz, digitalne preobrazbe zdravstvene nege ter digitalne vključenosti v informacijski družbi, do raziskav na področju kognitivne znanosti, zdrave dolgoživosti ter vzgoje in izobraževanja v informacijski družbi. Pridružujejo se konference o legendah računalništva in informatike, prenosu tehnologij, mitih in resnicah o varovanju okolja, odkrivanju znanja in podatkovnih skladiščih ter seveda Slovenska konferenca o umetni inteligenci.

Poleg referatov bodo okrogle mize in delavnice omogočile poglobljeno izmenjavo mnenj, ki bo pomembno prispevala k oblikovanju prihodnje informacijske družbe. »Legende računalništva in informatike« predstavljajo domači »Hall of Fame« za izjemne posameznike s tega področja. Še naprej bomo spodbujali raziskovanje in razvoj, odličnost in sodelovanje; razširjeni referati bodo objavljeni v reviji *Informatica*, s podporo dolgoletne tradicije in v sodelovanju z akademskimi institucijami ter strokovnimi združenji, kot so ACM Slovenija, SLAIS, Slovensko društvo Informatika in Inženirska akademija Slovenije.

Vsako leto izberemo najbolj izstopajoče dosežke. Letos je nagrado *Michie-Turing* za izjemen življenjski prispevek k razvoju in promociji informacijske družbe prejel **Niko Schlamberger**, priznanje za raziskovalni dosežek leta pa **Tome Eftimov**. »Informacijsko limono« za najmanj primerno informacijsko tematiko je prejela odsotnost obveznega pouka računalništva v osnovnih šolah. »Informacijsko jagodo« za najboljši sistem ali storitev v letih 2024/2025 pa so prejeli Marko Robnik Šikonja, Domen Vreš in Simon Krek s skupino za slovenski veliki jezikovni model GAMS. Iskrene čestitke vsem nagrajencem!

Naša vizija ostaja jasna: prepoznati, izkoristiti in oblikovati priložnosti, ki jih prinaša digitalna preobrazba, ter ustvariti informacijsko družbo, ki koristi vsem njenim članom. Vsem sodelujočim se zahvaljujemo za njihov prispevek — veseli nas, da bomo skupaj oblikovali prihodnje dosežke, ki jih bo soustvarjala ta konferenca.

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

FOREWORD TO THE MULTICONFERENCE INFORMATION SOCIETY 2025

The 28th International Multiconference on the Information Society takes place at a time of remarkable growth in artificial intelligence, its applications, and its impact on humanity. Each year we enter a new era in which generative AI and other innovative approaches shape the path toward superintelligence and singularity — phenomena that will shape the future of human civilization. The conference is both a traditional scientific forum and an academically open incubator for new, bold ideas and perspectives.

In addition to artificial intelligence, this year's conference addresses other pressing issues of our time: environmental preservation, demographic challenges, healthcare, and the transformation of social structures. The rapid development of AI offers potential solutions to many of today's challenges and highlights the importance of collaboration among researchers, experts, and policymakers in designing sustainable strategies. We are acutely aware that we live in an era of profound change, where innovative approaches and deep knowledge are essential to creating an information society that is safe, inclusive, and sustainable.

This year's multiconference brings together twelve thematically diverse meetings reflecting the breadth and depth of the information sciences: from artificial intelligence in healthcare, demographic and family studies, and the digital transformation of nursing and digital inclusion, to research in cognitive science, healthy longevity, and education in the information society. Additional conferences include Legends of Computing and Informatics, Technology Transfer, Myths and Truths of Environmental Protection, Knowledge Discovery and Data Warehouses, and, of course, the Slovenian Conference on Artificial Intelligence.

Alongside scientific papers, round tables and workshops will provide opportunities for in-depth exchanges of views, making an important contribution to shaping the future information society. *Legends of Computing and Informatics* serves as a national »Hall of Fame« honoring outstanding individuals in the field. We will continue to promote research and development, excellence, and collaboration. Extended papers will be published in the journal *Informatica*, supported by a long-standing tradition and in cooperation with academic institutions and professional associations such as ACM Slovenia, SLAIS, the Slovenian Society Informatika, and the Slovenian Academy of Engineering.

Each year we recognize the most distinguished achievements. In 2025, the Michie-Turing Award for lifetime contribution to the development and promotion of the information society was awarded to **Niko Schlamberger**, while the Award for Research Achievement of the Year went to **Tome Eftimov**. The »Information Lemon« for the least appropriate information-related topic was awarded to the absence of compulsory computer science education in primary schools. The »Information Strawberry« for the best system or service in 2024/2025 was awarded to Marko Robnik Šikonja, Domen Vreš and Simon Krek together with their team, for developing the Slovenian large language model GAMS. We extend our warmest congratulations to all awardees.

Our vision remains clear: to identify, seize, and shape the opportunities offered by digital transformation, and to create an information society that benefits all its members. We sincerely thank all participants for their contributions and look forward to jointly shaping the future achievements that this conference will help bring about.

Mojca Ciglarič, Chair of the Program Committee
Matjaž Gams, Chair of the Organizing Committee

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
Gašper Slapničar
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 28. mednarodne multikonference

INFORMACIJSKA DRUŽBA – IS 2025

Zvezek N

Proceedings of the 28th International Multiconference

INFORMATION SOCIETY – IS 2025

Volume N

Srednjeevropska konferenca o uporabnem teoretičnem računalništvu in informatiki (MATCOS)
Middle-European Conference on Applied Theoretical Computer Science (MATCOS)

Uredniki / Editors

Andrej Brodnik, Gábor Galambos, Rok Požar

http://is.ijs.si

9.–10. oktober 2025 / 9–10 October 2025
Koper, Slovenia

PREDGOVOR

To je že peti MATCOS – konferenca, ki povezuje dva na videz nepovezljiva pojma, aplikativnost in teorijo, v konferenco o uporabnem teoretičnem računalništvu. Ko smo leta 2013 organizirali prvo konferenco MATCOS, smo lahko le sanjali o ustvarjanju tradicije. Naše prejšnje konference so se osredotočale na gradnjo: poskušali smo ustvariti novo konferenčno prizorišče v Srednji Evropi in zelo trdo smo delali, da bi to dosegli. Številna mesta poskušajo z organizacijo konferenc na različnih področjih računalništva in/ali matematike, vendar ne uspe vsem uresničiti njihovih načrtov. Organizacija konference je težko delo, a zahvaljujoč neutrudnim prizadevanjem članov OO in PO ter vseh, ki so uporabili svoje povezave, smo tukaj.

Po zaključenem recenzentskem postopku se je PO odločil sprejeti 20 rednih prispevkov, ki so tudi vključeni v ta zbornik. V skladu z našimi prejšnjimi prizadevanji, želimo tudi letos dati priložnost mladim raziskovalcem. Zato smo posebno pozornost namenili prispevkom, ki so jih predložili doktorski študenti, in v nekaterih primerih nekaj od njih vključili med redne predstavitve. 14 kratkih predstavitev, ki so bile sprejete, ponuja priložnost za nadaljnjo razširitev spektra raziskav predstavljenih na MATCOS.

V preteklih letih smo na plenarna predavanja na konferencah MATCOS povabili predavatelje, ki so vodilni strokovnjaki na svojih področjih. Letošnja konferenca ni izjema: Michel Bierlaire, profesor na École Polytechnique Fédérale de Lausanne (EPFL), se je odzval našemu povabilu. Njegovo raziskovalno področje – raziskave prometa – se dobro ujema z okvirom MATCOS, njegova prisotnost pa ponuja dobro priložnost za vzpostavitev in poglobitev osebnih stikov.

Vse udeležence MATCOS-25 lepo pozdravljamo in upamo, da boste uživali v teh dveh dneh v Kopru. Spoznajte mesto, saj je Koper eden od diamantov Istre. Prepričani smo, da bodo prihajajoči dnevi prispevali k nadaljnjemu razvoju obstoječih odnosov in upamo, da se bodo s pomočjo konference MATCOS oblikovale tudi nove strokovne skupine.

V imenu organizatorjev

Andrej Brodnik in Gábor Galambos

Sopredsednika

FOREWORD

This is already the fifth MATCOS – the conference that connects two seemingly unconnectable terms application and theory into a conference on applied theoretical computer science. When we started to organize the first MATCOS conference back in 2013, we could only dream of creating a tradition. Our previous conferences focused on building: we tried to create a new conference venue in Central Europe, and we worked very hard to achieve this. Many cities are experimenting with organizing conferences in various fields of computer science and/or mathematics, but not everyone can manage to realize their plans. Organizing a conference is a hard work, but thanks to the tireless efforts of OC and PC members, and everyone who used their connections, here we are.

As a result of the review process, the PC accepted 20 regular papers, and these are included in the Proceedings. In keeping with our previous efforts, we want to give young researchers a chance this year as well, so we gave special consideration to the papers submitted by PhD students and, in some cases, included a few of them among the regular talks. The 14 short talks that were accepted provide an opportunity to further broaden the spectrum of research presented at MATCOS.

In previous years, we have invited speakers who are leading experts in their fields to give plenary presentations at MATCOS conferences. This year's conference is no exception: Michel Bierlaire, professor at the École Polytechnique Fédérale de Lausanne (EPFL), has accepted our invitation. His research field — transport research — fits well within the scope of MATCOS, and his presence provides a good opportunity to establish and deepen personal relationships.

A warm welcome to all MATCOS-25 participants, and we hope you enjoy these two days in Koper. Get to know the city, because Koper is one of the diamond of Istria. We are confident that the coming days will contribute to the further development of existing relationships, and we hope that new professional groups will also be formed with the help of the MATCOS conference.

On behalf of the organizers

Andrej Brodnik and Gábor Galambos

Co-chairs

PROGRAMSKI ODBOR / PROGRAMME COMMITTEE

Andrej Brodnik (Koper, Ljubljana, Slovenia) co-chair

Bo Chen (Warwick, UK)

Gábor Galambos (Szeged, Hungary) co-chair

Kathrin Hanauer (Vienna, Austria)

Gabriel Istrate (Bucharest, Romania)

Miklós Krész (Szeged, Hungary and Koper, Slovenia)

Silvano Martello (Bologna, Italy)

Andrzej Mizera (Warsaw, Poland)

Benedek Nagy (Famagusta, Cyprus, Turkey)

Bengt J. Nilsson (Malmö, Sweeden)

Ulrich Pferschy (Graz, Austria)

Gerhard Reinelt (Heidelberg, Germany)

Aleksi Saarela (Turku, Finland(

Attila Sali (Budapest, Hungary)

Yllka Velaj (Vienna, Austria)

Borut Žalik (Maribor, Slovenia)

Merging Operations in the Open-Shop Scheduling Problem

Daniil Baldouski
daniil.baldouski@iam.upr.si
University of Primorska, IAM
Koper, Slovenia

Balázs Dávid
balazs.david@innorenew.eu
InnoRenew CoE
Izola, Slovenia
University of Primorska, IAM and FAMNIT
Koper, Slovenia

Miklós Krész
miklos.kresz@innorenew.eu
InnoRenew CoE
Izola, Slovenia
University of Primorska, IAM and FAMNIT
Koper, Slovenia
University of Szeged
Szeged, Hungary

ABSTRACT

In this work we consider the open-shop scheduling problem with operation batching (OSSP-OB), which extends classical open-shop scheduling by allowing operations of the same category to be processed simultaneously on a machine. To address this problem, we develop an exact procedure for constructing optimal batches with the corresponding machine scheduling by developing a mixed-integer linear programming (MILP) model. The effectiveness of the method is evaluated on a dedicated benchmark instance set.

KEYWORDS

job scheduling, batch scheduling, open-shop scheduling, mixed-integer linear programming

1 INTRODUCTION

We consider a scheduling problem that combines ideas from open-shop and batch scheduling. Each job consists of several operations, each associated with a category and executable on one of the suitable machines for a specified processing time. Operations within a job can be processed in arbitrary order, so the problem environment belongs to the class of *open-shop scheduling* (OSSP) [3, 4, 1].

In addition, we allow multiple operations of the same category to be executed simultaneously on a machine. This grouping procedure, which we call *merging*, is related to the notion of *batching* in batch scheduling [5, 2, 7], but differs as it applies to operations rather than jobs.

The resulting formulation, referred to as the open-shop scheduling problem with operation batching (OSSP-OB), extends the classical OSSP by incorporating merging of operations and captures a wider range of scheduling scenarios. The OSSP can then be viewed as a special case of OSSP-OB in which a unique category is provided for every operation, thus leading OSSP-OB to be an NP-hard problem in the general case.

2 PROBLEM DESCRIPTION

The environment of the open-shop scheduling problem with the merging of operations (operations batching) is given by the sets of jobs, operations, and machines, with various parameters for each of the sets. Formally, OSSP-OB environment consists of:

- The time horizon of the problem is represented by the discretized set of consecutive time units T.
- A finite set J of **jobs**. For each job $j \in J$ there is a **time window** $[a_j, b_j]$ within the time horizon of the problem for which the operations of this job should be analyzed and approved.
- A finite set O of **operations** associated with jobs. For each job $j \in J$, a set $O_j \subseteq O$ of operations associated with the job.
- A finite set C of operation **categories** associated with operations. Each operation $o \in O$ has a category $C(o)$ and there are no two operations of the same category that belong to the same job. The time it takes to perform the operation is the same within the operation category and is then defined for each category $c \in C$ as $\ell(c)$ (the length of the category). The set of all operations of the same category is denoted by $O(c)$ for each category $c \in C$.
- A finite set M of **machines**. For each machine $m \in M$, a set $C_m \subseteq C$ of operation categories that can be carried out on the machine.
- Operations can be performed in batches if they belong to the same category. This will be referred to as **merging** of operations. Each merged operation $s \in S$ is a set of operations, that is, $s \subseteq O$ such that s contains operations of the same category. The set of all merged operations can be understood as a set $S \subseteq \mathcal{P}(O)$ that contains all the possible valid merged operations.

Additional parameters can include, but are not limited to:

- capacity parameters for the number of operations for each machine,
- capacity of the number of merged jobs for each category,
- time window for which the operations of the job should be analyzed,
- internal deadline of the job,
- periods of activity/inactivity,
- time to merge operations (e.g. constant, flexible, proportional to the execution time, proportional to the number of merged operations, and more)
- precedence relations between the operations within the same job,
- precedence relations between jobs.

In this work, we consider two additional parameters that are of high importance in the use-case of laboratory experiments: *internal deadlines* and *activity periods*.

The *internal deadline* (d_j) can be defined as k time units before b_j, which is the desired internal target for operations to be analyzed and approved, which then gives some time (k time units) for corrections in the case of exceptions (failures). A good

Information Society 2025, 6–10 October 2025, Ljubljana, Slovenia

example of a merging parameter would be a limit on the number of operations that can be merged ($m(C)$) for each category $c \in C$.

In the context of QA laboratory experiments, personnel scheduling plays an important role, since most tasks (such as machine preparation, machine loading, validation of the results, and more) are performed manually, each requiring a specific employee qualification to handle it. In this work, we consider personnel scheduling in a simplified format, assuming that all manual tasks are performed instantaneously and with infinite personnel capacity.

Let us call *periods of inactivity* all the gaps in the time horizon (e.g., weekends, national holidays, other special events) during which machines are unavailable and none of the tasks can be performed. Let us then consider an *activity period* to be any maximal interval of time units that does not contain periods of inactivity.

An example of the problem (Fig. 1) with a feasible solution (Fig. 2) is demonstrated. In this example, we are given 3 jobs ($J = \{J1, J2, J3\}$, sorted by the earliest allowed start time), each of them containing $3-4$ operations. The time windows of the jobs are given ($[A(j), B(j)]$ for each job $j \in J$), as well as the lengths of the operations (categories). The colors of the operations represent their categories and there are 6 categories in total $C = \{C1, \ldots, C6\}$. For example, $O(J2) = \{O1, O2, O3\}$ such that $C(O3) = C1$.

There are 3 machines ($M = \{M1, M2, M3\}$), each of them corresponds to a set of categories, of which operations are allowed to be processed. In this example, $C_{M1} = \{C1, \ldots, C4\}$, while $C_{M2} = C_{M3} = \{C4, \ldots, C6\}$. The example does not dive deep into all the possible parameters of the problem, assuming that merging is allowed in case operations belong to the same category, while satisfying the time constraints.

A feasible solution (Fig. 2) is given in the form of performed merges, (merged) operations-machines assignment, with the (merged) operations timeline. There are 4 potential merges for categories $C1, C2, C4$ and $C5$, representing different types of merges that can occur. For example, merging of category $C1$ is not allowed due to time limitation ($\ell(C1) > B(J1) - A(J2)$). Because of this, the schedule of machine $M1$ becomes tight, making the merging of operations of category $C2$ necessary. Merges of categories $C4$ and $C5$ are possible but not necessary and would be performed (or not) based on the specifics of the objective function.

In general, the goal of this work is to find the most efficient scheduling of operations to machines given the merging constraints, with the objectives to consider including tardiness, number of batches, flowtime, makespan, and overall costs.

3 SOLUTION METHODOLOGY

In this work, we formulate the OSSP-OB as a mixed-integer linear programming (MILP) model that integrates both the merging of operations and the scheduling of merged operations.

3.1 Problem data

For each $s \in S$, $j \in J$, $o \in O$, $m \in M$, $c \in C$, and $\tau \in T$, we establish the following parameters.

(1) **Assignment parameters:**
- $aOJ(o)$ - the index of the job associated with the operation o.
- $aOC(o)$ - the index of the category of the operation o.
- $aMC(m, c) = 1$ if and only if operations of category c can be carried out on the machine m.

(2) **Processing time parameters:**
- $tC(c)$ - time required to perform the operations of category c.
- $tJ^a(j)$ - the start of the time interval for which the operations of job j should be analyzed.
- $tJ^b(j)$ - the end of the time interval for which the operations of job j should be analyzed.
- $tJ^d(j)$ - the internal deadline of the time interval for which the operations of job j should be analyzed.
- A - set of activity periods, $A \subseteq T$.

(3) **Auxiliary parameters.** A set of derived parameters follows, simplifying the connection between the set of merged operations and:
- **Categories:**
  - $aSC(s, c) = 1$ if $c = aOC(o)$ for any $o \in s$, 0 otherwise, for a merged operation s.
  - Let $c^s \in C$ be the category of a merged operation s, that is $aSC(s, c^s) = 1$.
- **Jobs:**
  - $aSJ(s, j) = 1$ if $j \in \{aOJ(o) : o \in s\}$ and 0 otherwise, for a merged operation s. That is, if the merged operation s and the job j are associated with each other.
- **Machines:**
  - $aSM(s, m) = 1$ if there is at least one category $c \in C$ such that $aSC(s, c) = 1$ and $aMC(m, c) = 1$, 0 otherwise, for merged operation s and machine m.
  - Let set M^s be the set of machines compatible with the merged operation s, that is, for every $s \in S$:

$$M^s = \{m \,|\, m \in M, aSM(s, m) = 1\}.$$

Additionally, we introduce a big $\mathcal{M}$ parameter based on the time horizon T:

$$\mathcal{M} = 2 \cdot |T| + 1.$$

3.2 Variables

For each $s \in S$, $j \in J$, $o \in O$, $m \in M$, $c \in C$, and $\tau \in T$, we establish the following variables.

(1) **Assignment variables:**
- $x_{s,m,\tau} = 1$ if and only if merged operation s is active on machine m in time unit τ.
- $y_{s,\tau} = 1$ if and only if the merged operation s starts in the time unit τ.
- $z_{s,m} = 1$ if and only if the merged operation s and machine m are assigned to each other during the time horizon T.
- $r_s = 1$ if and only if the merged operation s is active during the time horizon T.

(2) **Auxiliary variables.** A set of helpful variables to simplify the model formulation, grouping the merged operations parameters together with the decision variable r_s that represents the activity status of a merged operation:
- $r^c_{s,c} = r_s \cdot aSC(s, c)$ for merged operation s and category c.
- $r^j_{s,j} = r_s \cdot aSJ(s, j)$ for merged operation s and job j.
- $r^m_{s,m} = r_s \cdot aSM(s, m)$ for merged operation s and machine m.
- $r^{cj}_{s,c,j} = r^c_{s,c} \cdot r^j_{s,j}$ for merged operation s, category c and job j.
- $k_{s,m,j,\tau} = r^j_{s,j} \cdot x_{s,m,\tau}$ for merged operation s, machine m, category c and time unit τ.

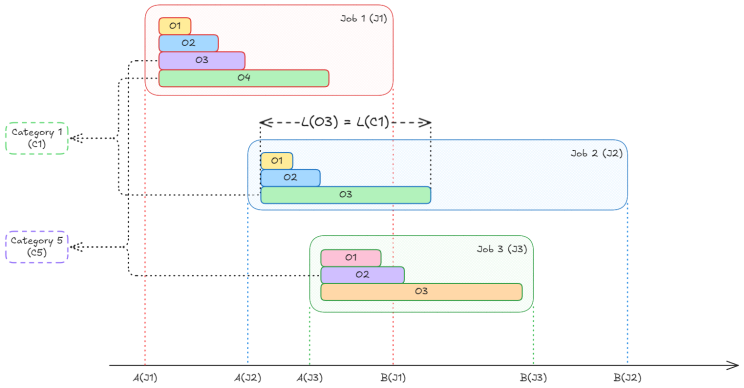


Figure 1: Example of OSSP-OB with 3 jobs and $3-4$ operations per job.

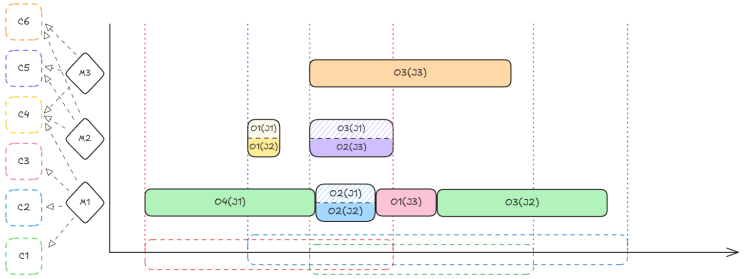


Figure 2: Example of an OSSP-OB solution, demonstrating operations to machines scheduling.

3.3 Constraints

(1) Auxiliary variables definitions. For each $s \in S$, $m \in M$, $c \in C$, $j \in J$, $\tau \in T$:
- $r^c_{s,c} = r_s \cdot aSC(s,c)$.
- $r^j_{s,j} = r_s \cdot aSJ(s,j)$.
- $r^m_{s,m} = r_s \cdot aSM(s,m)$.
- Linearization of r^{cj}:

$$r^{cj}_{s,c,j} \leq r^c_{s,c} \tag{1}$$
$$r^{cj}_{s,c,j} \leq r^j_{s,j} \tag{2}$$
$$r^c_{s,c} + r^j_{s,j} - 1 \leq r^{cj}_{s,c,j} \tag{3}$$

- Linearization of k:

$$k_{s,m,j,\tau} \leq r^j_{s,j} \tag{4}$$
$$k_{s,m,j,\tau} \leq x_{s,m,\tau} \tag{5}$$
$$r^j_{s,j} + x_{s,m,\tau} - 1 \leq k_{s,m,j,\tau} \tag{6}$$

(2) At any time unit, merged operations and machines have at most one unique schedule. For each time unit $\tau \in T$:
- For each merged operation $s \in S$:

$$\sum_{m \in M} x_{s,m,\tau} \leq 1 \tag{7}$$

- For each machine $m \in M$:

$$\sum_{s \in S} x_{s,m,\tau} \leq 1 \tag{8}$$

(3) For each merged operation $s \in S$ to be scheduled ($r_s = 1$):
- There is exactly one starting time unit:

$$\sum_{\tau \in T} y_{s,\tau} = r_s \tag{9}$$

- Exactly one machine is assigned:

$$\sum_{m \in M} z_{s,m} = r_s \tag{10}$$

(4) Merged operations can not be scheduled on machines that do not share the corresponding category. For each merged operation $s \in S$:
- For each machine $m \in M \setminus M^s$:

$$\sum_{\tau \in T} x_{s,m,\tau} = 0 \tag{11}$$
$$z_{s,m} = 0 \tag{12}$$

- For each machine $m \in M^s$:

$$\sum_{\tau \in T} x_{s,m,\tau} = r_s \cdot tC(c^s) \tag{13}$$

(5) Merged operations should start and end within the corresponding time interval. For each merged operation $s \in S$, job $j \in J$ and category $c \in C$:

$$\sum_{\tau \in T, \tau < tJ^a(j)} y_{s,\tau} \leq (1 - r^j_{s,j})\mathcal{M} \tag{14}$$
$$\sum_{\tau \in T, \tau > tJ^b(j) - tC(c)} y_{s,\tau} \leq (1 - r^{cj}_{s,c,j})\mathcal{M} \tag{15}$$

(6) Merged operations can be performed only on the dedicated machines. For each merged operation $s \in S$, machine $m \in M$ and category $c \in C$:

$$z_{s,m} \leq r^m_{s,m} \tag{16}$$

(7) Each merged operation $s \in S$ occupies $tC(c)$ consecutive time units for a unique $c \in C$ such that $r^c_{s,c} = 1$. For each

merged operation $s \in S$, machine $m \in M$, category $c \in C$, time unit $\tau \in T$ and $\tau' \in \{0, 1, \ldots, tC(c) - 1\}$:

$$x_{s,m,\tau+\tau'} \geq y_{s,\tau} - (1 - z_{s,m})\mathcal{M} - (1 - r^c_{s,c})\mathcal{M}, \tag{17}$$

where $\tau + \tau'$ denotes the time τ' units after time τ.

(8) Merged operations can not be active earlier than they start. For each merged operation $s \in S$, for each $\tau \in T$ and $\tau' \in T$ such that $\tau' < \tau$:

$$x_{s,m,\tau'} \leq y_{s,\tau} \tag{18}$$

(9) Machines are not assigned merged operations during the periods of inactivity. For each time unit $\tau \in T \backslash A$:

$$\sum_{s \in S, m \in M} x_{s,m,\tau} = 0 \tag{19}$$

(10) For each operation $o \in O$ only one merged operation $s \in S$ such that $o \in s$ can be active during the time horizon. For each $o \in O$:

$$\sum_{s \in S, o \in s} r_s = 1 \tag{20}$$

3.4 Objective function

Minimize the overall tardiness of merged operations with respect to internal deadline of the corresponding jobs:

$$\text{Minimize} \sum_{s \in S} \left(\sum_{m \in M} \sum_{j \in J} \sum_{\tau \in T, \tau \geq tJ^d(j)} k_{s,m,j,\tau} \right) \tag{21}$$

4 RESULTS

To evaluate our approach, we generated a new benchmark set for the OSSP-OB following the ideas of E.Taillard[6]. Instances are defined by the number of jobs, machines, and categories. We considered jobs of sizes 3, 5, 7, and 10, and for each case the number of machines and categories was set to either 2 or 3. All other parameters, such as processing times, time horizon, and number of operations per job, were kept constant. The construction procedure is flexible and can be extended to introduce randomness or additional parameters if needed.

Each valid merge corresponds to a subset of operations of the same category, meaning that the set of possible merges is a subset of the power set $\mathcal{P}(O)$. The size of this set is strongly influenced by the number of available category capacities: smaller values lead to larger collections of potential merges. This effect directly impacts solver memory usage and thus scalability.

Computational experiments were carried out using the Gurobi 11.0.0 solver on an AMD Ryzen 7 5800H 3.2 GHz CPU with 16 GB RAM, with a time limit of 60 minutes per instance. Table 1 demonstrates the maximum runtime observed for each instance configuration. All instances of this set were solved optimally.

Overall, the model is able to solve instances up to 10 jobs in a reasonable time, but the growth of the merged operation space remains the main computational bottleneck. For example, increasing the number of jobs to 15 leads to out-of-memory errors.

N jobs	N machines	N categories	Avg. Runtime (s)
3	2	2	1.11
3	2	3	0.56
3	3	2	1.50
3	3	3	0.67
5	2	2	6.90
5	2	3	2.55
5	3	2	6.36
5	3	3	3.09
7	2	2	16.81
7	2	3	7.54
7	3	2	20.33
7	3	3	10.70
10	2	2	124.92
10	2	3	40.62
10	3	2	156.67
10	3	3	47.94

Table 1: Average runtimes (in seconds) for OSSP-OB instances.

5 CONCLUSION

In this work, we introduced a complete MILP formulation for OSSP-OB. Our experiments show that the model can solve small instances to optimality within a reasonable time, but quickly becomes impractical as the number of jobs and possible merges increases. Rapid growth of the set of potential merged operations leads to excessive memory usage and limits scalability.

Future research will therefore focus on approximation methods. A potential approach to consider is separating the OSSP-OB environment into two subproblems, merging of operations and scheduling of the merged operations, combining exact methods for the two parts separately. Another approach is to design dedicated (meta-)heuristics specific to the merging structure. Such approaches are expected to extend the range of solvable instances and provide practical solutions for larger problem sizes.

ACKNOWLEDGEMENTS

The research was supported by the BioLOG project: Balázs Dávid and Miklós Krész are grateful for the support of National Center of Science (NCN) through grant DEC-2020/39/I/HS4/03533, the Slovenian Research and Innovation Agency (ARIS) through grant N1-0223 and the Austrian Science Fund (FWF) through grant I 5443-N. This work was partially supported by the Slovenian Research Agency, research program P1-0404 and by the research program CogniCom (0013103) at the University of Primorska. Balázs Dávid is grateful for the support of ARIS through grant J1-50000, and gratefully acknowledges the Slovenian Research and Innovation Agency (ARIS) and the Ministry of the Economy, Tourism and Sport (MGTŠ) for the grant V4-2512.

REFERENCES

[1] Danyu Bai, Zhi-Hai Zhang, and Qiang Zhang. 2016. Flexible open shop scheduling problem to minimize makespan. *Computers & operations research*, 67, 207–215.

[2] John W Fowler and Lars Mönch. 2022. A survey of scheduling with parallel batch (p-batch) processing. *European journal of operational research*, 298, 1, 1–24.

[3] Wieslaw Kubiak. 2022. *Book of Open Shop Scheduling*. Springer.

[4] B Naderi, SMT Fatemi Ghomi, Majid Aminnayeri, and Mostafa Zandieh. 2011. Scheduling open shops with parallel machines to minimize total completion time. *Journal of Computational and Applied Mathematics*, 235, 5, 1275–1287.

[5] Chris N Potts and Mikhail Y Kovalyov. 2000. Scheduling with batching: a review. *European journal of operational research*, 120, 2, 228–249.

[6] Eric Taillard. 1993. Benchmarks for basic scheduling problems. *European journal of operational research*, 64, 2, 278–285.

[7] Tanya Y Tang and J Christopher Beck. 2020. CP and hybrid models for two-stage batching and scheduling. In *Integration of Constraint Programming, Artificial Intelligence, and Operations Research: 17th International Conference, CPAIOR 2020, Vienna, Austria, September 21–24, 2020, Proceedings 17*. Springer, 431–446.

Flexibility vs. efficiency: a study in sawmill scheduling

Csaba Kebelei
Eötvös Loránd University
Faculty of Informatics
Budapest, Hungary
kebeleicsaba@student.elte.hu

Máté Hegyháti
University of Sopron
Sopron, Hungary
University of Pannonia
Veszprém, Hungary

Abstract

Sawmills are an important stage in the primary wood industry, producing timber and other side products from harvested logs. Increasing efficiency in a sawmill is not only an economic desire, but it also propagates to the competitiveness of using more renewable resources in dependent industries such as construction, packaging, furniture. Although the road from a tree log to a shipped product involves many stages, the key processing step is the sawing performed by high-value sawing machines. As the operation of these machines plays a key role in overall efficiency, several research papers have addressed their scheduling. Naturally, allowing more freedom in the production plan may result in better financial results in exchange for increased computational needs of the optimizer. However, more complex schedules may also pose an additional burden in their real-life execution. This paper presents several variants of a model formerly proposed by the authors and investigates this trade-off relationship empirically.

Keywords

sawmill, scheduling, MILP

1 Introduction and literature

The increasing utilization of renewable resources is a major directive of many nations and companies around the world. Wood is a natural resource that not only serves as a great carbon sequestration tool, but it is a suitable raw material for many industries, including construction and interior design, paper, packaging, and the boat industry. After logging, sawmills are generally the first stage in timber processing regardless of the end-use industry. Their main task is to turn unprocessed logs from the forest into well-cut lumber. While this is a multi-stage process, the main step is the sawing itself that can be done by two of the main machine types: band saws and frame saws. The efficient operation of these machines have a significant impact on the whole plant, thus, several works have addressed this issue. Efficiency is tackled on both the operational and the planning level by designing cutting patterns and scheduling, respectively [6]. A tree log may be cut in different patterns yielding different quantities of different products, as shown by in Figure 1. Designing patterns based on long-term demands for products, statistics about defects, etc. is a well-researched topic in the literature. Moreover, scanning tools available on modern sawing machines can provide real-time information about logs to make such decisions adaptive [5]. On the planning level, selecting the cutting patterns and the corresponding log quantities for each shift while considering orders, available logs, storage capacity, etc. is a scheduling problem. The base problem was introduced by Zanjani et al. [7] and Maturana et al. [9] for the stochastic and deterministic cases, respectively. Subsequent works extended the approach in various directions, e.g., addressing uncertainties [2, 11], integrating cutting pattern selections [3, 10], or minimizing waste [4].

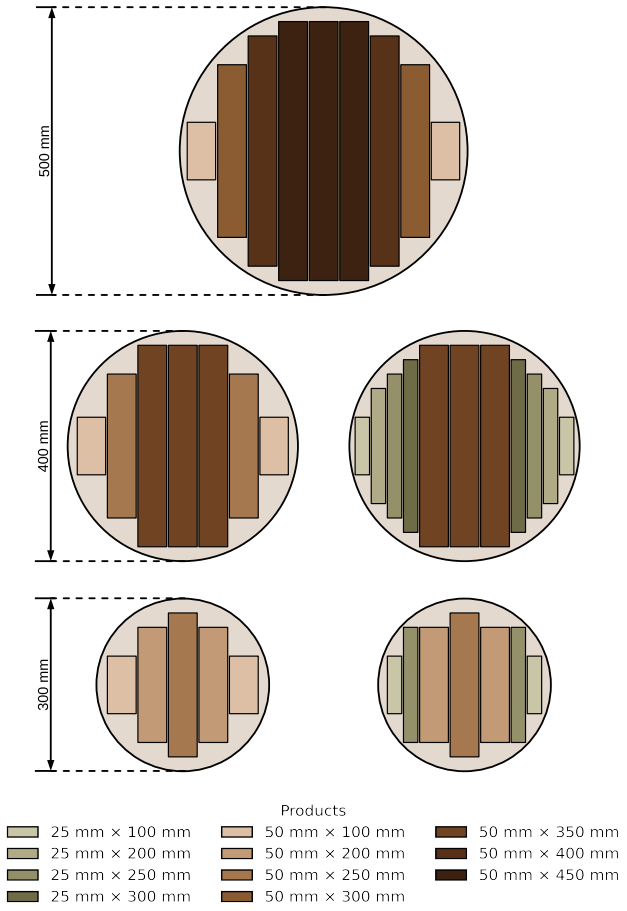


Figure 1: Cutting patterns for test cases in Section 5

In this paper, a previously proposed model by the authors [8] is further investigated, that addressed issues prevalent for small-scale sawmills: workforce availability and operation differences between sawing technologies. Compared to band saws, the changeover time for switching cutting patterns on a frame saw is non-negligible, thus the same formulation is not applicable in short-term scheduling. The proposed model introduced a significantly different formulation for the behavior of the frame saw, that only allows one change per

MATCOS-25, Koper, Slovenia
2025.

shift following industry practice. In later Sections this base model will be referred to as **C–EK**, where **EK** (expert knowledge) indicates at most one change in a shift, that may be timed continuously (**C**) at any time.

The aim of this work is to investigate, whether allowing more cutting pattern changes on a frame saw would yield significantly better results due to a broader solution space, and/or does it increase the computational costs comparably. While allowing such flexibility in a mathematical model may be simple, one has to keep in mind, that the execution of overly complicated schedules in an non-automated small-scale plant is not realistic, thus a set of models will also be introduced, where cutting pattern changes may only happen at predefined discrete (**D**) time points, as detailed in Section 4. Another decision freedom, whose effect will be investigated in Section 5 is the possible reallocation of the workforce from one machine to another during a shift, as discussed in Section 3.

2 Problem definition

The objective is the same as in [8]: minimizing the under-production cost in a small-scale sawmill equipped with one frame saw and one band saw. The sawmill may operate in multiple shifts, however, from the modeling point of view, only the total number of shifts is relevant. Thus, without the loss of generality, it is assumed, that each day has a single shift (of length H), and the two terms are used interchangeably. For each day ($d \in \mathcal{D}$), the number of specialists (HR_d^{SP}) and additional workers (HR_d^{AW}) are given, along with the requirements to operate each machine ($RR^{SP,F}$, $RR^{SP,B}$,$RR^{AW,F}$, $RR^{AW,B}$). Resource related data is also given, i.e., the quantity (I_l) and volume (V_l) of logs of different sizes ($l \in \mathcal{L}$) all available at the start of the planning horizon. For each log size several cutting patterns ($\mathcal{P}_l \subseteq \mathcal{P}$) may be available with known yields ($Y_{t,p}$) for lumber products ($t \in \mathcal{T}$). For each shift the production planner may decide which machines to operate and how many logs with which cutting patterns are processed.

By allowing different flexibilities, several different problem definitions will be addressed by their corresponding models:

- **C–EK** Original problem from [8]: the frame saw may change cutting pattern once per shift, but a saw is either operating through the whole shift or not.
- **C'–EK** Same as **C–EK**, but the frame saw may also be turned on/off, and the workforce reallocated to the band saw instead of changing the pattern, once per shift.
- **D[K]** The shift is subdivided into k segments of equal length. Pattern changes and workforce reallocations are allowed only at the end of these intervals.
- **D[K]–EK** Same as **D[K]**, but the frame saw is allowed to change cutting patterns at most once a day.

If $OPT(M)$ denotes the optimal (minimal) solution for the problem/model M, the following inequalities will hold naturally:

- $OPT(\textbf{C'–EK}) \leq OPT(\textbf{C–EK})$
- $OPT(\textbf{C'–EK}) \leq OPT(\textbf{D[}X\textbf{]–EK})$
- $OPT(\textbf{D1}) = OPT(\textbf{D1–EK})$
- $OPT(\textbf{D[}X\textbf{]}) \leq OPT(\textbf{D[}X\textbf{]–EK})$
- $OPT(\textbf{D[}Y\textbf{]}) \leq OPT(\textbf{D[}X\textbf{]})$ if $X \mid Y$
- $OPT(\textbf{D[}Y\textbf{]–EK}) \leq OPT(\textbf{D[}X\textbf{]–EK})$ if $X \mid Y$

3 Model with workforce reallocation: C'–EK

The event, when worker reallocation is also possible in addition to pattern changes on the frame saw, will be referred to as the changeover. As this model is based on **C–EK**, constraints (1)–(5), (7)–(8), and (10) from [8] are copied verbatim, expressing the objective, storage balances, daily production quantities, inventory limits, frame saw production bounds before the changeover, and frame saw activation bounds.

Binary variables w_d^F and w_d^B are split to $w_d^{F,-}$,$w_d^{B,-}$ and $w_d^{F,+}$,$w_d^{B,+}$, indicating whether the machines are active before or after the changeover. Duplicated versions of constraints (15)–(16) with these variables are also copied to express human resource needs. Superscripts + and − will indicate such division similarly in other variables, and ± will be used to indicate both cases.

The band saw time capacity constraint is reformulated as:

$$\sum_{p \in \mathcal{P}} ST_p^B \cdot q_{d,p}^B \leq \tau_d^{B,-} + \tau_d^{B,+} \qquad \forall d \in \mathcal{D} \quad (1)$$

Where variables $\tau_d^{B,\pm} \in [0, H]$ represent the available band saw cutting time, linked to changeover timing by the following constraints:

$$\tau_d^{B,\pm} \leq H \cdot w_d^{B,\pm} \qquad \forall d \in \mathcal{D},\ \pm \in \{-,+\} \quad (2)$$

$$\tau_d^{B,-} \leq t_d \qquad \forall d \in \mathcal{D} \quad (3)$$

$$\tau_d^{B,+} \leq H - t_d \qquad \forall d \in \mathcal{D} \quad (4)$$

Constraint (2) activates $\tau_d^{B,\pm}$ only when the band saw is active in the corresponding part of the shift, while constraints (3) and (4) bound them by t_d and $H - t_d$, respectively.

For the frame saw, the following constraint limits the production after the changeover:

$$\sum_{p \in \mathcal{P}} ST_p^F \cdot q_{d,p}^{F,+} \leq (H - t_d) - CT^F \cdot z_d^{F,CT} + H \cdot \left(1 - w_d^{F,+}\right) \qquad \forall d \in \mathcal{D} \quad (5)$$

Where $z_d^{F,CT} \in \{0, 1\}$ indicates whether changeover time must be deducted after the changeover. Constraint (6) sets $z_d^{F,CT}$ to one, if the frame saw is active through the whole shift and the pattern is changed.

$$z_d^{F,CT} \geq w_d^{F,-} + w_d^{F,+} + \left(s_{d,p}^{F,+} - s_{d,p}^{F,-}\right) - 2 \qquad \forall d \in \mathcal{D},\ p \in \mathcal{P} \quad (6)$$

Constraint (7) ensures that only an active frame saw can have an active cutting pattern, which is preserved by constraints (8) and (9) for the next day if the saw remains active.

$$\sum_{p \in \mathcal{P}} s_{d,p}^{F,\pm} = w_d^{F,\pm} \qquad \forall d \in \mathcal{D},\ \pm \in \{-,+\} \quad (7)$$

$$s_{d,p}^{F,+} \geq s_{d+1,p}^{F,-} - 1 \cdot \left(2 - w_d^{F,+} - w_{d+1}^{F,-}\right) \qquad \forall d \in \mathcal{D},\ p \in \mathcal{P} \quad (8)$$

$$s_{d,p}^{F,+} \leq s_{d+1,p}^{F,-} + 1 \cdot \left(2 - w_d^{F,+} - w_{d+1}^{F,-}\right) \qquad \forall d \in \mathcal{D},\ p \in \mathcal{P} \quad (9)$$

4 Discrete time-slot models: D[K], D[K]–EK

Similarly to the previous modification of the original model, some constraints from [8] remain unmodified, expressing the objective, storage balances: (1)–(3). Moreover, some variables are now defined for each segment of a shift, indexed by the set $\mathcal{K} = \{1, 2, \ldots, K\}$.

For example, the binary variables $w^F_{d,k}$, $w^B_{d,k}$ indicate machine usage, which are used in segment-wise copies of (15)–(16) of [8] to express human-resource requirements.

$w^B_{d,k}$ is also used in the band saw time capacity constraint, where $H^K = H/K$ denotes the length of a segment:

$$\sum_{p\in\mathcal{P}} ST^B_p \cdot q^B_{d,p} \le \sum_{k\in\mathcal{K}} H^K \cdot w^B_{d,k} \qquad \forall d \in \mathcal{D} \quad (10)$$

Another such variable is $q^F_{d,k,p} \in \mathbb{Z}_{\ge 0}$ indicating the frame saw production, used in constraints (11) and (12) to calculate the daily production quantity and limit the inventory:

$$y_{d,t} = \sum_{p\in\mathcal{P}} Y_{t,p} \cdot V_{l_p} \cdot \left(q^B_{d,p} + \sum_{k\in\mathcal{K}} q^F_{d,k,p} \right) \qquad \forall d \in \mathcal{D},\ t \in \mathcal{T} \quad (11)$$

$$\sum_{d\in\mathcal{D}} \sum_{p\in\mathcal{P}_l} \left(q^B_{d,p} + \sum_{k\in\mathcal{K}} q^F_{d,k,p} \right) \le I_l \qquad \forall l \in \mathcal{L} \quad (12)$$

Active consecutive segments on the frame saw should be merged together if the pattern is unchanged. These conditions are expressed by binary variable $s^F_{d,k,p}$ which is 1 iff pattern p is active in segment d, k and the one preceding it. Another binary indicator variable, $s^{c,F}_{d,k,p}$ is 1 iff p is active in segment k but not in the preceding one. This logic is enforced by the following constraints:

$$\sum_{p\in\mathcal{P}} \left(s^{c,F}_{d,k,p} + s^F_{d,k,p} \right) = w^F_{d,k} \qquad \forall d \in \mathcal{D},\ k \in \mathcal{K} \quad (13)$$

$$s^F_{d,k,p} \le s^F_{d,k-1,p} + s^{c,F}_{d,k-1,p} + \left(1 - w^F_{d,k-1}\right) \qquad \forall d \in \mathcal{D},\ k \in \mathcal{K}\setminus\{1\},\ p \in \mathcal{P} \quad (14)$$

$$s^F_{d,1,p} \le s^F_{d-1,K,p} + s^{c,F}_{d-1,K,p} + \left(1 - w^F_{d-1,K}\right) \qquad \forall d \in \mathcal{D}\setminus\{1\},\ p \in \mathcal{P} \quad (15)$$

Constraint (13) ensures that when the frame saw is operated in segment k, exactly one pattern is selected (either a changeover to p or continuation with p), and when it is idle, neither indicator is active. Constraint (14) allows pattern continuation only if p was active in the previous segment, and the frame saw is active. The same condition is set across days is by Constraint (15).

The merging of active segments with the same pattern is modeled by constraints (16), (17), and (18), introducing a spare time variable $\sigma_{d,k,p} \in [0, H^K]$ that records unused time in segment k for pattern p and can be carried to the next segment.

$$\sigma_{d,k,p} \le ST^F_p \cdot s^F_{d,k+1,p} \qquad \forall d \in \mathcal{D},\ k \in \mathcal{K}\setminus\{K\},\ p \in \mathcal{P} \quad (16)$$

$$ST^F_p \cdot q^F_{d,1,p} \le \left(H^K - CT^F\right) \cdot s^{c,F}_{d,1,p} + H^K \cdot s^F_{d,1,p} - \sigma_{d,1,p} \qquad \forall d \in \mathcal{D},\ p \in \mathcal{P} \quad (17)$$

$$ST^F_p \cdot q^F_{d,k,p} \le \left(H^K - CT^F\right) \cdot s^{c,F}_{d,k,p} + H^K \cdot s^F_{d,k,p} + \sigma_{d,k-1,p} - \sigma_{d,k,p} \qquad \forall d \in \mathcal{D},\ k \in \mathcal{K}\setminus\{1\},\ p \in \mathcal{P} \quad (18)$$

Constraint (16) limits the carryover from segment k to one log's sawing time ST^F_p and activates it only if the next segment continues with the same pattern, while the production time bounds follow from Eqs. (17) and (18), accounting for changeover time CT^F and the inflow/outflow of spare time.

To obtain D[K]–EK from D[K], Constraint (19) needs to be added, which enforces at most one cutting pattern change by the frame saw per day:

$$\sum_{k\in\mathcal{K}} \sum_{p\in\mathcal{P}} s^{c,F}_{d,k,p} \le 1 \qquad \forall d \in \mathcal{D} \quad (19)$$

5 Empirical results

To evaluate the new approaches, the randomly generated test set from our previous study [8] is used, where cutting pattern yields were precomputed with Pitago Optimizers [1] and technical parameters reflect industry practice and recommendations by experts. Figure 1 shows the five patterns defined for three log diameter classes in all of the test cases.

For practical reasons in execution, the values 1, 2, 4, and 8 were considered for K, and the **EK** variant for $K = 1$ is omitted, as it is equivalent to the non-**EK** case.

All problems in the dataset were solved with all of the models by Gurobi Optimizer 12.0.1 on a computer with an Apple M2 CPU and 16 GB of RAM available, with a time limit of 1500 seconds.

Table 1 shows the aggregated results from the 50 cases with 4-week planning horizon.

The left block shows objective value distributions (m^3) as box plots. The median relative deviation from the **C–EK** reference is also reported in the column labeled $\tilde{\Delta}_{\text{C–EK}}$ (%). On the right, CPU times (s) are shown as box plots on a base-10 logarithmic scale.

The results match the theory:

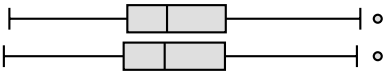
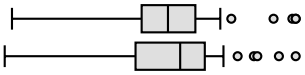
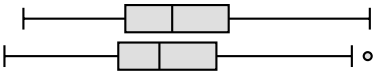
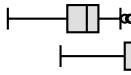
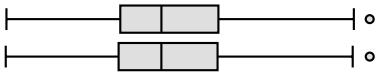
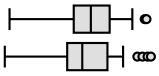
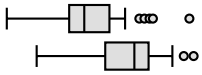
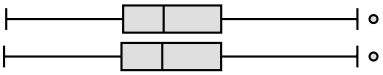
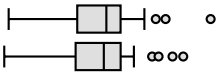
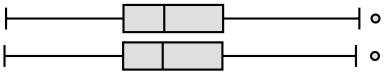
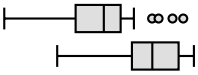
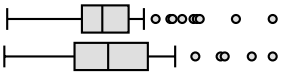
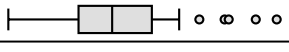
- Adding on/off reallocation in **C′–EK** increases quality
- Discretizing the changeover timing degrades it when only one cutting pattern change is allowed per shift (**EK** models), while smaller slot lengths mitigate the loss.
- Imposing the "at most one change" rule on the frame saw (**–EK**) has a negative impact on solution quality.

It can be observed, however, that the differences in objective values across models are small: $\tilde{\Delta}_{\text{C–EK}}$ ranges from −0.1072% for **C′–EK** (best) to +0.8536% for **D1** (worst), and difference between D[K] and D[K]–EK is less than 0.05 percentage points.

A key takeaway is that even **D8** does not outperform **C–EK**. For these test cases, continuously timing a single change is more valuable than permitting multiple changes at fixed time-slot boundaries. Even so, the loss from discretization is small: as k increases, the median gap shrinks from +0.85% (**D1**) to about +0.0058% for **D8–EK** or +0.0024% for **D8**.

Regarding runtimes, increasing K for the discrete models increases CPU time, as expected, yet the $K = 8$ variants still run faster

Table 1: Results on randomly generated instances with a 4-week planning horizon across model variants.

Model	Objective value (m^3) 950 1025 1100 1175 1250	$\tilde{\Delta}_{\texttt{C–EK}}$ (%)	CPU time (s) 10^{-2} $10^{-0.625}$ $10^{0.75}$ $10^{2.125}$ $10^{3.5}$	Avg.
C–EK		+0.0000		151.36
C′–EK		-0.1072		125.98
D1		+0.8536		0.24
D2–EK		+0.2908		1.37
D2		+0.2440		1.52
D4–EK		+0.1065		5.00
D4		+0.0760		4.21
D8–EK		+0.0058		67.76
D8		+0.0024		82.82

on average than the continuous baselines. Notably, **D8** is within ≈ 0.11 percentage points of **C′–EK** in objective value yet is faster on average. The **–EK** restriction leaves quality essentially unchanged and tends to reduce runtime.

Time-limit hits were infrequent (Table 2).

Table 2: Time-limit hits (counts).

Model	3-week	4-week	5-week	6-week
C–EK	3	2	5	4
C′–EK	0	1	2	3
D8–EK	0	1	1	0
D8	0	1	1	1

These counts suggest that the discrete-slot models remain computationally stable across longer planning horizons.

Overall, the discrete time-slot-based models offer a tunable trade-off: coarse slotting is extremely fast but less accurate, whereas $K = 8$ achieves near-continuous quality at a substantially lower average runtime than the continuous baselines.

6 Concluding remarks

We revisited a MILP scheduling model for small-scale sawmills and examined three design choices: (i) allowing workforce reallocation, (ii) restricting changeovers to discrete within-shift time-slot boundaries, and (iii) allowing multiple changes per shift. Computational tests on a large synthetic instance set show that objective differences across variants are modest. Allowing on/off operation with reallocation provides a consistent, albeit small, improvement in solution quality, whereas discretizing the changeover into time slots introduces a controllable loss that diminishes as the slot granularity is refined. Overall, the discrete formulations offer a tunable trade-off between accuracy and speed. In our setting, even coarse slots perform well, while a moderate refinement (e.g., hourly segments) achieves near-continuous quality at substantially lower average runtimes. In practice, multiple within-shift changes bring marginal gains and complicate execution, whereas **–EK** yields simpler plans with similar quality and often lower runtime. For the investigated test cases, even completely forbidding changes within shifts resulted in less than 1 percent quality reduction. These findings support the discrete-slot approach as a practical planning tool for small, low-automation sawmills. Future work will expand the set of test instances, include an industrial case study to validate and calibrate the models on operational data, and, where appropriate, incorporate uncertainty in key inputs.

References

[1] *Pitago Optimizers.* Retrieved August 24, 2025 from https://pitago.eu/

[2] Pamela P Alvarez and Jorge R Vera. 2014. Application of robust optimization to the sawmill planning problem. *Annals of Operations Research* 219 (2014), 457–475. doi:10.1007/s10479-011-1002-4

[3] Diego Broz, Nicolás Vanzetti, Gabriela Corsano, and Jorge M. Montagna. 2019. Goal programming application for the decision support in the daily production planning of sawmills. *Forest Policy and Economics* 102 (2019), 29–40. doi:10.1016/j.forpol.2019.02.004

[4] Giacomo Da Col, Philipp Fleiss, Alice Tarzariol, Erich C. Teppan, and Elena Wiegelmann. 2025. A Declarative Approach to Tackle Sawmill Production Scheduling with Answer Set Programming. In *Computational Science and Computational Intelligence*, Hamid R. Arabnia, Leonidas Deligiannidis, Farid Ghareh Mohammadi, Soheyla Amirian, and Farzan Shenavarmasouleh (Eds.). Springer Nature Switzerland, Cham, 313–323. doi:10.1007/978-3-031-90341-0_23

[5] Kamran Forghani, Mats Carlsson, Pierre Flener, Magnus Fredriksson, Justin Pearson, and Di Yuan. 2024. Maximizing value yield in wood industry through flexible sawing and product grading based on wane and log shape. *Computers and Electronics in Agriculture* 216 (2024), 108513. doi:10.1016/j.compag.2023.108513

[6] Seyed Mohsen Hosseini and Angelika Peer. 2022. Wood Products Manufacturing Optimization: A Survey. *IEEE Access* 10 (2022), 121653–121683. doi:10.1109/ACCESS.2022.3223053

[7] Masoumeh Kazemi Zanjani, Daoud Ait-Kadi, and Mustapha Nourelfath. 2010. Robust production planning in a manufacturing environment with random yield: A case in sawmill production planning. *European Journal of Operational Research* 201, 3 (2010), 882–891. doi:10.1016/j.ejor.2009.03.041

[8] Csaba Kebelei and Mate Hegyhati. 2024. Sawmill scheduling: an application-oriented model. *Acta Technica Jaurinensis* 17, 3 (2024), 104–110. doi:10.14513/actatechjaur.00743

[9] Sergio Maturana, Enzo Pizani, and Jorge Vera. 2010. Scheduling production for a sawmill: A comparison of a mathematical model versus a heuristic. *Computers & Industrial Engineering* 59, 4 (2010), 667–674. doi:10.1016/j.cie.2010.07.016

[10] Nicolás Vanzetti, Diego Broz, Gabriela Corsano, and Jorge M. Montagna. 2018. An optimization approach for multiperiod production planning in a sawmill. *Forest Policy and Economics* 97 (2018), 1–8. doi:10.1016/j.forpol.2018.09.001

[11] Mauricio Varas, Sergio Maturana, Rodrigo Pascual, Ignacio Vargas, and Jorge Vera. 2014. Scheduling production for a sawmill: A robust optimization approach. *International Journal of Production Economics* 150 (2014), 37–51. doi:10.1016/j.ijpe.2013.11.028

Implementation of a Vehicle and Driver Scheduling Model: a Case Study

Viktor Árgilán
University of Szeged,
Juhász Gyula Faculty of Education,
Department of Applied Informatics
Szeged, Hungary
viktor.sandor.argilan@szte.hu

József Békési
University of Szeged, Institute of Informatics,
Department of Foundations of Computer Science
Szeged, Hungary
bekesi@inf.u-szeged.hu

Gábor Galambos
University of Szeged,
Juhász Gyula Faculty of Education,
Department of Applied Informatics
Szeged, Hungary
GalambosGabor@szte.hu

Imre Papp
University of Szeged,
Juhász Gyula Faculty of Education,
Department of Applied Informatics
Szeged, Hungary

Abstract

During combined vehicle and driver scheduling, we have to plan the daily work of vehicles and their drivers so that they perform a set of tasks at the lowest possible cost. The tasks are defined by time intervals, and the vehicles are located in different depots. In recent decades, several mathematical models have been defined, with which we can create regular schedules and search for optimal solutions based on different objective functions. However, in practical problems, there are many requirements that cannot be handled easily and usually have a significant computational demand. The results of our research and development project are presented through a case study based on real experiments carried out at the Budapest Transport Corporation.

Keywords

Optimization, Vehicle and driver scheduling problem, Public transport

1 Introduction

Operating costs are usually a significant item in the budget of public transport providers. The main components of these costs are vehicle fleet acquisition costs, fuel and maintenance costs, and driver wages. With the help of various decision support systems, comprehensive solutions for both the vehicle and the driver have been developed in recent decades to solve the optimization task. In public transport, vehicle and driver scheduling can be very complex. In theory, we generally look for a global optimum that minimizes both vehicle-related costs and driver scheduling costs. These two types of costs affect each other, so it's usually best to handle the tasks together [2].

If we want to find the optimal solution, combined vehicle and driver scheduling mathematical optimization models can be used. There are several such methods in the literature. The vehicle scheduling problem is usually formulated as a multicommodity network flow problem ([4], [10], [12]). The optimal schedule can be calculated as the solution of an integer programming problem. Other models are also known, for example the problem can be formulated as a set partitioning problem (see e.g. [14], [7]).

The Generate and Select (GaS) method is the most well-known technology for the driver scheduling part. In the initial phase, a substantial number of standard shifts are generated. In the subsequent selection phase, a subset of these regular shifts is selected to minimize cost and optimize coverage of trips. It is noteworthy that the execution of both phases requires substantial computational resources. The extent of this computational demand is contingent upon the number of trips and the intricacy of the operational guidelines. The selection phase can be modeled as a set covering or set partitioning problem.

In 2005, Huisman et al. [9] extended the former combined models and algorithms of the single-depot case [5, 6] to the multi-depot version. This was the first general mathematical formulation of the combined multi-depot problem. Later many authors investigated this version of the problem (see, for example, [8], [13], [15], [16]).

In practice, however, it turns out that there are many company-specific details and constraints that cannot be uniformly addressed by general systems, but which are important for the transport companies. As an example, if a transport company also uses alternative fuel vehicles in its fleet, their scheduling must take into account the number of kilometers per refueling, known as the radius, which can be much lower than the mileage of a conventional fuel vehicle. Such cases have been examined for example in [1] and [11].

In the paper [3] Békési and Nagy presented how the methods used in the above mentioned papers were adapted to develop a decision support system for the Budapest Transport Corporation. The aim of this project was to automatically calculate optimal or approximately optimal vehicle and driver schedules for a given list of trips based on the master data and the company specific requirements and parameters in compliance with labor regulations.

This paper overviews how the complete integration was implemented and what kind of specific developments were necessary to take into account all the practical requirements of the company.

2 The Automatic Solving Process

We summarize the system's key characteristics, requirements, input data, and settings based on [3]. All the data required for a computation is kept in packages so that it

MATCOS-25, Koper, Slovenia
2025. https://doi.org/10.1145/nnnnnnn.nnnnnnn

can automatically solve a particular problem. Trips from a single line or a collection of lines are typically included in a package.

The following details are included in each input package:

- line data,
- end stations, depots and their parameters,
- trips with details about the time and place,
- number of vehicles along with details about their type and availability,
- the limits of labor laws and break rules,
- parking capacities of stations, parking lots, and depots provided at 5-minute intervals for each type of vehicle
- driver change possibilities, break and detour permissions

The following conditions must be met by the problem's solution:

- every trip must be covered by a single vehicle and driver schedule,
- the number of vehicles specified in the package is the maximum number that can be used,
- schedules for drivers and vehicles must be consistent,
- parking regulations and fuel consumption must be followed,
- between two trips, the necessary technological and compensatory times must be maintained,
- driver change regulations must be followed.

There are three phases in our process. In the first phase a directed graph is produced after the relevant input data and parameters have been read. We consider a number of "side-conditions" when creating the graph, including location data, vehicle types, labor laws, technological and compensatory times, and other package elements. In the second phase all regular driver schedules are generated. Here, only schedules that comply with all labor laws are accepted. The third phase involves building and solving a mathematical model. The details of the model are available in [3]. If the solution is successful, an output package is used to send the results back to the company's information system after being read from the solver.

3 Implementation details

3.1 Parallel schedule generation

We employ the depth-first search strategy to generate the regular shifts, beginning with the special departure depot vertices of the graph. We implemented the shift generation as a parallel algorithm because it can be quite time-consuming. At this point, the paths of the graph that begin at the departure depot vertex and conclude at the arrival depot vertex reflect the shifts. As a first stage for the parallel generation, we create and save in a list every potential prefix of the paths from the departure depot vertex to a specified short depth (for example, 4).
The following is a summary of the procedure.

- Each thread chooses an unprocessed path-prefix from the list and uses it to create every potential regular shift in the main portion of the generation.
- Following the completion of the operation, the thread chooses a fresh unprocessed prefix, generates its shifts, and saves the generated shifts.
- The thread ends if no shifts remain unprocessed.
- The generation is complete if every thread is terminated.

3.2 Smart node contraction

According to [3], the greedy strategy is the fundamental method of node-contraction. This method allows us to eliminate a large number of edges from the graph. As a result, several opportunities for breaks and driver changes are also eliminated. It can occasionally result in infeasibility. We created an intelligent node-contraction algorithm after realizing this issue. Compared to the greedy approach, this algorithm retains more break and driver change possibilities based on its parameter values. The pseudocode of the algorithm is the following.

Algorithm 1 Smart Trip Grouper

```
1:  procedure groupTripsSmart(n:Integer, S : Set of Trips)
2:      for all T ∈ S do
3:          Next(T) ← The closest compatible trip to T
4:      for all T ∈ S do
5:          if Merged(T) = false then
6:              actTrip ← T
7:              MList ← ∅
8:              i ← 1
9:              while i ≤ n do
10:                 if i > 1 and actTrip.time - prevTrip.time ≥ MAXT_NOPROTECT then
11:                     if actTrip.hasInBreakArc or prevTrip.hasOutBreakArc then
12:                         Exit while
13:                     if actTrip.hasInDriverChArc or prevTrip.hasOutDriverChArc then
14:                         Exit while
15:                 Merged(actTrip) ← true
16:                 MList ← MList ∪ actTrip
17:                 prevTrip ← actTrip
18:                 i ← i + 1
19:                 if Next(actTrip) ≠ null then
20:                     actTrip ← Next(actTrip)
21:                 else
22:                     Exit for
23:             Add MList to the output
```

Similar to the greedy strategy, as the initial step of the algorithm, for each trip vertex in the graph, we search for the closest trip vertex to it. As a result, we get chains of subsequent trip vertices here as well. After that we take the trip vertex v that have not been included in a group before. Starting from v, we move forward on the chain at most $n-1$ times and decide whether we include the actual vertex in the current group or not. If the time difference between the time of the previous and the current trip vertex is less than or equal to the value of the parameter MAXT_NOPROTECT, then we proceed as for the greedy strategy. Otherwise, if the vertex before the actual vertex has an outgoing break arc, or the actual vertex has an incoming break arc, then we stop and the actual vertex is no longer included in the merging with vertex v. We do the same with the driver change arcs.

3.3 Parking and vehicle number constraints for several types of vehicles

The parking capacities for various vehicle types are also specified in our problem. The sizes of the categories vary. The parking lots of the larger categories may also be used by smaller ones. The parking lots of smaller categories are off-limits to larger categories. MDVSP-based models usually handle different vehicle categories by incorporating vehicle types into physical depots. This means that the number of depots in the theoretical model will be a multiple of the number of physical depots, as their number should be multiplied by the number of vehicle categories. Depots defined in this way can also be called logical depots.

Based on [3] we present shortly how the model handles the depots. We use the following notations in the description:

$G = (V, A)$	The graph used for the representation of the problem.
D	The set of depots.
U	The set of trips, represented by the nodes in G.
A_d	The set of arcs belonging to depot $d \in D$.
A_q^d	The set of arcs which should be covered by both vehicle and driver ($A_q^d \subseteq A_d$).
V_d	The set of trip vertices belonging to depot $d \in D$.
D_u	The set of depots, from which u can be served.
v^+	The set of all outgoing arcs of vertex $v \in V$ in G.
v^-	The set of all incoming arcs of vertex $v \in V$ in G.
e^+	The head vertex of $e \in A$ in G.
e^-	The tail vertex of $e \in A$ in G.
$at(d)$	The arrival vertex of depot $d \in D$ in G.
t_v	The running distance up to trip vertex v in its vehicle schedule.
L	A constant larger than the longest possible running distance.
δ_e	The running distance of arc e.
r_d	The maximal allowed running distance for the vehicles in depot $d \in D$.
k_d	The number of vehicles available in depot $d \in D$.
S_d	The set of valid driver schedules generated for depot $d \in D$
$S_d(v)$	The set of driver schedules containing the trip vertex $v \in V_d$ ($S_d(u) \subseteq S_d$).
$T_1, ..., T_m$	The time slots, for which parking capacities should be checked.
H	The set of those stations and parking locations that can be used by the vehicles.
$A_h^d(T)$	The set of those arcs that covers time slot T at location $h \in H$ ($A_h^d(T) \subseteq A_d$).
$p_h^d(T)$	The number of parking places at location h in time slot $T \in \{T_1, ..., T_m\}$ for vehicles of depot $d \in D$.
$S_d(e)$	The set of driver schedules containing the arc $e \in A_d$ ($S_d(e) \subseteq S_d$).
c_s^d	The cost of schedule $s \in S_d$.
c_e^d	The cost of arc $e \in A_d$.
x_e^d	Boolean variable indicating whether the arc e from depot d is included in the solution or not.
y_s^d	Boolean variable indicating whether the schedule $s \in S_d$ is included in the solution or not.

$$\min \quad \sum_{d \in D} \sum_{s \in S_d} c_s^d y_s^d + \sum_{d \in D} \sum_{e \in A_d} c_e^d x_e^d$$

Subject To

$$\sum_{d \in D_u, e \in u_d^+} x_e^d = 1, \quad \forall u \in U \tag{1}$$

$$\sum_{e \in v_d^+} x_e^d - \sum_{e \in v_d^-} x_e^d = 0, \quad \forall v \in V_d, \forall d \in D, \tag{2}$$

$$t_{e^+} \geq t_v + \delta_e - (1 - x_e)L, \tag{3}$$

$$\forall v \in V \setminus \{at(d) \mid d \in D\}, \forall e \in v^+ \tag{4}$$

$$t_{e^-} + \delta_e + x_e L \leq r_d + L \tag{5}$$

$$\forall v \in \{at(d) \mid d \in D\}, \forall e \in v^- \tag{6}$$

$$\sum_{e \in at(d)^-} x_e^d \leq k_d, \quad \forall d \in D, \tag{7}$$

$$\sum_{s \in S_d(v)} y_s^d - \sum_{e \in v_d^+} x_e^d = 0, \quad \forall v \in V_d, \forall d \in D \tag{8}$$

$$\sum_{s \in S_d(e)} y_s^d - x_e^d = 0, \quad \forall e \in A_q^d, \forall d \in D. \tag{9}$$

$$\sum_{e \in A_h^d(T)} x_e^d \leq p_h^d(T), \tag{10}$$

$$\forall T \in \{T_1, ..., T_m\}, \forall h \in H, \forall d \in D \tag{11}$$

$$x_e^d, y_s^d \in \{0, 1\}, t_v \geq 0, \quad \forall e \in A_d, \forall s \in S_d, \tag{12}$$

$$\forall d \in D, \forall v \in V \setminus \{at(d) \mid d \in D \tag{13}$$

4 Conclusions

In this paper, we reviewed how the optimization model solving the combined vehicle and driver scheduling problem was implemented in practice. We presented what specific developments were necessary in order to be able to take into account all the practical requirements that arose. We presented what implementation technologies we would use to facilitate the solvability of the problem for critical inputs, when in the basic case there was no solution, or the methods used for the solution proved to be too slow. We performed tests on real problems to verify the effectiveness of the technologies.

References

[1] Adler, J.D., Mirchandani P.B. (2016). The vehicle scheduling problem for fleets with alternative-fuel vehicles.Transportation Science, 51(2), 441–456, 2016.

[2] Békési, J., Brodnik, A., Krész, M., and Pas, D. (2009). An Integrated Framework for Bus Logistics Management: Case Studies, In: S. Voss, J. Pahl and S. Schwarze (eds.), Logistik Management: Systeme, Methoden, Integration, Springer, 389–411.

[3] Békési, J., Nagy, A. (2020). Combined Vehicle and Driver Scheduling with Fuel Consumption and Parking Constraints: a Case Study. *Acta Polytechnica Hungarica*, 17(7): 45–65.

[4] Bodin, L., Golden, B., Assad, A. and Ball, M. (1983). Routing and Scheduling of Vehicles and Crews: The State of the Art. *Computers and Operations Research*, 10: 63–211.

[5] Freling, R., Huisman, D. and Wagelmans, A.P.M. (2003). Models and algorithms for integration of vehicle and crew scheduling. *Journal of Scheduling*, 6: 63–85.

[6] Haase, K., Desaulniers, G. and Desrosiers, J. (2001). Simultaneous vehicle and crew scheduling in urban mass transit systems, *Transportation Science*, 35(3): 286–303.

[7] Hadjar, A., Marcotte, O. and Soumis, F. (2001). A Branch-and-Cut Algorithm for the Multiple Depot Vehicle Scheduling Problem. *Tech. Rept.* G–2001–25, Les Cahiers du Gerad, Montreal.

[8] Horváth, M., Kis, T. (2019), Computing strong lower and upper bounds for the integrated multiple-depot vehicle and crew scheduling problem with branch-and-price. Central European Journal of Operations Research, 27(1), pp 39–67.

[9] Huisman, D., Freling, R. and Wagelmans, A.P.M. (2005). Multiple-depot integrated vehicle and crew scheduling. *Transportation Science*, 39: 491–502.

[10] Kliewer, N., Mellouli, T. and Suhl, L. (2006). A time-space network based exact optimization model for multi-depot bus scheduling, *European Journal of Operational Research*, 175: 1616–1627.

[11] Li, J-Q. (2013). Transit bus scheduling with limited energy, *Transportation Science*, 48(4), 521–539.

[12] Löbel, A. (1997). Optimal Vehicle Scheduling in Public Transit, *Ph.D. thesis*, Technische Universitaet at Berlin.

[13] Mesquita, M., Moz, M., Paias, A., Paixao, J., Pato, M. and Respício, A. (2011), A new model for the integrated vehicle-crew-rostering problem and a computational study on rosters, *Journal of Scheduling*,14, pp 319–334.

[14] Ribeiro, C.C., Soumis, F. (1994). A Column Generation Approach to the Multiple-Depot Vehicle Scheduling Problem. *Operations Research*, 42(1): 41–52.

[15] Steinzen, I. (2007), Topics in integrated vehicle and crew scheduling in public transit. *PhD thesis*, University of Paderborn.

[16] Steinzen, I., Gintner, V., Suhl, L. and Kliewer, N. (2010), A Time-Space Network Approach for the Integrated Vehicle- and Crew-Scheduling Problem with Multiple Depots, *Transportation Science*, 44, pp 367–382.

ALGatorGraph: A Java Library for Graph Generation and Manipulation within the ALGator System

Boštjan Hren and Tomaž Dobravec
University of Ljubljana
Faculty of Computer and Information Science
Ljubljana, Slovenia

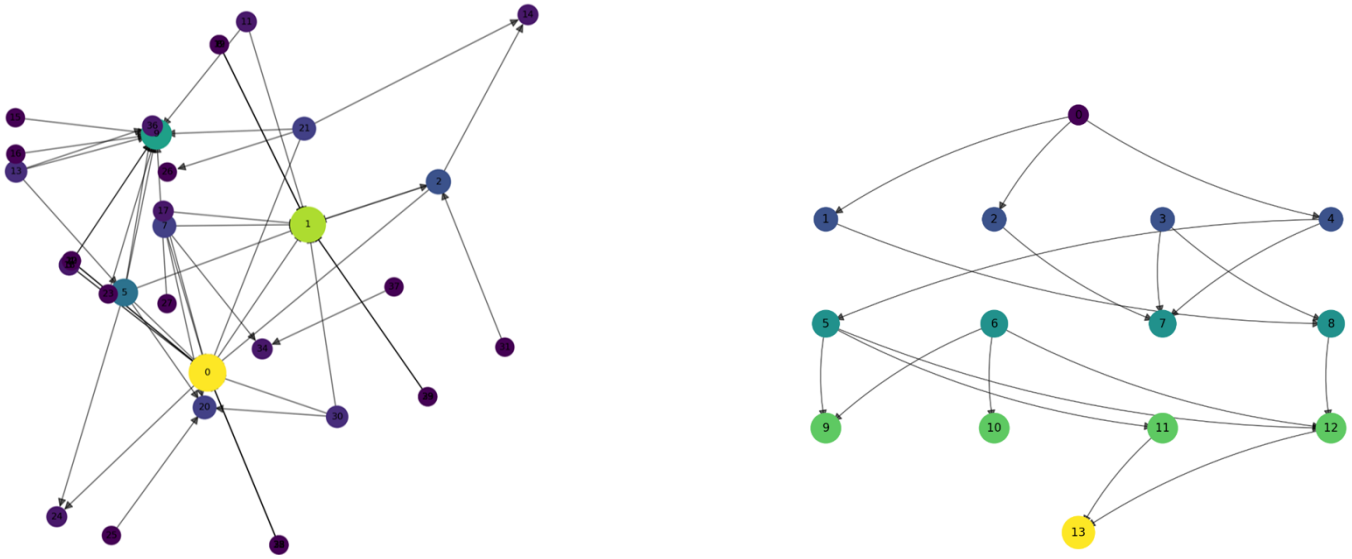


Figure 1: The `DIRECTED_SCALE_FREE` and `LAYERED` graphs generated with the ALGatorGraph library.

Abstract

In this paper, we present the ALGatorGraph Java library, an extension of the ALGator framework for graph-related problems. It provides a unified interface for graph data structures and supports generation of diverse graphs. A Maximum Flow problem case study shows how easily algorithms can be implemented and experimentally evaluated within ALGator using the ALGatorGraph library.

CCS Concepts

• **Theory of computation → Algorithm design techniques**; *Graph algorithms analysis*; *Approximation algorithms analysis*; **Data structures design and analysis**.

Keywords

Experimentation, algorithm evaluation, graph problems

1 Introduction

The development and analysis of algorithms has long been a central theme in computer science, traditionally rooted in rigorous theoretical frameworks. Theoretical computer science provides formal guarantees about algorithmic correctness, complexity, and optimality, often under idealized assumptions [1]. While this foundation remains indispensable, the increasing complexity of computational environments and problem domains has revealed the limitations of purely theoretical analysis. In response, several complementary methodologies have emerged—experimental algorithmics, algorithm engineering, and the broader empirical approach—each contributing unique perspectives and tools for studying algorithms in practice.

Experimental algorithmics [3] focuses on the empirical study of algorithms through computational experiments. Borrowing from the scientific method, it emphasizes reproducibility, hypothesis testing, and data-driven insights. Rather than proving abstract bounds, researchers design experiments to observe algorithm behavior across a wide range of inputs, environments, and configurations. This helps to uncover performance trends, identify bottlenecks, and challenge theoretical assumptions with real-world evidence.

Closely related, but distinct, is algorithm engineering, which integrates design, analysis, implementation, and experimental evaluation into a cohesive development cycle [5]. Treating algorithms as both mathematical objects and software components, algorithm engineering emphasizes modular design, rigorous implementation, and iterative refinement, often leveraging hardware characteristics and system-level concerns to improve performance. In parallel, the empirical approach provides a broader umbrella, extending to benchmarking, comparative evaluation, and large-scale testing, prioritizing observation and measurement over formal proofs.

While these methodologies differ in emphasis, they share a commitment to complementing theoretical analysis with practical evidence.

Experimental results not only validate formal claims but also expose behaviors, trade-offs, and phenomena that purely theoretical reasoning may overlook. In this landscape, dedicated tools and frameworks play a critical role: they provide researchers with environments in which reproducibility, transparency, and fairness of experiments are guaranteed.

ALGator is one such framework [2]. Designed for the systematic evaluation of algorithms, it enables researchers to compose testing environments in which multiple algorithms can be implemented, tested, and compared across diverse test cases and test sets. By ensuring repetition, fair judgment, transparency, and the public availability of results, ALGator supports rigorous experimental practice while fostering collaboration and reproducibility in the research community.

Building on this foundation, ALGatorGraph extends ALGator with native support for directed and undirected graphs, a central domain in algorithmics. Many of the most challenging and widely studied algorithmic problems—from shortest paths and network flows to clustering and graph isomorphism—are inherently graph-based. By facilitating the use of graphs as both input and output data structures, ALGatorGraph lowers the barrier for integrating graph-related projects into the experimental environment of ALGator. This allows researchers not only to implement and test graph algorithms more easily but also to benefit from the reproducible and transparent evaluation framework that ALGator provides.

With ALGator and ALGatorGraph, experimental algorithmics is equipped with a robust infrastructure for studying graph algorithms under controlled and repeatable conditions. This integration highlights the importance of aligning theoretical insights, empirical experimentation, and practical implementation—ultimately advancing the broader field of experimental algorithmics.

2 The ALGatorGraph library

The `ALGatorGraph` library is a new ALGator library that represents a key component for working with graphs within the system. The library extends ALGator's core functionalities by providing specialized tools for graph construction, transformation between different representations, and graph management within the ALGator framework. The `ALGatorGraph` library builds upon the `JGraphT` library [4], leveraging its core data structures while extending essential functionality for constructing and managing graphs within the ALGator testing environment. The primary components enabling graph construction and manipulation are the `GraphCreator` and `GraphConverter` classes.

`GraphCreator`. This class provides a comprehensive set of capabilities for generating graphs with diverse topologies. It supports over 50 different types of graphs, ranging from basic structures such as complete and bipartite graphs to more complex forms, including Petersen graphs, lattices, hypercubes, and various special graphs (e.g., the diamond graph, the Buckminsterfullerene graph, or the Zachary's Karate Club graph). Each graph type is accessible through a simple interface, allowing the user to specify the desired structure using an enum value along with relevant parameters. The class also supports random graph generation according to various models, including the Barabási–Albert model, the Watts–Strogatz small-world model, and the Erdős–Rényi model of random graphs.

The `GraphCreator` class also includes the method called `importGraphsFromFolder()`, which enables reading graphs from various file formats, including popular formats such as GraphML, CSV, DOT, and JSON. This functionality is essential for integrating external data sources into the ALGator system, allowing algorithm testing on real-world networks and graphs. The importer automatically detects the file format and appropriately processes both the graph structure and any edge weights, providing a high degree of flexibility when working with diverse input data. The function returns the data as `JGraphT` objects. `JGraphT` is a robust library that serves as the foundation for the classes we have developed and implemented. We selected it as the most effective library in the graph domain, and it is therefore used to facilitate graph import.

The `ALGatorGraph` package also includes utility functions for graph description, where the `getGraphDescription` method provides a human-readable summary of the properties and parameters for each graph type, significantly facilitating system usage. The package is designed to simplify the integration of graph algorithms into the ALGator system while maintaining a high degree of flexibility and adaptability for various types of analyses. The combination of automated graph generation, import capabilities, and robust methods for converting graphs from files or `JGraphT` objects makes this package an invaluable tool for developers aiming to test and compare algorithms on graphs within a controlled environment.

`GraphConverter`. The `GraphConverter` class enables the conversion of graphs from various `JGraphT` formats into ALGator's native graph representations. This class provides methods for converting both directed and undirected graphs, supporting both simple edges (`DefaultEdge`) and weighted edges (`DefaultWeightedEdge`). Special attention is given to the robust handling of different node types: the class automatically recognizes and converts both numeric and string nodes into a unified integer format, ensuring consistency when working with algorithms. With this converter, nearly any graph representation can be imported into ALGator for algorithm testing.

3 Graph data structure

The classes in the ALGatorGraph package support working with both directed and undirected graphs. The `DEdge<V, W>` class represents a directed edge with a source vertex, a target vertex, and a weight, where the order of vertices is significant. It is used together with the `DGraph<V, W>` class, which implements a directed graph using adjacency lists (via a `HashMap`) and supports generic types for vertices (`V`) and weights (`W`). Methods include adding/removing vertices and edges, as well as querying neighbors, with time complexities indicated in the comments (e.g., `O(1)` for adding a vertex).

For undirected graphs, the `Edge<V, W>` and `Graph<V, W>` classes are provided. The `Edge` class stores two vertices (`vertex1`, `vertex2`) and a `weight`, where the vertex order is irrelevant (e.g., the edge A–B is equivalent to B–A). The `Graph` class implements an undirected graph using adjacency lists, where adding an edge A–B automatically adds B–A. Both classes override the `equals()`, `hashCode()`, and `toString()` methods and provide operations similar to their directed counterparts, but adapted for undirected edges.

The main difference lies in directionality: DEdge/DGraph require an explicit direction (e.g., for paths), whereas Edge/Graph treat edges as symmetric (e.g., for friendship networks). Both graph types are optimized for fast adjacency-list operations and are compatible with the JGraphT library.

4 Usage Example: The MaxFlow Project

To illustrate the usage of the ALGatorGraph library, we developed a simple example project focused on the Maximum Flow problem [6]. The primary goal of this project is not to provide a comprehensive analysis or optimization of the problem itself, but rather to demonstrate how a project can be structured around graph-based data. In this example, the core data structure is a graph, implemented as an object from the ALGatorGraph library, highlighting how the library facilitates graph creation, manipulation, and integration within a computational workflow. This approach showcases the practical application of ALGatorGraph for managing complex network structures and serves as a template for building more sophisticated graph-based projects in the ALGator environment.
To use the ALGatorGraph library in the project, the corresponding JAR packages must be imported, thereby providing access to the classes and methods within the ALGatorGraph package. This is accomplished by configuring the ALGator ProjectJAR array variable. In the following we list the properties that were defined in the project.

Parameters. The names and types of the parameters required to generate input for the test. Source (int) and Sink (int) are used in every test to specify the source and sink vertices in the graph. Nodes (int) and Edges (int) are used for generating DIRECTED_SCALE_FREE graphs, determining the number of vertices and edges in the graph. Layers (int) and NodesPerLayer (int) are used when generating a LAYERED graph to specify the number of layers and the number of vertices per layer. For examples of both types of graphs, see Figure 1. Folder (String) provides the path to the directory containing graph files; in our case, these are the graphs and layered_graphs folders within the project directory. NumOfGraph (int) is used when the folder contains multiple graph files, indicating which graph to use in sequence.

Generators. In the project we defined several generators to provide different types of tests.

- Type0 and Type3 generators produce explicitly specified graphs that are used for trivial testing. Graphs are created using a simple API; for example:

```
DGraph<Integer, Integer> graph = new DGraph<>();
for (int i = 0; i < 6; i++) {
  graph.addVertex(i);
}
graph.addEdge(0, 1).setWeight(3);
// ...
```

  The only inputs to the generator are the Source and Sink parameters, allowing a limited set of basic tests to be performed.
- Type1 generator receives additional nodes and edges parameters and generates a random graph with a specified number of vertices and edges.

```
String[] args = { "DIRECTED_SCALE_FREE",
                  0.3, 0.9, 0.1, 0.2, edges, nodes};
Graph graph = GraphCreator.generateGraph(args);
```

- Type4 generator generates a layered graph with the given layerCount and nodesPerLayer parameters. In this case, a method createLayeredGraph(int layerCount, int nodesPerLayer, double connectionProbability) is used. In all the cases the probability of creating edges between vertices in consecutive layers, connectionProbability, is set to 0.6.
- Type2 and Type5 generators receives the folderName and the graphNumber parameters and import the corresponding graph from the given folder using the GraphCreator.importGraphsFromFolder() method..

Input. A Java class used to store the input data of a test case contains a triple (graph, source, sink) and is provided to algorithms as an argument.

```
public class Input extends AbstractInput {
  DGraph<Integer, Integer> graph;
  int source;
  int sink;
}
```

Output. Output contains an integer representing the maximum flow; this object is returned by the algorithm.

```
public class Output extends AbstractOutput {
  int maxFlow;
}
```

Indicators. In our project, we have an indicator called Check, which verifies whether the algorithm's output is correct by performing a simple comparison with the expected output. The method IndicatorTest_Check returns a result of "OK" or "NOK" in case of failure. Additional indicators of algorithm success can also be added, returning more complex objects if needed.

```
public class IndicatorTest_Check extends
  AbstractIndicatorTest<TestCase, Output> {

  @Override
  public Object getValue(TestCase tc, Output output) {
    return
      output.maxFlow
        ==
      tc.getExpectedOutput().maxFlow ? "OK" : "NOK";
  }
}
```

Algorithms. In the project, we implemented two algorithms:

- Edmonds–Karp implementation of the Ford–Fulkerson algorithm, which uses BFS to find augmenting paths and achieves a time complexity of $O(nm^2)$.
- Dinic's algorithm, which combines BFS and DFS to compute the maximum flow with a time complexity of $O(n^2m)$.

According to ALGator's convention, an algorithm is represented by a class that implements an execute() method, which receives an Input object as a parameter and returns an Output object as the result, as shown in the following listings.

```
// An implementation of the EdmondsKarp algorithm
public class Algorithm extends ProjectAbstractAlgorithm {

  protected Output execute(Input input) {
```

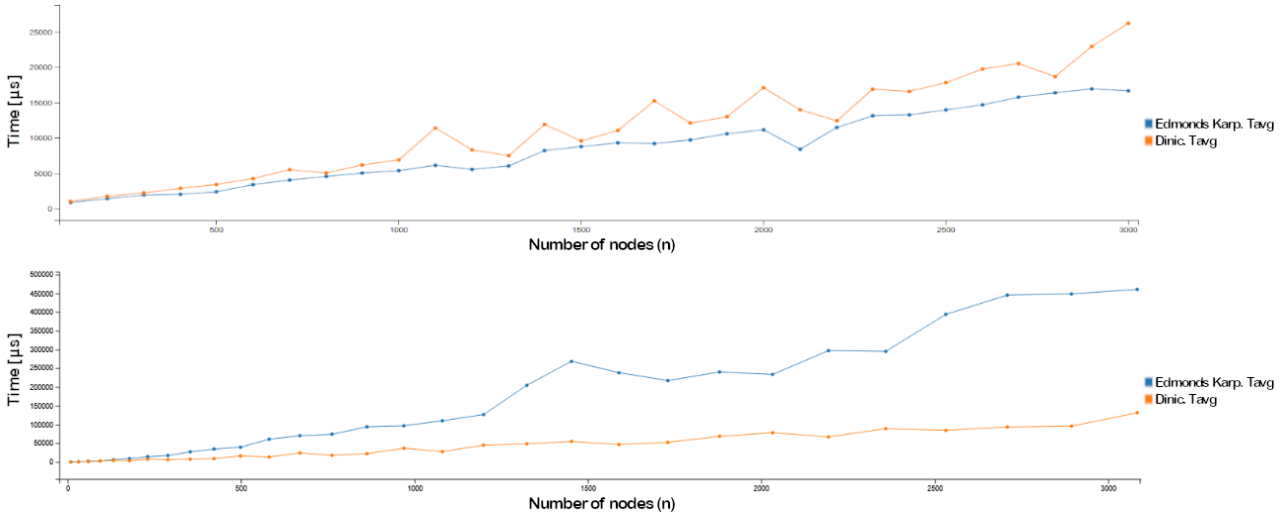


Figure 2: Comparison of the complexities of the Edmonds-Karp and Dinic algorithms on (a) `DIRECTED_SCALE_FREE` graphs (upper chart) and (b) `LAYERED` graphs (lower chart).

```
    DGraph<Integer, Integer> graph = input.graph;
    int source = input.source;
    int sink = input.sink;

    int maxFlow = edmondsKarp(graph, source, sink);

    Output output = new Output(maxFlow);
    return output;
  }
}
```

Testsets. In the project, several test sets were implemented. The first test set contains directed scale-free graphs generated by a `Type1` test set generator. The test cases were created using the following for-loop–based description:

```
$for{i,1,30,1}:Type1::0:${100*i-1}:${100*i}:${500*i}
```

This generates 30 `Type1` test cases with parameters $n = 100 * i$, $m = 500 * i$, $source = 0$, and $sink = 100i - 1$, for $i = 1, 2, \ldots, 30$. Running the algorithms on this test set yields the results shown in Figure 2 (upper chart). For this configuration, the complexity of both algorithms increases with the number of nodes. On average, the Edmonds–Karp algorithm performs about 33% faster than Dinic's algorithm on this test set. Further tests revealed that the advantage of the Edmonds–Karp algorithm on directed scale-free graphs is greater when edges are evenly distibuted. As the disparity between node degrees increases, this advantage diminishes, and in some cases Dinic's algorithm even outperforms Edmonds–Karp algorithm.

Another test set used in this project consists of layered graphs generated by a `Type4` test set generator. The test cases were defined using the following for-loop–based description:

```
$for{i,1,30,1}:Type4::0:${(i*3-2)*(i+5)+2-1}:${i*3}:${i+5}
```

This generates layered graphs with $3 * i$ layers and $i + 5$ nodes per layer, for $i = 1, 2, \ldots, 30$. Running both algorithms on this test set shows that, for these layered graphs, Dinic's algorithm is faster than Edmonds–Karp algorithm (see Figure 2, bottom chart). The average speedup in this scenario is 2.9x.

5 Conclusions

The ALGatorGraph Java library has been developed as a supporting tool for ALGator projects that address graph-related problems. Its main benefits are twofold: it offers a unified interface for implementing graph data structures, and it provides the capability to generate a wide variety of graphs—both through parameter-based generation and by reading graphs from the library.

Through the MaxFlow project we have demonstrated how straightforward it is to implement graph-related problems in ALGator using ALGatorGraph. The results obtained in the experiments highlight some characteristics of the two implemented algorithms, but their specific values are not the main point. The real significance lies in how easily such results can be obtained by combining ALGatorGraph's ability to handle graph data structures with ALGator's functionality for conducting and analyzing experiments.

References

[1] S. Arora and B. Barak. 2009. *Computational Complexity: A Modern Approach.* Cambridge University Press.

[2] Tomaž Dobravec. 2025. ALGator: Online Implementation of the System for Automated Algorithm Execution and Evaluation. (2025). https://algator.fri.uni-lj.si

[3] Catherine C. McGeoch. 2012. *A Guide to Experimental Algorithmics.* Cambridge University Press.

[4] Dimitrios Michail, Joris Kinable, Barak Naveh, and John V. Sichi. 2020. JGraphT–A Java Library for Graph Data Structures and Algorithms. *ACM Trans. Math. Softw.* 46, 2, Article 16 (May 2020), 29 pages.

[5] Matthias Müller-Hannemann. 2010. *Algorithm Engineering, Bridging the Gap Between Algorithm Theory and Practice.* Springer-Verlag Berlin and Heidelberg GmbH & Co. K.

[6] Ruiheng Zhang. 2024. Network Flow of Graph Theory and Its Application. In *Proceedings of the 1st International Conference on Engineering Management, Information Technology and Intelligence - EMITI.* INSTICC, SciTePress, 692–696. https://doi.org/10.5220/0012968800004508

Engineering CSFLOC: A Subsumption-Driven Clause-Counting SAT Solver

Gábor Kusper
kusper.gabor@uni-eszterhazy.hu
Eszterházy Károly Catholic University
Eger, Hungary

Abstract

Clause counting, a technique from the #SAT domain, offers a complementary angle to conflict-driven clause learning (CDCL). We revisit CSFLOC (*Counting Subsumed Full-Length Ordered Clauses*) and present an engineered implementation that turns its theoretical "last-1-bit" observation into practical speedups. The solver represents full-length clauses with a binary counter and performs sign-aware bucket scans keyed by the last-literal index, allowing safe jumps that skip a block of consecutive full-length clauses subsumed by input clauses. We map the known pseudocode of CSFLOC line-by-line to Java, detailing the data structures that make the inner loop fast: POS/NEG buckets per index, small effected/learned caches, and carry-like counter updates that implement jumps of size 2^{n-j} without big integers. Preordering (variable renaming) strategies—composable flags `B,C,H,I,R,S,W`—shape the distribution of last-literal indices and, via an island/strait view of the counter, bias the solver toward longer trailing 1-blocks and shorter 0-gaps. A concise empirical snapshot shows where CSFLOC is competitive (over-constrained, dense-structure instances) and where CDCL remains preferable. Source code is available at `http://fmv.ektf.hu/tools.html`.

Keywords

CSFLOC, SAT solving, clause counting

1 Introduction

The propositional satisfiability problem (SAT) asks whether a propositional CNF formula has an assignment of truth values to its variables that makes the formula true. SAT is one of the most-researched NP-complete [7] problems in computer science, with applications ranging from theoretical computer science and artificial intelligence to hardware design and formal verification [4].

Modern SAT solvers build on conflict-driven clause learning (CDCL), an extension of the classical Davis Putnam Logemann Loveland (DPLL) procedure [9] that adds conflict analysis, clause learning, and non-chronological backtracking (backjumping) [4].

An interesting question is how many models a SAT instance has; this is the #SAT problem [11, 15, 3], i.e., counting the satisfying assignments of a CNF formula [10].

CSFLOC (*Counting Subsumed Full-Length Ordered Clauses*) is a clause-counting approach that enumerates full-length clauses via a binary counter and uses subsumption to jump over blocks. It was introduced in [14, 13]. Although CSFLOC is a general SAT solver, it is inspired by classical #SAT techniques such as full-length clause counting.

MATCOS-25, 9–10 October 2024, Koper,Slovenia

This paper briefly recalls CSFLOC and focuses on an optimized Java implementation.

1.1 Related work

Modern SAT solving is dominated by conflict-driven clause learning (CDCL) with powerful preprocessing and restart heuristics; see the updated *Handbook of Satisfiability* for a broad survey of techniques and applications [4]. Alongside CDCL, a classical but distinct line is *clause counting* and inclusion–exclusion–based reasoning, which reasons about sets of full-length (maximal) clauses and their subsumption structure.

Early work by Iwama established counting-style satisfiability tests with average-case guarantees [11]. Lozinskii developed exact propositional model counting (the so-called #SAT problem) in [15], while Birnbaum–Lozinskii connected counting tightly to Davis–Putnam–style branching [2]. A complementary strand is Andrei's "inverting resolution," which frames satisfiability counting via inverse propositional resolution and normalization [1]. These works already emphasize that ordering, last-literal positions, and subsumption can unlock substantial contiguous "jumps" in the enumeration space.

The inclusion–exclusion principle has been repeatedly explored for model counting and even SAT itself. Bennett and Sankaranarayanan proposed an inclusion–exclusion counter with subsumption pruning for k-SAT [3], while Zaleski implemented a SAT solver in Maple using inclusion–exclusion and Bonferroni inequalities [16]. More recently, inclusion–exclusion has been combined with dynamic programming over small treewidth for projected model counting (PMC), yielding practically competitive PMC/#SAT solvers [8]. These results corroborate the general message behind CSFLOC: structural regularities (e.g., small treewidth, strong subsumption) can be exploited to skip large contiguous blocks in the search space.

CSFLOC itself belongs to the full-length clause–counting family. It is the successor of the `Optimized CCC` algorithm [14]. It uses a counter to count subsumed full-length clauses. By studying `Optimized CCC` we observed that its full-length clause counter can be increased on its last 1 bit in the best case [13]. CSFLOC is based on this observation. It also uses a data structure in which the clauses are ordered by the index of their last literal. These two improvements result in a faster algorithm which can compete with a state-of-the-art SAT solver on problems with lots of clauses, like Black-and-White 2-SAT problems [6] and weakly nondecisive SAT problems [5].

1.2 Revisiting the CSFLOC algorithm

CSFLOC was introduced in [13]. In this subsection, we recall its pseudocode and the main theoretical results.

The following properties underpin Algorithm 1; we state them without proof, since these theorems are already proven in [13].

Lemma 1 (Observation 2.). *In the inner loop of CSFLOC, if k =IndexOfLastPositiveLiteral(C) and j is the last-literal index of a*

Algorithm 1 CSFLOC(S)

Require: S is a non-empty list of ordered clauses with variable index function I.
Ensure: If S is satisfiable it returns a solution of S, otherwise returns the empty set.

```
1:  n := number of variables in S;
2:  S[i] := {C|C ∈ S∧IndexOfLastLiteral(C) = i}, where i = 1..n;
3:  count := 0;
4:  while count < 2^n do
5:      increment := 0;
6:      C :=FullLengthClauseRepresentationOf(count);
7:      for j :=IndexOfLastPositiveLiteral(C); j <= n; j := j + 1
    do
8:          if ∃D ∈ S[j] such that D subsumes C then
9:              increment := 2^(n−j);
10:             j := n + 1;
11:         end if
12:     end for
13:     if increment = 0 then
14:         return ¬C;
15:     else
16:         count := count + increment;
17:     end if
18: end while
19: return {};
```

subsuming clause $D \in S[j]$, then $j \geq k$, hence the largest possible jump is 2^{n-k}.

Theorem 1 (Soundness and Completeness of CSFLOC). *Algorithm 1 is sound and complete.*

The 2. Observation Lemma, i.e., Lemma 1 states that in case of CSFLOC the largest possible jump is 2^{n-k}, and Theorem 1 states that this trick is valid, i.e., CSFLOC is a general SAT solver algorithm. See the proof of these theoretical results in [13]. In this paper we discuss the practical implementation of CSFLOC.

2 Pseudocode-to-Java Mapping

The Java implementation referenced in this paper is available at `http://fmv.ektf.hu/files/CSFLOC19.java`. This section is highly technical. We suggest consulting the Java source code to understand this section.

2.1 Entry Points and Call Flow

The Java implementation uses the following functions and flow of calls:

- **main** calls
- `DIMACSReader` (parse DIMACS CNF), which calls
- `HighLevelReader` (variable ordering, bucket build), which calls
- `CSFLOCSolver.CSFLOC_v7()` (outer loop), which calls
- `usingBestClause_v4` (inner scan), which calls
- `increaseCounter_v5` (jump)
- the outer loop returns eventually a model or UNSAT;
- if a model is found, then main calls `SimpleCheckSolution` which validates the model.

2.2 Element-by-Element Mapping

We map each step of Algorithm 1 with concrete classes and methods.

Buckets $S[1..n]$ **(lines 1–2):** built by `HighLevelReader` using two Clause arrays
- `clauseListOrderedByLastVarIndexPos[i]` and
- `clauseListOrderedByLastVarIndexNeg[i]` and $i \in \{1..n\}$,

storing clauses with IndexOfLastLiteral(C) = i whose last literal is positive / negative.

Counter init **(line 3):** *count* is represented by the Boolean vector `counter[1..n]` (1 = positive, 0 = negative). Implementation uses a practical initial value: the *best black clause*; it sets the first 1-bit and calls `addEffectedClause` to prime caches.

Full-length clause C **(lines 4–6):** Not materialized: the current `counter` encodes C; subsumption is tested directly via methods like `Clause.subsumedBy(counter)`.

Setting j **(line 7):** Computed as the index of the last 1-bit in `counter`; passed as `index` to `usingBestClause_v4(index, counter)`.

Scan $j..n$ ***for a subsumer*** **(line 7):** One of the main methods `usingBestClause_v4(i)` performs a *sign-aware* traversal. It tries to find a subsumer clause by testing
- `effected/oldEffected` at level i,
- `learnedClausesPos` at level i
- `clauseListOrderedByLastVarIndexPos` at level i,
- `clauseListOrderedByLastVarIndexNeg` from level i+1 in a loop.

This realizes the theoretical $j..n$ loop once the sign pattern of C is taken into account.

Subsumption test **(line 8):** Bit-level subsumption check is done by `D.subsumedBy(counter)`.

Increment **by** 2^{n-j} **(line 9):** is done by the other main function `increaseCounter_v5` which executes the jump with carry-like bit flips on `counter`, equivalent to adding 2^{n-j} to the integer counter; hooks update caches and may insert learned clauses via `addEffectedClause`.

Return $\neg C$ **(line 14):** Complement the `counter` and emit the model (optionally map the model back to the original variable indices if a renaming was applied).

Termination **(lines 15–19):** The outer loop in the method `CSFLOC_v7()` repeats until a model is returned or the jumps exhaust $[0, 2^n)$, in which case UNSAT is reported.

2.3 Supporting Data Structures and Caches

- **Sign-aware buckets**: In the pseudocode $S[j]$ denotes the set of clauses whose last-literal index is j. In the implementation we maintain two arrays per index, $S^+[j]$ and $S^-[j]$, for clauses whose last literal is positive or negative, respectively; these are the Java arrays
  – `clauseListOrderedByLastVarIndexPos[j]` and
  – `clauseListOrderedByLastVarIndexNeg[j]`.
  The inner scan first probes $S^+[j]$ and then $S^-[k]$ for $k > j$, matching the positive/negative hit order.
- **Effected / oldEffected**: per-index, per-sign small working sets prioritized before base buckets.
- **Learned clauses**: bounded per-index pools
  – `learnedClausesPos`,
  – `learnedClausesNeg`
  filled after successful subsumption hits; probed before base buckets.

- **Best black clause fields**: `bestBlackClause` is used to set the first index before the main loop to gain a small speed-up.
- **Renaming metadata**: permutation *translate* maintained by `HighLevelReader` (see section 4); used for bucket indices and for mapping models back.

3 Counter Dynamics, Hits, Islands, and Straits

In this section, we introduce the intuitive notions of *positive hit*, *negative hit*, *islands* and *straits* to help explain how CSFLOC works.

Hit types. Let C be the current full-length clause encoded by *counter*, and let $k = \text{IndexOfLastPositiveLiteral}(C)$ denote the index of the last 1-bit in C. For $j \in \{k, \ldots, n\}$ we say there is a *positive hit at level j* if there exists a clause $D \in S[j]$ whose *last literal is positive* and $D \subseteq C$. For $j > k$ we say there is a *negative hit at level j* if there exists a clause $D \in S[j]$ whose *last literal is negative* and $D \subseteq C$. Because positions greater than k are 0s in C, a positive last literal cannot subsume C at those levels, hence only negative hits are possible above k. In both cases, CSFLOC jumps by 2^{n-j} (see Lemma 1).

CSFLOC enumerates full-length clauses through a *counter* which is a Boolean array (1=positive, 0=negative) and leverages subsumption to jump. In the *best case*, the next iteration in the main loop increases the index of the *last 1-bit* by exactly 1, i.e., the trailing 1-block grows by one. In the general case, adding 1 in base-2 may also flip higher 1-bits to 0 (carry), so a single step can translate into a *large jump* in the integer view of *count*.

Islands and straits. We interpret *counter* as a sea of zeros with occasional ones. A contiguous block of 1s is an *island*; a contiguous block of 0s between islands is a *strait*. Let the rightmost island (closest to index n) be the *last island*. Growing the last island by one (i.e., increasing the last 1-bit index by one) is the locally optimal step: it preserves previously gained structure and maximizes the chance of a larger safe jump because carries only affect positions to the left of the last island.

When is a one-step growth possible? A one-step growth occurs if there exists a clause D whose last-literal index equals $j = \text{IndexOfLastPositiveLiteral}(C)$ and $D \subseteq C$ (a positive hit at level j). Otherwise the best we can hope for is that the next 1-bit sits immediately to the left of j (i.e., a negative hit at level j+1), so the last island still grows by one. If the nearest 1-bit is farther left, the intervening 0s form a wider strait, which negatively affects the performance of CSFLOC.

Implications for ordering and renaming. Because the inner scan starts at $j = \text{IndexOfLastPositiveLiteral}(C)$ and then checks for negative hits at $k > j$, we profit from (i) *large islands* (long trailing 1-blocks) and (ii) *small straits* between successive islands. This aligns with runtime data: fewer, longer islands typically enable larger 2^{n-j} jumps, whereas many short islands correlate with smaller increments and more scans. Our preordering strategies aim to bias the distribution of last-literal indices accordingly; see the next section, especially the I (Island), S (Strait), and C (Clustering) variable-renaming strategies.

Why no dynamic renaming? One might attempt to dynamically adjust the order so that the next 1-bit always follows the last one. However, the implementation reaches clauses by their *last literal* (a variant of watched-literal indexing). Swapping two variables (say, 8 and 25) can change the last-literal index in many clauses (e.g., $\{-1, 8, 23\}$ becomes $\{-1, 25, 23\}$, moving the last literal from 23 to 25). Maintaining bucket memberships and all derived caches at runtime would impose a prohibitive *burden*, so our solver performs renaming only *once* before the main loop (see section 4).

In conclusion, CSFLOC runs faster when the last island is long and the straits between islands are short. Proving these observations in full generality is challenging; for random 3-SAT, however, parts of them can be established rigorously. A complete theoretical treatment remains an open line of research.

The next section provides more information on variable renaming strategies.

4 Variable Renaming Strategies

CSFLOC scans buckets from $j = \text{IndexOfLastPositiveLiteral}(C)$ upwards. Hence the *variable order* determines the distribution of last-literal indices and which buckets are hit early. Renaming aims to (i) concentrate strong constraints early, (ii) lengthen useful trailing 1-bit islands in the counter (larger jumps), and (iii) improve cache locality.

Interface. The solver accepts a composite string `variableRenamingStrategy` consisting of letters from `{B,C,H,I,R,S,W}` (uppercase or lowercase). Strategies are applied in the order of appearance; lowercase variants indicate a milder weighting/priority. Typical presets in the implementation include: `"IWCR"`, `"HWCR"`, `"BHWCR"` for random 3-SAT, and `"B"` for pigeonhole families.

List of strategies and method names.

B : renameBlackClauses. Renames variables so that *black clauses* (all-negative) receive smaller last-literal indices. Intuition: black clauses seed the formation of 1-bit *islands* in the counter; a long trailing island implies a high-probability large jump. Bringing black clauses forward reduces warm-up and favors long final islands or at least smaller straits.

W : renameWhiteClauses. Symmetric to **B**. It orders *white clauses* (all-positive). It reduces fragmentation caused by early all-positive buckets and helps separate positive hits from later negative scans. Used together with **B** to shape polarity structure.

S : renameStraitClauses. A clause is *strait* if it contains exactly one negative literal (all others are positive). Such clauses tend to start a new island. This strategy assigns a low index to the unique negative variable (and, if still unassigned, to the remaining variables of these clauses), thereby shrinking the 0-gaps (straits) between islands and lowering last-literal positions.

I : renameIslandClauses. Symmetric to **S** (which targets clauses with exactly one *negative* literal), **I** targets clauses with exactly one *positive* literal (all others negative), i.e., definite Horn clauses. This strategy moves the unique positive variable to the largest index in the clause (its *last literal*), but still assigns it a small global index, increasing the chance of positive hits and, in the island–strait view, tending to cancel the last island.

H : renameDefiniteHornClauses. Uses the same syntactic filter as **I** (clauses with exactly one positive literal), but is conservative: it processes a clause only if *none* of its literals has been renamed by any earlier strategy. Otherwise, it is the same as **I**. Because **H** triggers only on untouched clauses, it composes well with polarity-oriented strategies such as **W** (white) and **B** (black); the combined effect is

Table 1: Where CSFLOC excels vs struggles.

Instance type	Best solver	Intuition
Random SAT (uf50/uuf50)	Glucose	weak structure
Pigeonhole (small n)	Glucose	black and white
WSN BW 2-SAT	CSFLOC	over-constrained
WnD UNSAT	CSFLOC	many clauses
SM, dense >10%	CSFLOC	over-constrained
SM, sparse <10%	Glucose	few subsumers

to lower last-positive positions and stabilize long trailing islands.

R : simplerVariableRenaming. It orders variables by their occurrence counts; high-frequency variables result in earlier hits.

C : clusterVariables. Clustering orders variables so that variables that frequently co-occur in clauses are placed close to each other. The cluster size is controlled by the *clustering factor* (an integer ≥ 2). Implementation-wise, we compute variable-pair frequencies, greedily form and merge clusters up to the factor, and then assign low consecutive indices cluster by cluster (applying the resulting translate to the formula). This shortens gaps between variables that tend to appear together, compresses last-literal positions, and improves the hit rate in the positive/negative hit scans. The option is enabled by including `C` in the renaming string and can be combined effectively with **R**, **H**, and **S** (and their compositions). Following Jebelean's original clustering idea [12], our experiments confirm it is the most consistently helpful preordering among the options we evaluated.

Heuristic notes. Promoting black clauses (**B**) early tends to create fewer but longer trailing islands, enabling larger jumps; many short islands are typically unfavorable, though we do not state formal lemmas due to space. Island- and Horn-aware orders (**I**,**H**) often dominate on over-constrained inputs; randomized tie-perturbation inside clusters mitigates adversarial cases.

5 Empirical Snapshot

This section may contain informal terms due to lack of space, but these have been clarified in [13].

The tests were done on iMac macOS Sierra (CPU: 2,5GHz Intel Core i5, Memory: 4GB 1333MHz DDR3).

We follow the measurement settings of our previous report [13]: the Java implementation of CSFLOC is compared against off-the-shelf CDCL baselines (Glucose 3.0[1]) on standard suites (SATLIB uf/uuf and pigeonhole), generator-based families (WnD UNSAT), a Black-and-White 2-SAT model of wireless sensor networks (WSN), and graph-induced SAT encodings (SM/BM/SBB). Timeouts and machine details are kept fixed across solvers; instances are run in a single thread.

Where CSFLOC shines (and where it does not). CSFLOC is not a random-SAT solver; its strengths appear on over-constrained or structurally dense inputs where subsumption enables large jumps.

Effect of preordering (clustering and renaming). Preordering is crucial for CSFLOC because the inner loop scans by last-literal index and sign. A simple clustering of variables into small contiguous groups (section 4) consistently improves run time on SATLIB: uf50 becomes notably faster; uf75/uf100, which tend to time out without clustering, become solvable under 2–3 variable clusters. Frequency- and Horn-biased orders (R/H) further reduce the last-positive indices encountered, increasing positive hits.

[1] http://www.labri.fr/perso/lsimon/glucose/

Table 2: Qualitative impact of clustering on SATLIB.

Suite	no clustering	cluster-2	cluster-3
uf50	*baseline*	↓ run time	↓↓ run time
uf75	often timeout	✓ many solved	✓✓ most solved
uf100	timeout	✓ some solved	✓✓ many solved

Graph-induced encodings. On SM/BM/SBB encodings derived from directed graphs, we observe a density threshold: for edge densities above roughly 10%, CSFLOC tends to outperform Glucose; below that, CDCL remains preferable. The effect aligns with the island/strait picture: higher density increases the chance that a positive hit occurs or that a nearby negative bucket contains subsumers, extending the last island and triggering larger jumps.

6 Conclusion and Future Work

We engineered a faithful and fast implementation of CSFLOC, clarifying how the last-1-bit observation can be realized efficiently via bucketed, sign-aware data structures, clause learning and variable preordering. Future work includes parallel traversal of counter ranges, richer learning schemes, and broader benchmarks.

References

[1] S.Andrei, Counting for Satisfiability by Inverting Resolution, *Artificial Intelligence Review*,Volume 22, Issue 4,339–366, 2004.

[2] E. Birnbaum, E. L. Lozinskii, The Good Old Davis-Putnam Procedure Helps Counting Models, *Journal Of Artificial Intelligence Research*, Volume 10, pages 457–477, DOI: https://doi.org/10.1613/jair.601, 1999.

[3] H. Bennett and S. Sankaranarayanan, Model Counting Using the Inclusion-Exclusion Principle, *Theory and Applications of Satisfiability Testing - SAT 2011 Lecture Notes in Computer Science*, Volume 6695, 362–363, 2011.

[4] A. Biere, M. Heule, H. van Maaren, T. Walsh, Handbook of Satisfiability, 2nd. edition, *IOS Press*, Amsterdam, 2021.

[5] Cs. Biro and G. Kusper, How to generate weakly nondecisive SAT instances, *Proceedings of 11th International IEEE Symposium on Intelligent Systems and Informatics (SISY)*, 265–269, Subotica, 2013.

[6] Cs. Biro and G. Kusper, Equivalence of Strongly Connected Graphs and Black-and-White 2-SAT Problems, *Miskolc Mathematical Notes*, accepted manuscript, MMN-2140.

[7] S. A. Cook, The Complexity of Theorem-Proving Procedures, *Proc. of STOC'71*, 151–158, 1971.

[8] J. K. Fichte, et al., Solving Projected Model Counting by Utilizing Treewidth and its Limits, *Artificial Intelligence* 314:103810, 2023.

[9] M. Davis, G. Logemann, D. Loveland, A Machine Program for Theorem Proving, *Communications of the ACM*, Volume 5, 394–397, 1962.

[10] Carla P. Gomes, Ashish Sabharwal, and Bart Selman, Model Counting, *Chapter 20 of Handbook of Satisfiability*, IOS Press, Amsterdam, 2009.

[11] K. Iwama, CNF-satisfiability test by counting and polynomial average time, *SIAM Journal on Computing*, Volume 18, Issue 2, 385–391, 1989.

[12] T. Jebelean and G. Kusper, Multi–Domain Logic and its Applications to SAT, (invited talk), *SYNASC'08*, DOI: 10.1109/SYNASC.2008.93, IEEE Computer Society Press, ISBN 978-0-7695-3523-4, 3–8, 2008.

[13] G. Kusper, Cs. Biró, Gy. B. Iszály, *SAT solving by CSFLOC, the next generation of full-length clause counting algorithms*, Proceedings of IEEE International Conference on Future IoT Technologies 2018, DOI: 10.1109/FIOT.2018.8325589, 2018.

[14] G. Kusper, Cs. Biró, *Solving SAT by an Iterative Version of the Inclusion-Exclusion Principle*, Proceedings of SYNASC 2015, DOI: 10.1109/SYNASC.2015.38, pp. 189–190, 2015.

[15] E. L. Lozinskii, Counting propositional models, *Information Processing Letters*, Volume 41, 327–332, 1992.

[16] A. Zaleski, Solving Satisfiability using Inclusion-Exclusion, *arXiv preprint*, DOI: arXiv:1712.06587, 2017.

Non-redundant Systems of Independence Atoms in Relational Databases

Lucas Alland*
Swarthmore College
Swarthmore, PA, United States

Attila Sali
HUN-REN Alfréd Rényi Institute
Budapest, Hungary

Nicole Wu*
Harvey Mudd College
Claremont, CA, United States

Abstract

Let $R = \{A_1, A_2, \ldots, A_n\}$ be a relational database schema and r be a relation over R. For $X, Y \subset R$ with $X \cap Y = \emptyset$, r is said to satisfy independence atom $X \perp Y$, if the projection $r(XY)$ of R to $X \cup Y$ is the Cartesian product of the projections $r(X)$ and $r(Y)$, i.e $r(XY) = r(X) \times r(Y)$. Implication of independence atoms is defined naturally and a collection of independence atoms is non-redundant, if none of its members is impled by the remaining ones. In the present paper the maximum possible size of non-redundant system of independence atoms is investigated.

Keywords

relational databases, independence atoms, non-redundant system

1 Introduction

In this paper, we investigate an efficient subclass of embedded multivalued data dependencies which are called – in accordance with [1] – independence atoms.

Definition 1.1. A relation r satisfies the independence atom $X \perp Y$ between two disjoint sets X and Y of attributes, if for all tuples $t_1, t_2 \in r$ there is some tuple $t \in r$ which matches the values of t_1 on all attributes in X and matches the values of t_2 on all attributes in Y.

In other words, in relations that satisfy $X \perp Y$, the occurrence of X-values is independent of the occurrence of Y-values.

If Σ is a collection of independence atoms and σ is an independence atom, then Σ implies σ, in notation $\Sigma \models \sigma$, if any database relation r that satisfies every atom in Σ also satisfies σ. The implication problem was axiomatized by Kontinen , Link and Väänänen [2]. The following four rules are sound and complete system for the implication of independence atoms.

(1) $X \perp \emptyset$ (trivial independence, $\mathcal{T}$).
(2) $X \perp Y \Rightarrow Y \perp X$ (symmetry, $\mathcal{S}$).
(3) $X \perp Y \cup Z \Rightarrow X \perp Y$ (decomposition, $\mathcal{D}$).
(4) $X \perp Y \wedge X \cup Y \perp Z \Rightarrow X \perp Y \cup Z$ (exchange, $\mathcal{E}$).

That is, $\Sigma \models \sigma$ iff σ has a finite derivation from Σ using the rules above.

A set of independence atoms, Σ is **non-redundant** if, for each atom $\sigma \in \Sigma$, $\Sigma \setminus \{\sigma\} \not\models \sigma$—no atom can be derived from the other atoms.

The following interesting problem was asked by Sebastian Link [3]. Let $f(n)$ denote the largest size of a non-redundant collection of independent atoms over a schema of n attributes. Determine or give good bounds for $f(n)$.

For the sake of convenience, we identify the schema of n attributes $\{A_1, A_2, \ldots, A_n\}$ with the set $[n]$ of the first n positive integers. It was proven in [2] that every system Σ of independence atoms has an Armstrong database, that is a database that satisfies an independence atom σ iff $\Sigma \models \sigma$. This implies that every non-redundant system of independence atoms actually occurs as a system satisfied by a particular database, so for us it is enough to consider implications at the schema level.

2 Lower Bounds

One may check easily that $f(1) = 0$ and $f(2) = 1$. An immediate lower bound for $f(n)$ follows from the observation that none of the derivation rules introduces new attributes. That is, $X \perp Y$ cannot be derived from a set $\Sigma = \{X_i \perp Y_i : i \in I\}$ if $X_i \cup Y_i \not\supseteq X \cup Y$ for all $i \in I$.

Proposition 2.1. *For $n \geq 3$ we have*

$$f(n) \geq \binom{n}{\lceil \frac{n}{2} \rceil}.$$

Proof. Let $\{X_i \cup Y_i : i = 1, 2, \ldots, \binom{n}{\lceil \frac{n}{2} \rceil}\}$ be list of nontrivial partitions of the $\lceil \frac{n}{2} \rceil$-element subsets of $[n]$. The system $\Sigma = \{X_i \perp Y_i : i = 1, 2, \ldots, \binom{n}{\lceil \frac{n}{2} \rceil}\}$ is clearly non-redundant, as $X_i \cup Y_i \not\subseteq X_j \cup Y_j$ for $i \neq j$. □

It is interesting to observe that this simple lower bound is sharp for small n.

Proposition 2.2. *$f(3) = 3$ and $f(4) = 6$.*

The proof of the first statement is easy, the latter one needs detailed case-by-case analysis.

We have improved on the simple lower bound above by using a recursive construction.

Theorem 2.3. *For $n \geq 2$ we have $f(n) \geq f(n-1) + \binom{n-1}{\lceil \frac{n-1}{2} \rceil}$ implying*

$$f(n) \geq f(1) + \sum_{i=2}^{n} \binom{i-1}{\lceil \frac{i-1}{2} \rceil} = \sum_{i=1}^{n-1} \binom{i}{\lceil \frac{i}{2} \rceil}.$$

Proof. The proof is based on the following observation of [2]. Let Σ be a collection of independence atoms over $[n]$ and $X \perp Y$ be an independence atom. Let $\Sigma' = \Sigma[XY] = \{V \cap (X \cup Y) \perp W \cap (X \cup Y) : V \perp W \in \Sigma\}$. If there is no non-trivial atom $U \perp V \in \Sigma'$ where $U \cup V = X \cup Y$, then $\Sigma \not\models X \perp Y$.

Now assume that Σ_{n-1} is a non-redundant system of independence atoms of size $f(n-1)$ over $[n-1]$. Define $\Sigma_n = \Sigma_{n-1} \cup \{\{n\} \perp X : X \subset [n-1] \text{ with } |X| = \lceil \frac{n-1}{2} \rceil\}$. We claim that Σ_n is non-redundant. Indeed, none of the atoms of the form $\{n\} \perp X$ can be derived from $\Sigma_n \setminus \{\{n\} \perp X\}$ as no other atom contains all attributes of $\{n\} \perp X$. On the other hand, the atoms added to Σ_{n-1} cannot be used in a derivation for an atom in Σ_{n-1} since their intersection with $[n-1]$ is a trivial atom of the form $\emptyset \perp X$. □

*This research was done under the auspicies of Research Opportunties course at Budapest Semesters in Mathematics

MATCOS-25, October 9th and 10th, 2025., Ljubljana, Slovenia

3 Upper bounds

A useful concept in obtaining upper bounds is the concept of *atom shapes*.

Definition 3.1. An atom $X \perp Y$ is defined as having the shape $(|X|, |Y|)$. Notice by symmetry ($\mathcal{S}$) that an (a, b) shape is equivalent to an (b, a) shape.

Non-trivial shapes on five attributes, with examples.

(1) $(1, 1)$: $1 \perp 2$
(2) $(1, 2)$: $1 \perp 2\ 3$
(3) $(2, 2)$: $1\ 2 \perp 3\ 4$
(4) $(1, 3)$: $1 \perp 1\ 2\ 3$
(5) $(1, 4)$: $1 \perp 2\ 3\ 4\ 5$
(6) $(2, 3)$: $1\ 2 \perp 3\ 4\ 5$

Proposition 3.2. *A maximum sized non-redundant system of independence atoms over 5 attributes cannot contain an atom of shape* $(1, 4)$.

Proof. Let $\alpha = 1 \perp 2\ 3\ 4\ 5$ and Σ be a non-redundant set on 5 attributes, containing α. We see that $\alpha \models 1 \perp Y$ for all $Y \subseteq \{2, 3, 4, 5\}$. We partition $\Sigma \setminus \{\alpha\}$ by the sets S and T where:

$$S = \{1 \cup X \perp Y \in \Sigma \mid X \neq \emptyset\}$$
$$T = \{U \perp V \in \Sigma \mid U \cup V \subseteq \{2, 3, 4, 5\}\}.$$

Now define

$$S' = \{X \perp Y \mid 1 \cup X \perp Y \in S\}.$$

We will show $\Sigma' = S' \cup T$ is non-redundant. Take any $\sigma' \in \Sigma'$ and consider two cases:

- $\sigma' = U \perp V \in T$ Every atom in S' is derivable from its corresponding atom in S. So if $\sigma' \in T$ were derivable from $\Sigma' \setminus \{\sigma'\}$ then the same sequence of implications would show $\Sigma \setminus \{\sigma'\} \models \sigma'$.
- $\sigma' = X \perp Y \in S'$ Let $\sigma = 1 \cup X \perp Y$. Now, we observe:

  $$\alpha \models 1 \perp X \cup Y \text{ by } \mathcal{D}$$
  $$1 \perp X \cup Y \wedge X \perp Y \models \sigma \text{ by } \mathcal{E}.$$

  So if $\Sigma' \setminus \{\sigma'\} \models \sigma'$ then with the above two implications we have $\Sigma \setminus \{\sigma\} \models \sigma$.

So we conclude Σ' is non-redundant. Since $f(4) = 6$, and Σ' is a set on four attributes, $|\Sigma'| \leq 6$. Then,

$$|\Sigma| = |\{\alpha\}| + |S| + |T| = 1 + |S'| + |T| = 1 + |\Sigma'| \leq 7.$$

□

We note that the previous argument generalizes to all n from the given $n = 5$ case. We can in general conclude that if Σ is a non-redundant set of maximum size on n attributes that contain a $(1, n-1)$ shape atom:

$$|\Sigma| \leq 1 + f(n-1).$$

Since our lower bound for $f(n)$ increases faster than 1 when $n \geq 3$, the maximum-order non-redundant attribute set will never contain a $(1, n-1)$ shape atom.

Proposition 3.3. *A maximum sized non-redundant set of independence atoms on 5 attributes cannot contain an atom of shape* $(2, 3)$.

Proof. Let Σ be non-redundant, suppose we have $K \subseteq \Sigma$ such that $K \models \alpha$ and then define

$$L = \{1 \perp Y \in \Sigma \setminus K\}$$
$$S = \{1 \cup X \perp Y \in \Sigma \setminus K \mid X \neq \emptyset\}$$
$$T = \{U \perp V \in \Sigma \setminus K \mid U \cup V \subseteq \{2, 3, 4, 5\}\}.$$

Then, we may partition Σ by

$$\Sigma = K \cup L \cup S \cup T.$$

Note, that $|L| \leq 1$ by $\mathcal{D}$. Now, recalling the proof that there are no $(1, 4)$ atoms, we can conclude

$$|S \cup T| \leq 6.$$

Indeed, since α is derived from K and $S \cup T \subseteq \Sigma \setminus K$, we may use α in the derivations from $S \cup T$. This implies that

$$|\Sigma| \leq |K| + 7.$$

Suppose $\beta = 1\ 2 \perp 3\ 4\ 5 \in \Sigma$, and Σ is non-redundant. Note that β implies α by exchange ($\mathcal{E}$) if $\gamma = 1 \perp 2$ is derivable from $\Sigma \setminus \{\beta\}$. Consider two cases:

(1) If $\Sigma \not\models \gamma$ then the only allowed atoms in $\Sigma \setminus \{\beta\}$ are from the sets:

$$P = \{U \perp V \mid U \cup V \subseteq \{3, 4, 5\}, U \neq \emptyset, V \neq \emptyset\}$$
$$Q = \{2 \cup U \perp V \mid U \cup V \subseteq \{3, 4, 5\}, U \neq \emptyset, V \neq \emptyset\}$$
$$R = \{1 \cup U \perp V \mid U \cup V \subseteq \{3, 4, 5\}, U \neq \emptyset, V \neq \emptyset\}$$
$$W = \{1\ 2 \cup U \perp V \mid U \cup V \subseteq \{3, 4, 5\}, U \neq \emptyset, V \neq \emptyset\}.$$

In particular, $L = \emptyset$, since $Y \subseteq \{3, 4, 5\}$ would hold, thus $\beta \models 1 \perp Y$ by $\mathcal{D}$.
Observe that for $U \cup V \subseteq \{3, 4, 5\}, U \neq \emptyset, V \neq \emptyset$ $\{U \perp V, \beta\} \models 1\ 2 \cup U \perp V$ by $\mathcal{D}, \mathcal{E}$. So for a given pair $U \cup V \subseteq \{3, 4, 5\}, U \neq \emptyset, V \neq \emptyset$ at most one atom from the sets P, Q, R, W can be in Σ. That is, if $Q' = \{U \perp V : 2 \cup U \perp V \in Q\}$, $R' = \{U \perp V : 1 \cup U \perp V \in R\}$ and $W' = \{U \perp V : 1\ 2 \cup U \perp V \in W\}$, then these sets are pairwise disjoint and also $P \cup Q' \cup R' \cup W'$ is a non-redundant system of independence atoms over the attribute set $\{3, 4, 5\}$. Thus, $|P \cup Q \cup R \cup W| = |P \cup Q' \cup R' \cup W'| \leq 3$ implying $|\Sigma| \leq 4 < 12$.

(2) If $\Sigma \models \gamma$ then we have an atom from the set

$$S = \{1 \cup X \perp 2 \cup Y \mid X \cup Y \subseteq \{3, 4, 5\}\}$$

With β, one such atom will derive α. Then, $|K| \leq 2$ which forces $|\Sigma| \leq 9 < 12$.

□

Proposition 3.4. $f(5) \leq 30$

Proof. We use the above two facts to find an upper bound on $f(5)$. Let $|\Sigma| = 31 = 5f(4) + 1$ on 5 attributes. Then, for Σ to be non-redundant:

(1) Each attribute from $[5]$ may miss at most $f(4)$ atoms, since those atoms must form a non-redundant system on the remaining four attributes. Thus, each attribute must appear in at least $|\Sigma| - f(4) = 31 - 6 = 25$ times;
(2) Each atom can contain at most 4 attributes.

Comparing the possible number of appearances and the necessary number of appearances over all attributes would imply the following inequality:

$$125 = 5(24 + 1) \leq 4|\Sigma| = 124$$

which clearly does not hold. So for any non-redundant atom-set Σ, $|\Sigma| \leq 30$. Thus, $f(5) \leq 30$. □

We can give a general upper bound.

Proposition 3.5. $f(n) < (3-\varepsilon)^n$

Proof. Consider the set of all independence atoms to be a partially ordered set according to the rule $Z \perp W \preccurlyeq X \perp Y$ if $X \perp Y \vDash Z \perp W$ by decomposition ($\mathcal{D}$). This is a graded poset by the rank function

$$\rho(X \perp Y) = |X \cup Y|.$$

For example, the elements of level 1 are the trivial atoms with the empty set on one side and one attribute on the other.

A rank k element of this poset is covered by $2(n-k)$ elements of rank $k+1$, since we have two choices to which side of the atom the new attribute is added. On the other hand, a rank $k+1$ element covers $k+1$ rank k elements. This implies that this graded poset has the *Sperner property* — no antichain is larger than the largest rank level. Since every rank-level is itself an antichain, the largest rank-level will be a maximum-sized antichain.

Thus, since any non-redundant set must be the subset of some antichain, $f(n)$ is bounded by the size of the largest rank-level:

$$f(n) \leq \max\{L(k) \mid 1 \leq k \leq n\}$$

where $L(k)$ denotes the size of rank-level k. We consider how $L(k)$ relates to $L(k+1)$.

The covering numbers determined above give us the following relation:

$$2 \cdot (n-k) \cdot L(k) = (k+1) \cdot L(k+1).$$

Using the above iteratively and

$$L(1) = |\{k \perp \emptyset, \emptyset \perp k \mid k \in [n]\}| = 2n$$

we obtain

$$L(k) = n \cdot \frac{2(n-1)}{1+1} \cdot \ldots \cdot \frac{2(n-k+1)}{k-1+1} = 2^k \binom{n}{k}.$$

Then, the maximum of $L(k)$ is reached (non-uniquely) when $k = \lfloor \frac{2n}{3} \rfloor$.

Hence,

$$f(n) \leq L(\lfloor \frac{2n}{3} \rfloor) = 2^{\lfloor \frac{2n}{3} \rfloor} \binom{n}{\lfloor \frac{2n}{3} \rfloor} < (3-\varepsilon)^n$$

where the final bound can be obtained by applying some asymptotics on the binomial coefficient. □

4 Conclusions

We have studied non-redundant systems of independence atoms of relational database schemata. We defined $f(n)$ to be the largest possible size of such a system over a schema of n attributes. $f(n)$ was determined for small n and we gave general lower and upper bounds. We conjecture that

Conjecture 4.1. $f(5) = 12$.

Also, we ask whether the general lower bound construction is optimal. We believe so, but we do not dare to put it as a conjecture.

References

[1] Erich Grädel and Jouko Väänänen. 2013. Dependence and independence. *Studia Logica*, 101, 399–410. https://www.jstor.org/stable/23488329.

[2] Juha Kontinen, Sebastian Link, and Jouko Väänänen. 2013. Independence in database relations. In *Logic, Language, Information, and Computation*. Leonid Libkin, Ulrich Kohlenbach, and Ruy de Queiroz, editors. Springer Berlin Heidelberg, Berlin, Heidelberg, 179–193. ISBN: 978-3-642-39992-3.

[3] Sebastian Link. 2024. Personal Communication. (2024).

Scrambler Automaton Block Cipher for IoT Devices

Pál Dömösi
University of Debrecen
4028 Debrecen, Kassai Road 26., Hungary
domosi@unideb.hu

Géza Horváth
University of Debrecen
4028 Debrecen, Kassai Road 26., Hungary
horvath.geza@inf.unideb.hu

ABSTRACT

In this article, we introduce a new block cipher based on finite automata. Its structure is simple, using few and inexpensive operations, making it particularly suitable for lightweight cryptographic applications.

KEYWORDS

finite automata, Internet of Things, lightweight cryptography, block cypher

1 INTRODUCTION

Communication has undergone significant changes in the 21st century. While communication initially took place between people, and then in many cases between computers by the end of the 20th century, in the 21st century countless small smart devices communicate with each other and their environment via the Internet. These devices and this changed environment are collectively referred to as the Internet of Things, or IoT for short. These changes have inevitably forced changes in secret communication and encryption. Since encrypted communication must be implemented using inexpensive and simple tools, procedures that use few simple operations, require little memory and storage space, and still provide fast and secure communication have come to the fore. These procedures are collectively referred to as lightweight cryptography.

A significant step towards lightweight cryptography based on automata theory was the stream cipher introduced by Pál Dömösi and Géza Horváth in 2017 [1]. This stream cipher was subjected to thorough testing, which confirmed that the system is resistant to side-channel attacks, a type of attack that plays a significant role in attacks against IoT devices [2]. The authors of paper [1] described a scrambler method in patent [5], demonstrating its use as a pseudo-random number generator in the articles [3] and [4]. This article describes the use of the scrambler method described in patent [5] as a block cipher. A well-known common weakness of symmetric encryption is that a suitable method (such as asymmetric encryption) is required to exchange (synchronize) the secret key without revealing confidential information. Therefore, the cipher presented in this paper does not provide a solution to this difficulty.

2 BASIC CONCEPTS

In this lecture we consider a novel type of block cipher based on abstract finite automata. This cipher consists of three parts. One of them is a counter which sends its current state as input to a so-called scrambler automaton, which also belongs to the cipher. The second part is a feeder, which passes the plaintext to the cipher block by block for encryption. The third part is a scrambler automaton which changes from the state received from the feeder to another state in response to an input signal received from the counter. The new state will be the next output block of the cipher, i.e. the next ciphertext block. Decryption is essentially done in the same way, but the role of the scrambler automaton is taken over by a so-called inverse scrambler automaton. It also changes from the state, the next ciphertext block, received from the feeder to another state in response to an input signal received from the counter. This response is nothing but the next block of the plaintext (i.e. the decrypted secret text block). The counter starts its operation with the same secret initial state as during encryption. After then sends its current state as input to the inverse scrambler automaton of the cipher.

The term "counter" is used to refer to a method and apparatus that is presumed to operate according to a discrete time scale such that at the start of its operation it is in a fixed state s_0, and in every subsequent time instant t its state is an element s_t of the nonempty, finite state set S, where $s_t \in S$ denotes the state of the counter at the time instant immediately preceding the time instant t, while $f : S \to S$ is a function adapted to map the nonempty, finite set S to itself in a bijective manner. The triplet $\mathcal{S} = (S, s_0, f)$ will herein after also be referred to as the base structure of the counter, set S will be called the state set of the counter, state $s_0 \in S$ the core of the counter, and function $f : S \to S$ the state transition function of the counter.

In the following, it is assumed that the state transition functions f applied in this application are very simple, preferably $S = \{0, 1, \ldots, 2^{128} - 1\}$, and for each $k \in S, f(k) = k + 1$, if $k + 1 < 2^{128} - 1$, and $f(k) = 0$, if $k + 1 = 2^{128} - 1$.)

3 TRANSPOSITION-CONTROLLED AUTOMATON

For every $m = 1, 2, \ldots$, define the permutations $P_1, P_2, \ldots, P_m$ over the set $\{1, \ldots, 2^m\}$ in the following way.

Let n be a fixed positive integer power of 2, and let us define the following permutations that are specified as a product of transpositions (for example, for a permutation P the transposition $(9, 13)$ " which thus denotes such a pair " means that $P(9) = 13$ and $P(13) = 9$). For specifying these permutations, let us consider the following algorithm for the vector $(1, \ldots, n)$:

If $n = 2$, then let $P_1 = (1, 2)$, and we are ready.

Else, let us consider the vectors $(1, \ldots, n/2)$ and $(n/2{+}1, \ldots, n)$, and let us generate the permutation P_1 such that P_1 is a product of such transpositions where for each $k \in \{1, \ldots, n/2\}$ the first component of the k-th factor of this transposition-product is the k-th component of the vector $(1, \ldots, n/2)$, while the second component thereof is the k-th component of the vector $(n/2 + 1, \ldots, n)$. Taken to an expression: $P_1 = (1, n/2 + 1)(2, n/2 + 2) \ldots, (n/2, n)$.

If $n = 4$, then let $P_2 = (1, 2)(3, 4)$, and we are ready.

Else, let us carry out the above process separately for the vector $(1, \ldots, n/2)$ and for the vector $(n/2 + 1, \ldots, n)$. The product of the two permutations $(1, n/4 + 1)(2, n/4 + 2) \cdots (n/4, 2n/4)$ and $(2n/4 + 1, 3n/4 + 1)(2n/4 + 2, 3n/4 + 2) \cdots (3n/4, 4n/4)$ thus

Information Society 2024, 7–11 October 2024, Ljubljana, Slovenia

obtained will be the permutation $P_2 = (1, n/4+1)(2, n/4+2)\cdots(n/4, 2n/4)\ (2n/4+1, 3n/4+1)(2n/4+2, 3n/4+2)\cdots(3n/4, 4n/4)$.

Else, like with the above, let us carry out the process separately for the vectors $(1,\ldots,n/4), (n/4+1,\ldots,2n/4), (2n/4+1,\ldots,3n/4), (3n/4+1,\ldots,4n/4)$. The product of the four permutations $(1, n/8+1)(2, n/8+2)\cdots(n/8, 2n/8)$, $(2n/8+1, 3n/8+1)(2n/8+2, 3n/8+2)\cdots(3n/8, 4n/8)$, $(4n/8+1, 5n/8+1)(4n/8+2, 5n/8+2)\cdots(5n/8, 6n/8)$, $(6n/8+1, 7n/8+1)(6n/8+2, 7n/8+2)\cdots(7n/8, 8n/8)$ thereby obtained will be the permutation $P_3 = (1, n/8+1)(2, n/8+2)\cdots(n/8, 2n/8)(2n/8+1, 3n/8+1)\cdots(5n/8, 6n/8)(6n/8+1, 7n/8+1)(6n/8+2, 7n/8+2)\cdots(7n/8, 8n/8)$.

If $n = 16$, then let $P_4 = (1,2)(3,4)\ldots,(15,16)$.

Else, like with the above, let us carry out the process separately for the vectors $(1,\cdots,n/8), (n/8+1,\cdots,2n/8), (2n/8+1,\cdots,3n/8), (3n/8+1,\cdots,4n/8), (4n/8+1,\cdots,5n/8), (5n/8+1,\cdots,6n/8), (6n/8+1,\cdots,7n/8), (7n/8+1,\cdots,8n/8)$.

The product of the eight permutations $(1, n/16+1)(2, n/16+2)\cdots(n/16, 2n/16)$, $(2n/16+2, 3n/16+2)\cdots(3n/16, 4n/16)$, $(4n/16+1, 5n/16+1)\ (4n/16+2, 5n/16+2)\cdots(5n/16, 6n/16)$, $(6n/16+1, 7n/16+1)\ (6n/16+2, 7n/16+2)\cdots(7n/16, 8n/16)$, $(8n/16+1, 9n/16+1)(8n/16+2, 9n/16+2)\cdots(9n/16, 10n/16)$, $(12n/16+1, 13n/16+1)(12n/16+2, 13n/16+2)\cdots(13n/16, 14n/16)$, $(14n/16+1, 15n/16+1)(14n/16+2, 15n/16+2)\cdots(15n/16, 16n/16)$ thus obtained will be the permutation $P_4 = (1, n/16+1)(2, n/16+2)\cdots(n/16, 2n/16)(2n/16+1, 3n/16+1)(2n/16+2, 3n/16+2)\cdots(3n/16, 4n/16)(4n/16+1, 5n/16+1)(4n/16+2, 5n/16+2)\cdots(5n/16, 6n/16)(6n/16+1, 7n/16+1)(6n/16+2, 7n/16+2)\cdots(7n/16, 8n/16)(8n/16+1, 9n/16+1)(8n/16+2, 9n/16+2)\cdots(9n/16, 10n/16)\ (10n/16+1, 11n/16+1)(10n/16+2, 11n/16+2)\cdots(11n/16, 12n/16)\ (12n/16+1, 13n/16+1)(12n/16+2, 13n/16+2)\cdots(13n/16, 14n/16)\ (14n/16+1, 15n/16+1)(14n/16+2, 15n/16+2)\cdots(15n/16, 16n/16)$.

If $n = 32$, the let $P_5 = (1,2)(3,4)\cdots(31,32)$ and we are ready.

Else, by continuing the process in an analogous manner, for every $n > 32$,where n is a power of two, we get $P_1 = (1, n/2+1)(2, n/2-12)\cdots(n/2, n)$, $P_2 = (1, n/4+1)(2, n/4+2)\cdots(n/4, 2n/4)(2n/4+1, 3n/4+1)(2n/4+2, 3n/4+2)\cdots(3n/4, 4n/4)$, $P_3 = (1, n/8+1)(2, n/8+2)\cdots(n/8, 2n/8)(2n/8+1, 3n/8+1(2n/8+2, 3n/8+2))\cdots(3n/8, 4n/8)(4n/8+1, 5n/8+1)(4n/8+2, 5n/8+2)\cdots(5n/8, 6n/8)\ (6n/8+1, 7n/8+1)(6n/8+2, 7n/8+2)\cdots(7n/8, 8n/8)$, $P_4 = (1, n/16+1)(2, n/16+2)\cdots(n/16, 2n/16)(2n/16+1, 3n/16+1)\ (2n/16+2, 3n/16+2)\cdots(3n/16, 4n/16)(4n/16+1, 5n/16+1)\ (4n/16+2, 5n/16+2)\cdots(5n/16, 6n/16)(6n/16+1, 7n/16+1)\ (6n/16+2, 7n/16+2)\cdots(7n/16, 8n/16)\ (8n/16+1, 9n/16+1)(8n/16+2, 9n/16+2)\cdots(9n/16, 10n/16)(10n/16+1, 11n/16+1)(10n/16+2, 11n/16+2)\cdots(11n/16, 12n/16)(12n/16+1, 13n/16+1)(12n/16+2, 13n/16+2)\cdots(13n/16, 14n/16)(14n/16+1, 15n/16+1)(14n/16+2, 15n/16+2)\cdots(15n/16, 16n/16), \cdots P_{log_2 n-1} = (1,3)(2,4)(5,7)(6,8)\cdots(n-3, n-1)(n-2, n)$, $P_{log_2 n} = (1,2)(3,4)\cdots(n-1, n)$.

For example, if $n = 16$, then

$P_1 = (1,9)(2,10)(3,11)(4,12)(5,13)(6,14)(7,15)(8,16)$,

$P_2 = (1,5)(2,6)(3,7)(4,8)(9,13)(10,14)(11,15)(12,16)$,

$P_3 = (1,3)(2,4)(5,7)(6,8)(9,11)(10,12)(13,15)(14,16)$,

$P_4 = (1,2)(3,4)(5,6)(7,8)(9,10)(11,12)(13,14)(15,16)$.

Let us define the automata $\mathcal{B}_i = (A^n, X^n, \delta_{B,i})$ for each $i \in \{1,\ldots,log_2 n\}$ such that for any (the definition formula below specifies what was referred to above as the calculation "turns": based on the index j a "with comma" or a "without comma" character component is used in the formula; the formula distinguishes the permutations according to indices $a_1,\ldots,a_n \in A, x_1,\ldots,x_n \in X$, $\delta_{B,i}((a_1,\ldots,a_n),(x_1,\ldots,x_n)) = (a"_1,\ldots,a"_n)$, where

$a"_1 = \delta(a_1, \delta(a_{P_i(1)}, x_{P_i(1)}))$,

$\vdots$

$a"_j = \delta(a_j, \delta(a_{P_i(j)}, x_{P_i(j)}))$, if $j < P_i(j)$, and $a"_j = \delta(a_j, \delta(a"_{P_i(j)}, x_{P_i(j)}))$, if $j \geq P_i(j) (j \in 1,\ldots,n)$,

$\vdots$

$a"_n = \delta(a_n, \delta(a"_{P_i(n)}, x_{P_i(n)}))$.

For example, if $i = 1$ and $P_1 = (1,9)(2,10)(3,11)(4,12)(5,13)(6,14)(7,15)(8,16)$, then

$a"_1 = \delta(a_1, \delta(a_9, x_9))$, because $P_1(1) = 9$ and $1 < P_1(1)(= 9)$,

$a"_2 = \delta(a_2, \delta(a_{10}, x_{10}))$, because $P_1(2) = 10$ and $2 < P_1(2)(= 10)$,

$\vdots$

$a"_8 = \delta(a_8, \delta(a_{16}, x_{16}))$, because $P_1(8) = 16$ and $8 < P_1(8)(= 16)$,

$a"_9 = \delta(a_9, \delta(a"_1, x_1))$, because $P_1(9) = 16$ and $9 > P_1(9)(= 16)$,

$a"_{10} = \delta(a_{10}, \delta(a"_2, x_2))$, because $P_1(10) = 2$ and $10 > P_1(10) = 2$,

$\vdots$

$a"_{16} = \delta(a_{16}, \delta(a"_8, x_8))$, because $P_1(16) = 8$ and $16 > P_1(16)(= 8)$.

For the example included below, let us define the above such that the automaton $\mathcal{B} = (A^n, X^{nlog_2 n}, \delta_B)$ is defined such that for any $a_1,\ldots,a_n \in A, (x_1,\ldots,x_{nlog_2 n}) \in X^{nlog_2 n}$, the transition $\delta_B((a_1,\ldots,a_n),(x_1,\ldots,x_{nlog_2 n}))$ is generated by first generating the state vector that can be obtained by applying the transition function $\delta_{B,1}$ for the vector $(a_1,\ldots,a_n)$ as a state, and for the vector $(x_1,\ldots,x_n)$ as an input signal. Taken to an expression: first the transitions $\delta_{B,1}((a_1,\ldots,a_n),(x_1,\ldots,x_n))$ are generated. Thereafter, the transition function $\delta_{B,2}$ is applied for the result of this transition and the vector $(x_{n+1},\ldots,x_{2n})$ as an input signal. Taken to an expression: the transition $\delta_{B,3}(\delta_{B,2}(\delta_{B,1}((a_1,\ldots,a_n),(x_1,\ldots,x_n)),(x_{n+1},\ldots,x_{2n})),(x_{2n+1},\ldots,x_{3n}))$ is generated (therefore, in the permutation approach the steps of this process have to be implemented applying the above-described permutations). This process is carried on in $log_2 n$ steps, wherein, in the last step, the transition function $\delta_{B,log_2 n}$ is applied for the state vector obtained, and for the vector $(x_{n(log_2 n-1)},\ldots,x_{blog_2 n})$ as an input signal. Taken to an expression: the transition $\delta_{B,log_2 n}(\delta_{B,log_2 n-1}(\cdots\delta_{B,2}(\delta_{B,1}((a_1,\ldots,a_n),(x_1,\ldots,x_n)),(x_{n+1},\ldots,x_{2n}))\ldots,x_{n(logn-1)+1},\ldots,x_{nlog_2 n}))$ is generated. The automaton $\mathcal{B}$ defined in such a manner is a scrambler automaton having $log_2 n$ components that is determined by the automaton $\mathcal{A}$, where the automaton $\mathcal{A}$ is the base automaton of the automaton $\mathcal{B}$. By our definition, $\mathcal{B}$ is an automaton contolled by transpositions of its state components. So, for short, $\mathcal{B}$ will be called *transposition-controlled automaton.* It is also assumed that the transition matrix of the base automaton forms a Latin square.

4 THE NOVEL BLOCK CIPHER

The scrambler unit $\mathcal{B}$ of the cipher is called *transposition-controlled automaton,* using which, receiving a character string w having a length of power of two n yields a string $g(w)$ (where $g : \{0,1\}^n \to \{0,1\}^n$ is a function that bijectively maps the set of all possible strings of length n onto itself).

The function g will hereinafter be referred to as a scrambler function. A matrix of which each row and each column is a permutation of the elements of H is a Latin square over the set H. An automaton without output is an algebraic structure consisting of two non-empty sets, namely, the state set and the

input signal set, and a function named transition function that maps the Cartesian product of the state set and the input signal set onto the state set. Therefore, an automaton without outputs is usually described in the form $\mathcal{A} = (A, X, \delta)$, where A is the state set, X is the input signal set, and $\delta : A \times X \rightarrow A$ is the transition function (which assigns a state to each pair having a state as its first element and an input signal as its second element). If for every triplet $a, b \in A$, $x \in X$, $a \neq b$ implies $\delta(a, x) \neq \delta(b, x)$ then $\mathcal{A}$ is called a *permutation automaton*. Obviously, if the transition matrix of $\mathcal{A}$ forms a Latin square then $\mathcal{A}$ is a pemutation automaton.

Next we show the following statement without proof. (See Appendix for a proof.)

Theorem 1. *A transposition-controlled automaton is a permutation automaton if and only if its base automaton is also a permutation automaton.*

We note that, as a consequence of our proposition, plaintext cannot always be decrypted from ciphertext even in the possession of the secret keys, if the basic automaton of the automaton is not a permutation automaton. Therefore, the cipher is useless if the basic automaton is not a permutation automaton.

Finally, the randomness of the secret text is helped if the base automaton is not only a permutation automaton, but also its transition matrix forms a Latin square. Thus we suppose this property of the discussed cipher.

5 EXAMPLE

The encoding and decoding procedure will be presented by the next toy example (where $n = 4$ only).

5.1 Encryption

Let us consider the following automaton $\mathcal{A}$ with four states and four input signals (its transition matrix is shown in Table 1 below), which automaton will be called the base automaton of the exemplary scrambler automaton (in the case of encryption, the base automaton can be called a "key automaton"):

δ	**state 0**	**state 1**	**state 2**	**state 3**
input 0:	1	2	3	0
input 1:	3	0	1	2
input 2:	2	3	0	1
input 3:	0	1	2	3

Table 1

In the 0-th row of the transition matrix specified in Table 1, the states are listed, with the 0-th column thereof containing the possible input signals. The condition that the state set and input set of the automaton are identical is fulfilled also in this example; however, in certain embodiments the state and input sets of the automaton may be different. The above transition matrix constitutes a *Latin square*. The state set of the automaton is $\{0, 1, 2, 3\}$, which is identical to the input signal set of the automaton, and to the character set of both the plaintext and the ciphertext. The scrambling and the descrambling operations will now be illustrated applying this example. Let us consider the hexadecimal ASCII code of the word "OK", $4F4B$ (the non-encrypted data correspond to the word "OK"). Converting this hexadecimal value, $4F4B$, into the quaternary number system, the character sequence 10331023 is obtained. This character sequence is the text to be scrambled. Let us assume that the block length is eight, i.e., the text to be scrambled constitutes a single block. Let us assume that we have only one permutation of the form $P_1 = (1, 2)$, that is, $n = 2$. Let the message to be encrypted be the encoded form of the word "OK", i.e., 10331023. First, the first (two-element) block of the message 10331023, i.e., "10" is encrypted. Let us assume that the counter core is $s_0 = 00$, while the state of the counter is calculated applying the formula $s_t = s_{t-1} + 1 (mod 16)$ (that is, when it attains 16 in the quaternary number system, it restarts), and the counter is adapted for passing on its states as two-digit quaternary numbers. In this case, with $a_1a_2 = 10, x_1x_2 = 01$ $a"_1 = \delta(a_1, \delta(a_2, x_2)) = \delta(1, \delta(0, 1)) = \delta(1, 3) = 1$, $a"_2 = \delta(a_2, \delta(a"_1, x_1)) = \delta(0, \delta(1, 0)) = \delta(0, 2) = 2$, and, denoting $a"_1$ with a_1, and $a"_2$ with a_2, considering the next counter state $x_1x_2 = 02$ $a"_1 = \delta(a_1, \delta(a_2, x_1)) = \delta(1, \delta(2, 0)) = \delta(1, 3) = 1$, $a"_2 = \delta(a_2, \delta(a"_1, x_2)) = \delta(2, \delta(1, 2)) = \delta(2, 3) = 2$, that is, the encrypted block corresponding to the first plaintext block, i.e., 10, is 12. The first of the two steps is carried out according to the permutations, with the index of the component xi being identical to the state component applied for the second time (i.e., not with "itself without comma"). The permutation, however, contains only one inversion, so it ends with the first combinations. The second "twist" is already a chain embodiment, wherein the index of the input signal character component x_i corresponds to the index to be carried over, i.e., to the "itself without comma", for which the "itself with comma" is calculated. The second block of the plaintext is $a_1a_2 = 33$, while the subsequent state of the counter is $x_1x_2 = 03$.Then $a"_1 = \delta(a_1, \delta(a_2, x_2)) = \delta(3, \delta(3, 3)) = \delta(3, 3) = 3$, $a"_2 = \delta(a_2, \delta(a"_1, x_1)) = \delta(3, \delta(3, 0)) = \delta(3, 0) = 0$, and, denoting $a"_1$ with a_1, and $a"_2$ with a_2, and considering the next counter state $x_1x_2 = 10$ (in the quaternary number system, $03 + 1 = 10$) $a"_1 = \delta(a_1, \delta(a_2, x_1)) = \delta(3, \delta(0, 1)) = \delta(3, 3) = 3$, $a"_2 = \delta(a_2, \delta(a"_1, x_2)) = \delta(0, \delta(3, 0)) = \delta(0, 0) = 1$, that is, the encrypted block of the second plaintext block 33 is 31. The third block of the plaintext is $a_1a_2 = 10$, while the subsequent state of the counter is $x_1x_2 = 11$. Then $a"_1 = \delta(a_1, \delta(a_2, x_2)) = \delta(1, \delta(0, 1)) = \delta(1, 3) = 1, a"_2 = \delta(a_2, \delta(a"_1, x_1)) = \delta(0, \delta(1, 1)) = \delta(0, 0) = 1$, and, denoting $a"_1$ with a1, and $a"2$ with a_2, considering the next counter state $x_1x_2 = 12$ $a"_1 = \delta(a_1, \delta(a_2, x_1)) = \delta(1, \delta(1, 1)) = \delta(1, 0) = 2, a"_2 = \delta(a_2, \delta(a"_1, x_2)) = \delta(1, \delta(2, 2)) = \delta(1, 0) = 2$, that is, the encrypted block of the third plaintext block 10 is 22. The fourth block of the plaintext is $a_1a_2 = 23$, while the subsequent state of the counter is $x_1x_2 = 13$. Then $a"_1 = \delta(a_1, \delta(a_2, x_2)) = \delta(2, \delta(3, 3)) = \delta(2, 3) = 2$, $a"_2 = \delta(a_2, \delta(a"_1, x_1)) = \delta(3, \delta(2, 1)) = \delta(3, 1) = 2$, and, denoting $a"_1$ with a_1, and $a"_2$ with a_2, and considering the next counter state $x_1x_2 = 20$ (in the quaternary number system, $13 + 1 = 20$) $a"_1 = \delta(a_1, \delta(a_2, x_1)) = \delta(2, \delta(2, 2)) = \delta(2, 0) = 3$, $a"_2 = \delta(a_2, \delta(a"_1, x_2)) = \delta(2, \delta(3, 0)) = \delta(2, 0) = 3$, that is, the encrypted block of the fourth plaintext block 23 is 33. The ciphertext is therefore 12312233. As it is shown by this simple example, in this case the basic block size is two (the initial and the output character block, i.e., generally speaking, the first and the second character block " applied in a separate respective step " have the same length, as well as the applied input signal block; two characters already constitute a block), the character sequence to be encrypted being processed applying a step length corresponding to this basic block size; in this example, the character sequence to be encrypted consists of eight characters, so the encryption process includes four main steps. As it is illustrated above, a certain type of a "double scrambling (mixing)" is applied in each encryption

step, i.e., the second character block obtained in a given "turn" is recycled into the role of the first character block.

5.2 Decryption

Now, let us consider the process of decryption (recovery). The ciphertext to be deciphered is therefore the following: 12312233. The inverse permutation automaton of the base automaton applied for scrambling (which base automaton is a permutation automaton) is the following (the corresponding transition matrix is illustrated in Table 5; for the process of generating the transition matrix of the inverse permutation automaton from the transition matrix see at Table 2):

δ^{-1}	**state 0**	**state 1**	**state 2**	**state 3**
input 0:	3	0	1	2
input 1:	1	2	3	0
input 2:	2	3	0	1
input 3:	0	1	2	3

Table 2

The first block of the ciphertext is 12, while the first and second generated counter states are 01 and 02. Then $a"_2 = \delta^{-1}(a_2, \delta(a_1, x_2)) = \delta^{-1}(2, \delta(1, 2)) = \delta^{-1}(2, 3) = 2, a"_1 = \delta^{-1}(a_1, \delta(a"_2, x_1)) = \delta^{-1}(1, \delta(2, 0)) = \delta^{-1}(1, 3) = 1$, and, denoting a'_1 with a_1, and $a"_2$ with a_2, considering the next counter state $x_1x_2 = 01$ $a"2 = \delta^{-1}(a2, \delta(a1, x1)) = \delta^{-1}(2, \delta(1, 0)) = \delta^{-1}(2, 2) = 0$, $a"_1 = \delta^{-1}(a_1, \delta(a"_2, x_2)) = \delta^{-1}(1, \delta(0, 1)) = \delta^1(1, 3) = 1$, that is, the deciphered block of the ciphertext block 12 is 10. The second block of the ciphertext is 31, while the subsequent two generated counter states are 03 and 10. Then $a"_2 = \delta^{-1}(a_2, \delta(a_1, x_2)) = \delta^{-1}(1, \delta(3, 0)) = \delta^{-1}(1, 0) = 0$, $a"_1 = \delta^{-1}(a_1, \delta(a"_2, x_1)) = \delta^{-1}(3, \delta(0, 1)) = \delta^{-1}(3, 3) = 3$, and, denoting $a"_1$ with a_1, and $a"_2$ with a_2, considering the next counter state $x_1x_2 = 03$ $a"_2 = \delta^{-1}(a_2, \delta(a_1, x_1)) = \delta^{-1}(0, \delta(3, 0)) = \delta^{-1}(0, 0) = 3, a"_1 = \delta^{-1}(a_1, \delta(a"_2, x_2)) = \delta{-}1(3, \delta(3, 3)) = \delta^{-1}(3, 3) = 3$, that is, the deciphered block of the second ciphertext block 23 is 33. The third block of the ciphertext is 22, while the subsequent two generated counter states are 11 and 12. Then $a"_2 = \delta^{-1}(a_2, \delta(a_1, x_2)) = \delta^{-1}(2, \delta(2, 2)) = \delta^{-1}(2, 0) = 1$, $a"_1 = \delta^{-1}(a_1, \delta(a"_2, x_1)) = \delta^{-1}(2, \delta(1, 2)) = \delta^{-1}(2, 0) = 1$, and, denoting $a"_1$ with a_1, and $a"_2$ with $a2$, considering the next counter state $x_1x_2 = 11$ $a"_2 = \delta^{-1}(a_2, \delta(a_1, x_1)) = \delta^{-1}(1, \delta(1, 1)) = \delta^{-1}(1, 0) = 0, a"_1 = \delta^{-1}(a_1, \delta(a"_2, x_2)) = \delta^{-1}(1, \delta(0, 1)) = \delta^{-1}(1, 3) = 1$, that is, the deciphered block of the third ciphertext block 12 is 10. The fourth block of the ciphertext is 33, while the subsequent two generated counter states are 13 and 20. Then $a"_2 = \delta^{-1}(a_2, \delta(a_1, x_2)) = \delta^{-1}(3, \delta(3, 0)) = \delta^{-1}(3, 0) = 2$, $a"_1 = \delta^{-1}(a_1, \delta(a"_2, x_1)) = \delta^{-1}(3, \delta(2, 2)) = \delta^{-1}(3, 0) = 2$, and, denoting $a"_1$ with a_1, and $a"_2$ with $a2$, considering the next counter state $x_1x_2 = 13$ $a"_2 = \delta^{-1}(a_2, \delta(a_1, x_1)) = \delta^{-1}(2, \delta(2, 1)) = \delta^{-1}(2, 1) = 3, a"_1 = \delta^{-1}(a_1, \delta(a"_2, x_2)) = \delta^{-1}(2, \delta(3, 3)) = \delta^{-1}(2, 3) = 2$, that is, the deciphered block of the fourth ciphertext block 22 is 23. Combining the deciphered blocks, the plaintext 10331023 is obtained. Converting this back into hexadecimal form, the ASCII-coded version of the text "OK" is obtained.

In the case of both the cryptographic apparatus for block-encrypting plaintext data and the cryptographic apparatus for decrypting block-encrypted ciphertext data according to the invention the scrambler/descrambler apparatus is a scrambler/descrambler automaton. The invention also relates to a counter-based full-cycle pseudorandom number generator method and apparatus, in which apparatus a counter passes on its state values to a scrambler automaton, with the string obtained as a result of the scrambling process is represented by the method and apparatus either as a single pseudorandom character string, or "dividing this character string into equal-length portions" as a sequence of pseudorandom character strings.

6 SUMMARY

The aim of this paper is to show a symmetric cipher based on novel technologies. Our solution is an innovation regarding both the idea and the technology, so a new technology based on a new idea is developed. The cipher we have developed can replace older, out-of-date technology, and due to its simplicity they can be used well in all cases where older, more complex systems cannot be used due to memory requirements, operation requirements, or complexity. Currently, the most common block ciphers use several different complex operations, can be attacked by side-channel attacks, whereas the block encryption structure we have developed is much simpler, making the implementation more transparent and resistant to side-channel attacks. We can state in case of our system that it provides high security with simple operations, so its integration into any application is easy.

REFERENCES

[1] Pál Dömösi and Géza Horváth. 2017. A Novel Stream Cipher Based on Deterministic Finite Automaton. *In: R., Freund; F, Mráz; D, Prusa (eds): Ninth Workshop on Non-Classical Models of Automata and Applications, (NCMA 2017), Short Papers.* Wien, Austria, Technical University of Vienna. pp. 11–16, 6 p.

[2] Pál Dömösi, Géza Horváth, Ferenc Tamás Molnár, Szabolcs Kovács and Adama Diene. 2021. A side-channel attack against an automata theory based stream cipher. *SURIKAISEKIKENKYUSHO KOKYUROKU / RIMS KOKYUROKU 2193.* pp. 64-72, 9 p.

[3] Pál Dömösi, József Gáll, Géza Horváth, Bertalan Borsos, Norbert Tihanyi and Yousef Al Hammadi. 2021. A Full Cycle Length Pseudorandom Number Generator Based on Compositions of Automata. *Informatica–Journal of Computing and Informatics 45 : 2.* pp. 179-189, 11 p.

[4] Pál Dömösi, Géza Horváth and Norbert Tihanyi. 2023. Simple chain automaton random number generator for IoT devices. *Acta Informatica 60 : 3.* pp. 317-329, 13 p.

[5] Pál Dömösi and Géza Horváth. 2025. Scrambler Apparatus and Method in Particular for Cryptographic Applications, and Descrambler Apparatus and Method Therefor. *USA Patent.* Patent No. US 12,328,384 B2, Filing Year: 2025, Filing Number: 17/908,301.

7 APPENDIX

Let n be an arbitrary positive integer and let $\mathcal{A}_i = (A, X_i, \delta_i), i = 1, \ldots, n$ be a finite sequence of automata with a common state set A. Define the automaton $\mathcal{B} = (A, X_1 \times X_2 \times \cdots \times X_n, \delta)$ such that for every $a \in A$, $(x_1, \ldots, x_n) \in X_1 \times X_2 \times \cdots \times X_n$, $\delta(a, (x_1, \ldots, x_n)) = \delta_n(\delta_{n-1}(\cdots \delta_1(a, x_1), x_2), \cdots, x_{n-1}), x_n)$. Then we say that $\mathcal{B}$ is the *temporal product* of $\mathcal{A}_1, \ldots, \mathcal{A}_n$. Then we also say that $\mathcal{A}_i = (A, X_i, \delta_i), i = 1, \ldots, n$ are *component-automata* of the temporal product $\mathcal{B}$.

Proposition 2. *A temporal product of automata is a permutation automaton if and only if each of its component automata is a permutation automaton.*

Now we are ready to prove our main statement.

Theorem 3. *A transposition-controlled automaton is a permutation automaton if and only if its base automaton is also a permutation automaton.*

Proof: Consider a transposition-controlled automaton $\mathcal{B} = (A^n, X^{nlog_2 n}, \delta_B)$ consisting of component-automata $\mathcal{B}_i = (A^n, X^n, \delta_{B,i}), i \in \{1, \ldots, log_2 n\}$. Obviously, then $\mathcal{B}$ is a temporal product of components $\mathcal{B}_i, i \in \{1, \ldots, log_2 n\}$. Hence, by Proposition 2, $\mathcal{B}$ is a permutation automaton, if and only if, each of its component-automata $\mathcal{B}_i, i \in \{1, \ldots, log_2 n\}$ are permutation automata.

First we assume that the base automaton is a permutation automaton.

Consider a positive integer $j \in \{1, \ldots, n\}$ and a pair $(a_1, \ldots, a_n), (b_1, \ldots, b_n) \in A^n$ with $(a_1, \ldots, a_n) \neq (b_1, \ldots, b_n)$. Then there exists a $j \in \{1, \ldots, n\}$ with $a_j \neq b_j$. Let $(x_1, \ldots, x_n) \in X^n$ be an arbitrary input letter of $\mathcal{B}_i$. Put $(a''_1, \ldots, a''_n) = \delta_j((a_1, \ldots, a_n), (x_1, \ldots, x_n))$ and $(b''_1, \ldots, b''_n) = \delta_j((b_1, \ldots, b_n), (x_1, \ldots, x_n))$.

Suppose $j < P_i(j)$. Then, by definition, $a"_j = \delta(a_j, \delta(a_{P_i(j)}, x_{P_i(j)}))$ and $b"_j = \delta(b_j, \delta(a_{P_i(j)}, x_{P_i(j)}))$. Therefore, by our assumptions, $a''_j \neq b''_j$. Therefore, $(a''_1, \ldots, a''_n) \neq (b''_1, \ldots, b''_n)$.

Suppose $j \geq P_i(j)$. Then there are two possibilities. First, $a"_{P_i(j)} \neq b"_{P_i(j)}$. In this case, $(a''_1, \ldots, a''_n) \neq (b''_1, \ldots, b''_n)$ again. Second, $a"_{P_i(j)} = b"_{P_i(j)}$. In this case, $a"_j = \delta(a_j, \delta(a"_{P_i(j)}, x_{P_i(j)}))$, and $b"_j = \delta(b_j, \delta(b"_{P_i(j)}, x_{P_i(j)})) = \delta(b_j, \delta(a"_{P_i(j)}, x_{P_i(j)}))$. Because the base automaton is a permutation automaton, we obtain $a''_j \neq b''_j$ which leads to $(a''_1, \ldots, a''_n) \neq (b''_1, \ldots, b''_n)$ again. Therefore, if the base automaton is a permutation automaton then the transposition controlled automaton is also a permutation automaton.

Next we assume that the base automaton is not a permutation automaton. Then there are $a, b \in A, x \in X$ with $a \neq b$ and $\delta(a, x) = \delta(b, x)$. Consider a pair $(a_1, \ldots, a_{j-1}, a_j, a_{j+1}, \ldots, a_n)$, $(a_1, \ldots, a_{j-1}, b_j, a_{j+1}, \ldots, a_n))$.

Assume $j < P_i(j)$. Then choosing $x_{P_i(j)} = x$, we obtain $a''_j = b''_j$. Finally, assume $j \geq P_i(j)$. Then because of $P_i(j) \neq j$ we have $a_{P(j)} = b_{P(j)}$ and thus $a''_{P(j)} = b''_{P(j)}$. But then $a"_j = \delta(a_j, \delta(a"_{P_i(j)}, x_{P_i(j)})) = b"_j = \delta(b_j, \delta(b"_{P_i(j)}, x_{P_i(j)})) = b''_j$ again.

Thus, if the base automaton is not a permutation automaton then the transposition controlled automaton is also not a permutation automaton. The proof is complete.

Automata for context-free trace languages and permutation languages (extended abstract)

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

ABSTRACT

Traces are sets of equivalent words (sequences of actions); they are efficient tools to describe parallel processes. A trace language is a language of traces usually used in linearized form, i.e., it is a standard formal language that contains all words representing the traces of the language. Trace languages can be described by a base language and with a(n in)dependency relation on the alphabet. Context-free trace languages are based on context-free languages. Permutation grammars are context-free grammars extended by permutation rules (type $AB \to BA$ productions). They generate permutation languages. In this paper an extended variant of the pushdown automata is shown that is able to characterize the permutation languages. We also show that every context-free trace language is a permutation language, and thus, all of these languages are also accepted by such automata.

Categories and Subject Descriptors

F.4.3 [**Formal Languages**]

General Terms

Theory, Automata, Languages, Parallel processes

Keywords

Permutation grammars, formal languages, traces and trace languages, modeling parallel processes, pushdown automata

1. INTRODUCTION

The Chomsky type grammars and the generated language families are very basic and important parts of theoretical computer science [5, 29]. There are various refinements of the hierarchy where some further classes are properly inserted: e.g., the class of union-free regular languages is a subregular class [1, 12], the class accepted by deterministic 2-head automata (2detLIN) is properly between the regular and linear classes [18, 25, 26], the class of metalinear languages [5] is properly between the linear and context-free classes and the class of permutation languages [9, 13, 14] is properly between the context-free and context-sensitive classes. In this paper, we consider the class of permutation languages (they are formally recalled in Section 2, from [14]).

One of the main motivations for studying permutation grammars/languages comes from computational linguistics, as there are various non-context-free structures that occur in natural languages, in general. The non-context-free languages of multiple agreements $\{a^n b^n c^n\}$, cross dependencies $\{a^n b^m c^n d^m\}$ and copy $\{ww | w \in \{a,b\}^*\}$ are structures that occur in some natural languages. There is a big gap between the efficiency of context-free and context-sensitive grammars. Although, the class of context-free languages has some nice computational properties, they are not enough to describe several phenomena of the world. On the other hand, the context-sensitive family is too large, and has many inconvenient properties (e.g., PSPACE-complete word problem), thus, generally, in applications it is not used. Several branches of extensions of context-free grammars were introduced by controlling the derivations in various ways [2]. In this paper, we do not use any additional control, we focus on permutation grammars that allow to use context-free productions and permutation (aka. interchange) rules (of type $AB \to BA$). These additional rules are monotone rules having exactly the same letters in both sides. Permutation languages were analyzed from computational linguistical point of view in [15, 19] showing, e.g., that each of the three above mentioned famous mildly context-sensitive languages can be obtained as an intersection of a permutation language with a regular language. The word problem was addressed in [4, 20, 21] proving that, in general, it is NP-complete. Hierarchy, based on the lengths of the permutation rules (e.g., $ABC \to CBA$ has a length 3) was established in [16] and an infinite hierarchy was presented in [8]. These grammars and languages are also connected to concurrency and theory of parallel processes, where the order of some processes can be interchanged (or may go in parallel). The relation semi-commutation is known as a mapping that allows to permute two consecutive letters of a word if their order was in the given one. A partial commutation allows to permute two consecutive letters independently of their original order [28]. They are intensively studied by the connection to Petri-nets. Traces and trajectories [10] give formal framework to describe these processes [3]. Commutative closure

allows all permutations. In Section 3, we formally recall and describe some of these concepts.

The structure of this paper is as follows. In the next section we recall formal definitions for permutation languages. In Section 3, we show that all context-free trace languages are permutation languages. Further, in Section 4, we present a new variant of pushdown automata that is able to recognize permutation languages. Because the lack of space, for some of our results, instead of their formal proofs only their proof ideas are shown.

2. PERMUTATION GRAMMARS

Here, we recall some important definitions. We assume that the reader already knows the basic concepts of formal languages, generative grammars and automata theory. For any non-explained concept the reader is referred, e.g., to [5, 29]. We denote the length of a word u by $|u|$ and the empty word by λ. However, in this paper, we do not care if λ is in a language or not, that is, we consider two languages (or formalisms defining the languages) equivalent if they may differ at most in the empty word.

A *generative grammar* is a tuple $G = (N, T, S, P)$ with non-terminal and terminal alphabets, N and T, respectively, start symbol $S \in N$ and the finite set P of productions (aka. derivation rules). Each production is of the form $u \to v$ with $u \in (N \cup T)^* N (N \cup T)^*$ and $v \in (N \cup T)^*$. A grammar is *context-free* if $u \in N$ for each of its productions. A grammar is monotone if $|u| \leq |v|$ for each productions. A language is context-free (context-sensitive) if it is generated by a context-free (monotone, resp.) grammar.
It was known already in the early 70's that every monotone grammar is equivalent to a grammar having only productions of the following types $AB \to AC, AB \to BA, A \to BC, A \to B$ and $A \to a$ (where $A, B, C \in N$ and $a \in T$). On the one hand, in 1974 Penttonen showed that one-side context-sensitivity is enough to obtain the whole context-sensitive language class [27], i.e., grammars with only rules of type $AB \to AC, A \to BC, A \to B, A \to a$ suffice. On the other hand, in Turing-machine simulations the rules of type $AB \to BA$ can frequently be used representing the movement of the head of the machine. Furthermore, the grammars having non-context-free rules only in the form $AB \to BA$ generate the class of permutation languages, a proper subclass of the context-sensitive class.
Now we define formally the grammar and language class we are working with.

Definition 1. *A grammar $G = (N, T, S, P)$ is a* permutation grammar *if P contains context-free rules and only special type of non-context-free rules, namely, interchange (aka. permutation) rules, which are in the form $AB \to BA$ $(A, B \in N)$. We denote the class of permutation languages, the languages that are generated in this way, by* **PERM**.

Example 1. *Let $G_p = (\{S, A, B, C\}, \{a, b, c\}, S, P_p)$ be a permutation grammar with $P_p = \{S \to ABC, S \to ASBC, AB \to BA, AC \to CA, BC \to CB, CB \to BC, A \to a, B \to b, C \to c\}$. The language L_p containing all words with the same number of a-s, b-s and c-s is generated. This is a non-context-free language, and its intersection with the regular language $a^*b^*c^*$ results in the language $\{a^n b^n c^n\}$.*

There is a *normal form* for these grammars [14]: Every permutation language can be generated by a grammar where every production is in one of the following forms: $AB \to BA, A \to BC, A \to B, A \to a$ where $A, B, C \in N$ and $a \in T$. It is still open whether the chain rules $A \to B$ can be eliminated. Note, that in fact, these productions are in Kuroda normal form of context-sensitive grammars, moreover each non-context-free production is of the form $AB \to BA$.

Definition 2. *Let a permutation grammar G be given generating L. The language L_b generated by the context-free grammar obtained from G by deleting its non-context-free productions is a* basis language *of L.*

The basis language is usually not uniquely defined for a permutation language, it depends on the used grammar. However, any basis language L_b is letter equivalent to L (in Parikh sense). This fact has the following two important consequences: For each basis language $L_b \subseteq L$. Further, all permutation languages are semi-linear (in Parikh sense). This latter fact can be used, e.g., to show that some context-sensitive languages (e.g., $\{a^p \mid p \text{ is a prime}\}$) are not permutation languages. Other such tools are the so-called interchange lemmas proven in [14, 16].

3. CONTEXT-FREE TRACE LANGUAGES

This section starts by describing some formal concepts describing/modeling parallel computations. The *commutative closure* of a language L is the set of all words having Parikh-vectors included in the Parikh-map of the language, i.e. for each word of the language all words that are permutations of its letters are included. The commutative closure of $\{a^n b^n c^n\}$ is the language L_p that is shown in Example 1. A language is called commutative if it is the commutative closure of itself, e.g., L_p is commutative. Now some further concepts are recalled formally based on [3, 6, 11].

Definition 3. *Let T be a finite alphabet. A reflexive and symmetric binary relation $D \subset T \times T$ is a* dependency relation *on T: $(a, a) \in D$ for every $a \in T$, further, if $(a, b) \in D$ for some letters $a, b \in T$, then also $(b, a) \in D$. The* independency relation *(or also called* commutation*) induced by D is $I = (T \times T) \setminus D$. Obviously, I is irreflexive and symmetric. Let $w \equiv w'$ if $w = uabv$, $w' = ubav$ and $a, b \in T$, $u, v \in T^*$, $(a, b) \in I$. The equivalence relation $\equiv$ induced by I on T^* is called a* partial commutation. *For each word $w \in T^*$ the set of words that are equivalent to w form the commutation class $[w]$ of w. These classes are called* traces. *The definition is also extended to languages: $[L] = \{[w] \mid w \in L\}$, then $[L]$ is a* trace language *that consists of traces. Further, it is usual to use the language $L' = \bigcup_{[w] \in [L]} [w] = \{w' \mid [w'] \in [L]\}$. This language is the linearization of the trace language $[L]$, or shortly (if it is not confusing) also called a trace language. The equivalence class of the empty word is called empty trace. Let L be a context-free (regular) language, and I be an independency relation. Further, let $[M]$ be the set*

of classes induced by the independency relation I among the words over the alphabet T, i.e., the trace languages defined by I. Then $[L] \subset [M]$ is a context-free (rational, resp.) trace language. *Further,* **CFTRACE** *and* **RAT** *denote the sets of (linearizations of) the context-free and rational trace languages, respectively.*

Trajectories of grid paths are closely related since by permuting the order of some steps leads to the same target point in the grid. Trajectories, traces and trace languages are intensively studied in relation to modeling parallel processes and concurrency theory [7, 10, 22].

THEOREM 1. **CFTRACE** *is a subset of* **PERM**.

PROOF. Let L be a context-free language and I be an independency relation on the alphabet T. Let $G = (N, T, S, P)$ be a grammar generating the context-free language L in a Chomsky normal form. Let $G' = (N', T, S, P')$ be a new grammar with $N' = N \cup \{X_a \mid a \in T\}$, where each X_a is a new nonterminal. Let $P' = P \cup \{X_a \to a \mid a \in T\} \cup \{A \to X_a \mid A \to a \in P, A \in N\} \setminus \{A \to a \mid A \in N, a \in T\}$ Clearly, G' also generates L. Now, let $G_t = (N', T, S, P'')$ with $P'' = P' \cup \{X_a X_b \to X_b X_a \mid (a, b) \in I\}$. The permutation grammar G'' generates the context-free trace language defined by L and I. □

Each commutative semi-linear language is in **RAT**, and also in **CFTRACE**, thus, it is in **PERM**.

4. NEW EXTENDED VARIANTS OF PUSHDOWN AUTOMATA

In this section, we define some extensions of the pushdown automata. Note that here we use only stateless pushdown automata which are usually used without loss of generality [29]. Moreover, we use automata with special transition mappings that are closely related to grammars in normal forms. Formally, a *pushdown automaton* (PDA) is a tuple (T, Γ, δ, Z_0) with the input and stack alphabets T and Γ, respectively, $Z_0 \in \Gamma$ is the initial symbol in the stack, and $\delta : (T \cup \{\lambda\}) \times \Gamma \to 2^{\Gamma^*}$ is the transition relation where only finite subsets of Γ^* are used. A PDA simulates left-most derivations in a corresponding context-free grammar.

DEFINITION 4. *A* pushdown automaton with a temporary storage *(PDATS) is a 4-tuple (T, Γ, δ, Z_0) as the pushdown automata. The stack alphabet Γ is also used as the set of possible symbols in the temporary storage. A configuration of a PDATS is a triple (v, z, t), where v is the remaining input, z is the contents of the stack and t is the contents of the temporary storage. The computation step relation ($\vdash$) is defined among the configurations as follows. Suppose $\lambda \in \delta(a, A)$ with $a \in T, A \in \Gamma$, then for all $v \in T^*$ and $z \in \Gamma^*$: $(av, zA, \lambda) \vdash (v, z, \lambda)$. Suppose $z' \in \delta(\lambda, A)$ with $A \in \Gamma$, then for all $v \in T^*$ and $z, t \in \Gamma^*$: $(v, zA, t) \vdash (v, zz', t)$. Moreover for all $v \in T^*$ and $z, z' \in \Gamma^*$ $(v, zA, z') \vdash (v, z, z'A)$ and $(v, z, z'A) \vdash (v, zA, z')$. We can also define $\vdash^*$ as the reflexive and transitive closure of $\vdash$. By this relation we can define the accepted language: $L = \{v \mid (v, Z_0, \lambda) \vdash^* (\lambda, \lambda, \lambda), v \in T^*\}$.*

By the help of the temporary storage the PDATS can simulate arbitrary context-free derivations. Since it has no inner memory (it has no states) it cannot do more. By inductions on the set of configurations of the automata and on the set of sentential forms of the grammar in normal form it can be proven that we have:

PROPOSITION 1. *The class of languages accepted by pushdown automata with temporary storage coincides with the class of context-free languages.*

DEFINITION 5. *A* pushdown automaton with binary permutation *(PDABP) is a tuple $(T, \Gamma, \delta, \rho, Z_0)$. The components T, Γ, δ, Z_0 are the same as at PDA, and ρ is a binary relation on $\Gamma \times \Gamma$. A configuration of a PDABP is a pair (v, z) (as at a PDA). The relation $\vdash$ on configurations is defined as follows. Suppose $\lambda \in \delta(a, A)$ with $a \in T, A \in \Gamma$, then for all $v \in T^*$ and $z \in \Gamma^*$: $(av, zA) \vdash (v, z)$. Suppose $z' \in \delta(\lambda, A)$ with $A \in \Gamma$, then for all $v \in T^*$ and $z \in \Gamma^*$: $(v, zA) \vdash (v, zz')$. Suppose $(A, B) \in \rho$ with some $A, B \in \Gamma$ then $(v, zBA) \vdash (v, zAB)$. Further, $\vdash^*$ is defined as usual. Finally, $L = \{v \mid (v, Z_0) \vdash^* (\lambda, \lambda), v \in T^*\}$ is the language accepted by the PDABP.*

Since the new operation, the binary permutation can interchange the top two symbols in the stack, PDABP can simulate left-most derivations in permutation grammars. The next statement is the consequence of the fact that left-most derivations generate only context-free languages independently of the form of the productions of the grammar [29]. Note that left-most derivations without loosing the generative power in the context-sensitive case are shown in [17].

PROPOSITION 2. *The class of languages accepted by PDABP coincides with the class of context-free languages.*

DEFINITION 6. *A* pushdown automaton with temporary storage and binary permutation *(PDATSBP) is a pentuple $(T, \Gamma, \delta, \rho, Z_0)$. The components are the same as at PDABP. The configuration is defined in the same way as at PDATS. The relation $\vdash$ of a PDATS is extended with the following steps. Suppose $(A, B) \in \rho$ then $(v, zBA, t) \vdash (v, zAB, t)$ (with any $v \in T^*$, $z, t \in \Gamma^*$). The definitions of $\vdash^*$ and of the accepted language are straightforward.*

Actually, in PDATSBP we combined PDATS and PDABP. They can be redefined such that the top of the stack and the top of the temporary storage are interchanged. Although both PDATS and PDABP are equivalent to PDA concerning their accepting capacity, we have the next theorem.

THEOREM 2. *The class* **PERM** *is accepted by PDATSBP.*

The proof goes by induction. There is a correspondence between the configurations of the automata and the sentential forms of the grammars (using derivations where it is signed which part of the sentential form is rewritten by an applicable production).

Now, our example is continued.

EXAMPLE 1 (CONT.) *The language L_p is accepted by the PDATSBP $M = (\{a, b, c\}, \{S, A, B, C\}, \delta, \{(A, C), (A, B), (B, C), (C, B)\}, S)$, where $CBA \in \delta(\lambda, S)$, $CBSA \in \delta(\lambda, S)$, $\lambda \in \delta(a, A)$, $\lambda \in \delta(b, B)$, $\lambda \in \delta(c, C)$.*

As a corollary of Theorems 1 and 2, we have the following important result.

COROLLARY 1. *There is a PDATSBP for every context-free trace language that accepts it.*

5. CONCLUSIONS

Permutation grammars and the newly investigated variant of pushdown automata are capable to model parallel processes where the base language is context-free and also some linguistically important structures. It is known that the word problem for permutation grammars is NP-complete in general [21]. It is a future task to analyse what is the case for context-free trace languages or at least for those of them which may appear in modeling and in applications. Relation of these classes to languages accepted by pushdown automata with translucent letters [23, 24] should also be studied in the future. About the normal form for permutation grammars, it is an open problem whether the chain rules $A \to B$ can be eliminated.

6. REFERENCES

[1] S. Crvenković, I. Dolinka and Z. Ésik. On equations for union-free regular languages. *Inform. and Comput.* 164/1 pp. 152–172, 2001.

[2] J. Dassow and Gh. Păun. *Regulated rewriting in formal language theory*. Springer-Verlag, Berlin, 1989.

[3] V. Diekert and G. Rozenberg (eds.). *The book of traces*. World Scientific, River Edge, NJ, USA, 1995.

[4] K. Fogarasi and B. Nagy. A nondeterministic parser for Perm_2 grammars. *Abstract volume of the 10th Joint Conference on Mathematics and Computer Science (MaCS)*, Cluj-Napoca, Romania, 2014.

[5] M. A. Harrison. *Introduction to formal language theory*, Addison-Wesley, Reading, AM, USA, 1978.

[6] T. Herendi and B. Nagy. *Parallel Approach of Algorithms*. Typotex, Budapest, 2014.

[7] R. Janicki, J. Kleijn, M. Koutny and L. Mikulski. *Paradigms of Concurrency – Observations, Behaviours, and Systems – a Petri Net View*. Studies in Computational Intelligence 1020, Springer, 2022.

[8] G. Madejski. Infinite hierarchy of permutation languages. *Fundam. Inform.* 130/3 pp. 263–274, 2014.

[9] E. Mäkinen. On permutative grammars generating context-free languages. *BIT* 25/4 pp. 604–610, 1985.

[10] A. Mateescu, G. Rozenberg and A. Salomaa. Shuffle on trajectories: syntactic constraints. *Theoretical Computer Science* 197 pp. 1–56, 1998.

[11] A. W. Mazurkiewicz. Trace Theory. In: *Petri Nets: Central Models and Their Properties, Advances in Petri Nets, Part II, LNCS*, 255, pp. 279–324, 1986.

[12] B. Nagy. Union-free regular languages and 1-cycle-free-path-automata. *Publ. Math. Debrecen* 68 pp. 183–197, 2006.

[13] B. Nagy. Languages generated by context-free and type $AB \to BA$ rules. In: *Proc. CINTI 2007: 8th Int. Symp. of Hung. Research. Comput. Intelligence Inf.*, Budapest, Hungary, pp. 563–572, 2007.

[14] B. Nagy. Languages generated by context-free and type $AB \to BA$ rules. *J Autom. Lang. Combin.* 14 pp. 175–186, 2009.

[15] B. Nagy. Permutation languages in formal linguistics. In: *Proc. IWANN 2009, Part I, LNCS* 5517 pp. 504—511, 2009.

[16] B. Nagy. On a hierarchy of permutation languages. In: *Automata, Formal Languages and Algebraic Systems*, pp. 163–178, World Scientific, Singapore, 2010.

[17] B. Nagy. Derivation Trees for Context-Sensitive Grammars. In: *Automata, Formal Languages and Algebraic Systems*, pp. 179–199, World Scientific, Singapore, 2010.

[18] 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.

[19] B. Nagy. Linguistic power of permutation languages by regular help. In: *Bio-inspired models for natural and formal languages*. pp. 135–152, Cambridge Scholars, 2011.

[20] B. Nagy. On the NP-completeness of the word problem for permutation grammars. In: *Abstract volume of the Int. Workshop in honor of Masami Ito's 77th birthday and Pál Dömösi's 75th birthday: DLT's Satellite Workshop in Kyoto*, Japan, September 2018.

[21] B. Nagy. On the Membership Problem of Permutation Grammars – A Direct Proof of NP-Completeness. *Int. J. Found. Comput. Sci.* 31/4 pp. 515–525, 2020.

[22] B. Nagy and A. A. Akkeles. Trajectories and Traces on Non-traditional Regular Tessellations of the Plane. In: *Combinatorial Image Analysis, 18th Int. Workshop, IWCIA 2017, LNCS* 10256, pp. 16–29, 2017.

[23] B. Nagy and F. Otto. An Automata-Theoretical Characterization of Context-Free Trace Languages. In: *SOFSEM 2011: 37th Conference on Current Trends in Theory and Practice of Computer Science, LNCS* 6543, pp. 406–417 2011.

[24] B. Nagy and F. Otto. CD-systems of stateless deterministic R(1)-automata governed by an external pushdown store. *RAIRO Theor. Informatics Appl.* 45 pp. 413–448, (2011).

[25] 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.

[26] 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.

[27] M. Penttonen. One-sided and two-sided context in formal grammars. *Inf. Control* 25 pp. 371–392, 1974.

[28] R. Schott and J. C. Spehner. Two optimal parallel algorithms on the commutation class of a word *Theoretical Computer Science* 324 pp. 107–131, 2004.

[29] G. Rozenberg and A. Salomaa (eds.). *Handbook of formal languages*. Springer-Verlag, Berlin, 1997.

Towards a Category-Theoretic Informatics Model of PSPP Linkages in Biomaterials

Sylvert Prian Tahalea
sylvert@inf.u-szeged.hu
University of Szeged, Hungary
Universitas Pembangunan Nasional Veteran Yogyakarta, Indonesia

Miklós Krész
miklos.kresz@innorenew.eu
InnoRenew CoE, UP IAM and UP FAMNIT, University of Primorska, Slovenia
University of Szeged, Hungary

Abstract

This paper introduces a category-theoretic framework for modeling the causal and structural relationships in the Processing-Structure-Property-Performance (PSPP) paradigm, with a case study of bioluminescent bacterial interactions on engineered surfaces. The composition and interdependence of material behavior in mathematical form are captured by defining the stages of PSPP linkages into objects and their transformations as morphisms. This model is applied to a case study involving bacterial cultivation on three different engineered surfaces. The universal input configurations and multi-objective performance targets are identified using limit and colimit, respectively. Additionally, the functors and natural transformations applied compare the bacterial behaviors across material categories, illustrating the structural relationship between bacteria. While the current model assumes deterministic and discrete transitions, it opens pathways for future development using enriched, fibred, and computational categorical tools. This work demonstrates the potential of category theory as a unifying language for scientific modeling in biomaterials research.

Keywords

category theory, PSPP, biomaterials, natural transformations, functor

1 Introduction

The property-structure-processing-performance (PSPP) linkages are a core paradigm in material science to conceptualise the interdependence between material properties, processing methods, structural features, and resulting performance [12, 8, 1]. The PSPP linkages come from process-structure-property-performance relationships, which build up like a chain in a bottom-up (simulation-driven approach) or top-down (process design approach) [7]. Process refers to the methods and techniques used to transform the raw materials into finished products with desired shapes and properties; structure refers to the arrangements of atoms, molecules, or phases within the materials; properties are the measurable responses of a material to the stimuli and can be classified into various categories; while performance is the materials' ability to function effectively in a specific application, considering its properties and conditions [7, 6, 10, 23]. Despite its importance, PSPP linkages are described in an informal and domain-specific manner, which hinders systematic integration across disciplines and limits its computational implementation [15, 22]. This lack of formalism becomes a barrier when attempting to analyse the PSPP relationships at scale, particularly in data-driven or interdisciplinary contexts [25, 11].

Recent advances in material informatics have highlighted the importance of formal representation for enabling computational reasoning, interoperability, and data integration [2, 24, 14, 7]. In parallel, category theory has emerged as a mathematical language for abstraction and composition, providing a rigorous way to describe the structures and relationships between them [19]. Originally developed in mathematics, category theory has found its way to be implemented in computer science, biology, data science, where it serves to represent the complex interactions [18, 4].

The category theory implementation in informatics offers a pathway to move beyond the classical representation of materials knowledge, such as graphs and ontologies, to ensure the consistency, scalability, and interoperability in computational models of materials science. It has been recognised as a unifying language for science, offering tools such as objects, morphisms, limits, and colimits to formally capture relationships between complex systems [18, 4]. To address this gap, a categorical model is proposed in this paper to formalise the PSPP interconnections in a structured and compositional way. Our contribution lies in defining this formal representation, demonstrating how it can unify disparate descriptions into a coherent framework. The scope of this work is focused on establishing the formal model rather than developing a full computational implementation. To illustrate its potential, a small example that highlights how categorical structure captures PSPP linkages is provided.

The remainder of the paper is organised as follows. **Section 2** explains the methodology, **section 3** presents the case study example, **section 4** concludes the contribution, limitations, and future research.

2 Methodology

This section explains the category, subcategories, functors, natural transformations, limits, and colimits of the PSPP linkages.

2.1 PSPP Linkage

PSPP illustrates the chain of relationships that connect processing parameters to the resulting structure (microstructure), which governs properties and thus impacts component performance—a critical multiscale, multiphysics mapping essential for the modeling and design of materials [7]. Processing techniques, resultant structures, material qualities, and performance outcomes are all PSPP stages that are frequently examined separately or depicted informally using diagrams that lack formal semantics and mathematical rigor [7, 26, 3]. The PSPP framework helps to formalize the causal pathway from how a material is processed to its performance outcome, e.g. bacteria-surface interaction for bioluminescence, which is later used as the study case. The bacteria-surface

MATCOS-25, 9–10 October 2025, Koper, Slovenia

interaction involves a complex and multilayered relationship between the processing method and the metabolic output. However, current approaches for modeling these interactions suffer from several limitations, such as fragmented representation, lack of a unified framework, and lack of composability, which can be addressed with a category-theoretical model. The PSPP linkage presented in Figure 1 shows how a specific processing method (e.g., oxygen plasma treatment) alters the surface structure, which in turn affects bacterial interaction properties (such as adhesion or viability), ultimately leading to a measurable performance outcome like bioluminescence intensity.

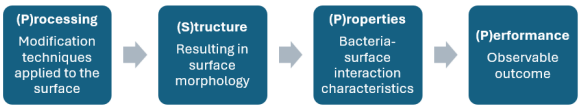


Figure 1: PSPP linkages in bacteria-surface interaction.

2.2 PSPP Category

The category theoretic model of PSPP is defined as follows.

2.2.1 Objects. Let C be a category. The set of the objects in C is defined as:

$$\mathrm{Ob}(C) = \{P, S, Pr, Pe\}$$

where: P represents the manufacturing process or processing in short form, S represents materials structure, Pr represents material properties, and Pe represents material performance. Thus, $\mathrm{Ob}(C)$ encodes the fundamental entities of the PSPP framework as objects within the categorical structure.

2.2.2 Morphisms. Define the set of morphisms (arrows) in the category C as follows:

$$Hom_{(C)}(X, Y)$$

Where $X, Y \in P, S, Pr, Pe$ and each morphism represents a relationship or transformation from one object to another. The morphisms in the category C are defined as follows:

$f : P \to S$	(Processing affect Structure)
$g : S \to Pr$	(Structure determine Properties)
$h : Pr \to Pe$	(Properties dictate Performance)
$i : Pe \to P$	(Performance feedback to Processing)
$j : P \to S'$	(Processing results Modified Structure)
$k : S' \to Pr$	(Modified Structure determine Properties)
$l : Pr \to Pr'$	(Properties transform to Modified Properties)
$m : Pr' \to Pe$	(Modified Properties dictate Performance)

2.2.3 Composition. The composition of morphisms in the category C follows the associativity and identity properties:

$$(h \circ g) \circ f = h \circ (g \circ f)$$

and for each object $X \in \mathrm{Ob}(C)$, there exists an identity morphism $\mathrm{id}_X : X \to X$ such that:

$$\mathrm{id}_Y \circ f = f \quad \text{and} \quad g \circ \mathrm{id}_X = g$$

In example for the bacterial surface interaction case, P = "oxygen plasma processing", S = "hydrophilic glass surface", Pr = "moderate bacterial adhesion, low EPS density", Pe = "5 μW/cm^2 luminescence output".
Define the morphisms in category C_{PSPP} as:

$$f : P \to S, \quad , g : S \to Pr, \quad , \text{and } h : Pr \to Pe$$

Then the composite morphisms are:

$$g \circ f : P \to Pr \quad \text{and} \quad h \circ g : S \to Pe$$

This composite morphism represents the entire PSPP causal chain from processing to performance. For this case, it maps:

$$\text{"plasma treatment"} \longrightarrow \text{"5 } \mu\text{W/cm}^2 \text{ luminescence"}$$

via structure and property stages. The identity morphism at each stage (e.g., $\mathrm{id}_S : S \to S$) satisfies: $g \circ \mathrm{id}_S = g, \quad \mathrm{id}_{Pr} \circ g = g$ confirming the category's identity and associativity laws.

2.2.4 Diagram Representation. A typical morphism of the PSPP category is only represented by processing, structure, properties, and performance. However, the processing can result in modified structure and properties, which are shown in Figure 2.

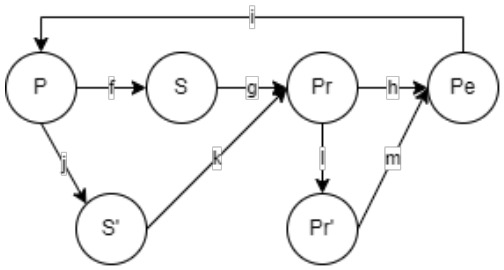


Figure 2: PSPP diagram representation including modified structure and properties.

2.3 Subcategories of PSPP Model

The category model of PSPP linkage has several subcategories by nature (as presented in Figure 2), which are meaningful subsets of the objects and morphisms that still satisfy the definition of a category.

2.3.1 Canonical Path Subcategory. The main causal chain C_{core} without branching.

- **Objects**. $\mathrm{Ob}(C_{core}) = P, S, Pr, Pe$
- **Morphisms**. $f : P \to S, g : S \to Pr, h : Pr \to Pe$

This paper will be focusing only on this canonical path or the main linkages.

2.3.2 Modified Path Subcategory. This modified subcategory C_{mod} is an expansion of the canonical path that includes only the branching route.

- **Objects**. $\mathrm{Ob}(C_{core}) = P, S, S', Pr, Pr', Pe$
- **Morphisms**. $f : P \to S, j : P \to S', g : S \to Pr, k : S \to Pr', l : Pr \to Pr', m : Pr' \to Pe$

2.3.3 Feedback Loop Subcategory. This loop subcategory C_{loop} focuses on feedback interactions within the model.

- **Objects**. $\mathrm{Ob}(C_{core}) = P, S, Pr, Pe$
- **Morphisms**. $f, g, h, i : Pe \to P$

2.3.4 Application Specific Subcategory. This subcategory can be expanded based on the application focus field. The application on bacteria-surface interaction will be used as the case study in this article, illustrated by following examples.

- Biofilm subcategory $C_{biofilm}$, with objects such as: $P, S, Pr_{adhesion}, Pe_{biofilm}$
- Bioluminescence subcategory C_{biolum}, with objects such as: $P, S, Pr_{light}, Pe_{lum}$

2.4 Functor and Natural Transformation

To define a functor and natural transformation of this model, first, we need to define another category which can be processed with this functor and natural transformation logic. This subsection was prepared in advance to preserve the relation and transformation.

2.4.1 Functor. A functor $F : C_1 \to C_2$ between categories C_1 and C_2 is defined by:

(1) **Object mapping**: for each object $X \in \mathrm{Ob}(C_1)$, there is object $F(X) \in \mathrm{Ob}(C_2)$
(2) **Morphism mapping**: for each morphism $f : X \to Y \in \mathrm{Hom}_{C_1}(X, Y)$, there is exist a morphism $F(f) : F(X) \to F(Y) \in \mathrm{Hom}_{C_2}(F(X), F(Y))$

such that:

- Composition is preserved:

$$F(f \circ g) = F(f) \circ F(g), \forall f, g \in \mathrm{Mor}(C_1)$$

- Identities are preserved:

$$F(\mathrm{id}_X) = \mathrm{id}_{F(X)}, \forall X \in \mathrm{Mor}(C_1)$$

Functor of PSPP category model example in our use case: Considering two distinct PSPP categories:

- C_{glass}: a category of PSPP stages for *Pseudomonas flourescens* treated on **glass** surface.
- C_{wood}: a category of PSPP stages for *Pseudomonas flourescens* on functionalized **wood**.

Then we can define Functor $F : C_{glass} \to C_{wood}$:

- **Objects**
  - $P_{glass} \to P_{wood}$: oxygen plasma cleaning → enzym-based-pretreatement
  - $S_{glass} \to S_{wood}$: smooth hydropphilic silica → porous lignovellulosic surface
  - $Pr_{glass} \to Pr_{wood}$: low biofilm density → moderate biofilm density
  - $Pe_{glass} \to Pe_{wood}$: weak bioluminescence → stable bioluminescence overtime
- **Morphisms**
  - $f_{glass} \to f_{wood}$: plasma alters hydrophilicity → enzymatic treatement alters lignin exposure
  - $g_{glass} \to g_{wood}$: smooth surface limit adhesion → roughness enhances microbial anchoring
  - $h_{glass} \to h_{wood}$: low metabolic activity → higher growth and igh emission
  - $i_{glass} \to i_{wood}$: bioluminescence feedback informas wood resin optimization

Having established the functor, a later section presents the use case of the PSPP model.

2.4.2 Natural Transformation. Let $F, G : C_1 \to C_2$ be functors. A natural transformation $\eta : F \Rightarrow G$ is a family of morphisms $\eta_X : F(X) \to G(X), \quad \forall X \in \mathrm{Ob}(C_1)$, such that for every morphism $f : X \to Y$ in C_1, $\eta_Y \circ F(f) = G(f) \circ \eta_X$.

Natural transformation of PSPP category model of our use case: Suppose there are $F, G : C_{concrete} \to C_{bact}$ two functors:

- F: using *Pseudomonas flourescence*
- G: using *Serratia marcescens*

The natural transformation $\eta : F \Rightarrow G$, for each PSPP stage (object) transformation, comes from one species to another.

2.5 Limit and Colimit

2.5.1 Cone. Let $\mathcal{J}$ be a category, C a category, and let $D : \mathcal{J} \to C$ be a functor (called a diagram in C). A cone over D consists of:

(1) An object $N \in \mathrm{Ob}(C)$, called the apex of the cone,
(2) A family of morphism in C, $\{\phi_j : N \to D(j)\}_{j \in \mathrm{Ob}(\mathcal{J})}$, such that for every morphism $f : j \to k$ in $\mathcal{J}$, the following commutative condition holds: $D(f) \circ \phi_j = \phi_k$

In other words, a cone points into every object in the functor D from the apex using the morphisms.

Cone of PSPP model of our use case: Let there be three different processing routes for bioluminescent bacterial coating: F_1: Processing 1, F_2: Processing 2, and F_3: Processing 3; which produce a full chain $P_i \xrightarrow{f_i} S_i \xrightarrow{g_i} Pr_i \xrightarrow{h_i} Pe_i$ and the functor F of this output can be expressed as: $F : \mathcal{J} \to C$, where $F(j) = Pe_j$.

2.5.2 Limit. Let $F : \mathcal{J} \to C$ be a functor. A limit of F is an object $L \in \mathrm{Ob}(C)$ together with morphisms $\pi_j : L \to F(j), \quad \forall j \in \mathrm{Ob}(\mathcal{J})$, such that for every morphism $f : j \to j'$ in $\mathcal{J}$,

$$F(f) \circ \pi_j = \pi_{j'}$$

Moreover, (L, π_j) is *universal*: for any other cone $\{\psi_j : N \to F(j)\}_{j \in \mathrm{Ob}(\mathcal{J})}$, there exists a unique morphism $u : N \to L$ such that

$$\psi_j = \pi_j \circ u, \quad \forall j \in \mathrm{Ob}(\mathcal{J}).$$

Limit of PSPP Model of our use case: The output of the PSPP linkage lies at the performance stage; the subdiagram from processing to performance can be used to find the optimal solution through the PSPP model. Assume that there are three different processing routes: (1) $F1 : P_1 \to S_1 \to Pr_1 \to Pe_1$, (2) $F2 : P_2 \to S_2 \to Pr_2 \to Pe_2$, and (3) $F1 : P_3 \to S_3 \to Pr_3 \to Pe_3$. The limit object L is universal, thus there exist projections to each case:

$$\pi_i : L \to Pe_i \text{ for } i = 1, 2, 3$$

2.5.3 Colimit. Let $F : \mathcal{J} \to C$ be a functor. A colimit of F is an object $C \in \mathrm{Ob}(C)$ together with morphisms $\iota_j : F(j) \to C, \forall j \in \mathrm{Ob}(\mathcal{J})$, such that for every morphism $f : j \to j'$ in $\mathcal{J}$, $\iota_{j'} \circ F(f) = \iota_j$. Moreover, (C, ι_j) is *universal*: for any other cocone $\{\psi_j : F(j) \to N\}_{j \in \mathrm{Ob}(\mathcal{J})}$, there exists a unique morphism $v : C \to N$ such that $\psi_j = v \circ \iota_j, \quad \forall j \in \mathrm{Ob}(\mathcal{J})$.

Colimit of PSPP Model of our use case: In the study of the PSPP linkage, suppose that there are three different properties, as follows. (1) $F_1 : Pr_A \to Pe_A$, (2) $F_2 : Pr_B \to Pe_B$, (3) $F_3 : Pr_C \to Pe_C$; one of the objectives is to merge or combine all the performances to produce a unified performance outcome $\iota_i : Pe_i \to C$ and any projection results factors uniquely through C.

3 Case Study: Optimising Surface Design for Bioluminescent Bacteria

The case study of this model is to determine the best surface configuration to support bioluminescence output from *Pseudomonas flourescens*. There are three different surface preparation routes

[16, 13, 17, 9, 20, 5, 21] using the PSPP framework and interpreting the model categorically, such as glass, wood, and concrete.

3.1 Experimental Pipelines

Each processing route is a distinct PSSP chain. (1) Oxygen plasma on glass: P_1: Oxygen plasma cleaning; S_1: Enhanced surface hydrophilicity and cleanliness; Pr_1: Reduced adhesion of bacteria; Pe_1: Produce a low bioluminescence signal caused by the low density of biofilm. (2) Enzyme treatment on wood: P_2: Enzymatic pretreatment of the surface; S_2: Surface may degrade and reduce the cohesion; Pr_2: Low biofilm viability; Pe_2: Low signal output due to diminished bacterial presence. (3) Sol-gel coated concrete: P_3: Sol-gel encapsulation; S_3: Potentially surrounded by silica gel, contains methanol; Pr_3: Severely limits bacteria's metabolic activity; Pe_3: Light output reduced drastically. The PSPP linkage on this study case is presented in Figure 1.

3.2 Categorical diagram

Given that each route forms a PSPP morphism chain:
$P_i \xrightarrow{f_i} S_i \xrightarrow{g_i} Pr_i \xrightarrow{h_i} Pe_i$, which is presented in Figure 3.

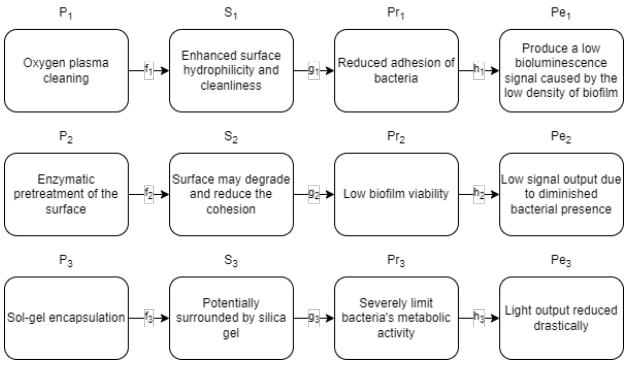


Figure 3: The routes of PSPP morphism chain

3.3 Limit Construction

A universal configuration L (e.g., a hybrid substrate combining features of all three processes) is defined as a cone in Figure 4. Universal configuration L represents an ideal design that projects

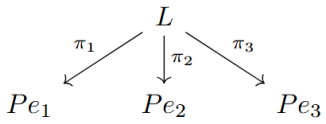


Figure 4: L is the universal input that relates to all observed performances

to each observed performance, supporting general optimization. The categorical properties are as follows.

- $L \to Pe_i$ form morphims
- Commutativity must hold: any internal morphism between Pe_i must be maintained if they exist
- L is the limit and universal: every cone factors uniquely through L

3.4 Colimit Construction

Conversely, a colimit object C which integrates all outcomes into a general-purpose performance target should be defined as presented in Figure 5. C serves as a composite performance

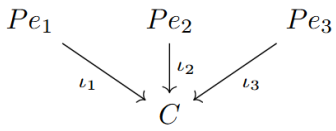


Figure 5: A cocone with apex C absorbing all performances output

specification, capturing multi-objective behaviors (e.g., balancing intensity and longevity) with the following categorical properties.

- Morphisms $\iota_i : Pe_i \to C$
- C is the colimit of every projection from Pe_i factors uniquely through C

3.5 Natural transformation

Suppose there is another experiment with the same three surfaces but different bacteria, i.e. *Serratia marcescens*. The functors and the natural transformation are presented in Figure 6 and can be defined as follows.

- **Functors:** F: PSPP route using *P. flourescens* and G: PSPP route using *S. marcescens*
- **Natural transformation:** $\eta : F \Rightarrow G$, where ηP: translate the process (e.g plasma treatment) parameters, ηS: map structure adaptation, and $\eta Pr, \eta Pe$: transform properties and performances across bacterial strains.

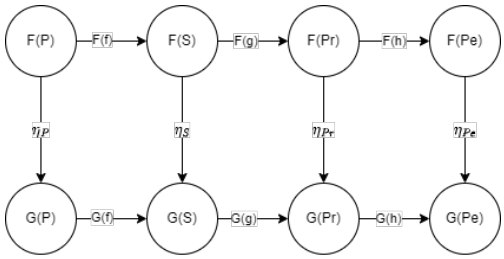


Figure 6: Natural transformation from *P. flourescence* to *S. marcescens*

3.6 Discussion

The categorical analysis of PSPP linkages in bioluminescent bacterial systems provides a structured lens to evaluate and compare multiple experimental design routes. In this case study, three distinct processing strategies—oxygen plasma treatment on glass, enzyme pretreatment on wood, and sol-gel silica encapsulation on concrete—were formalised as morphism chains within the category $\mathcal{C}_{\text{PSPP}}$. Each route defines a unique instance of a functor mapping the canonical PSPP stages (Processing → Structure → Property → Performance). The resulting categorical diagram allowed us to analyse their behaviours both individually and relationally.
The introduction of a *limit* object L, structured as a cone over the performance outcomes Pe_1, Pe_2, and Pe_3, offered a formal

approach to identifying an optimal or universal input configuration capable of reproducing all observed performance behaviors. Practically, L can be interpreted as a generalized hybrid surface or processing strategy that maintains compatibility with the range of observed bioluminescence behaviors. Conversely, the dual construction of a *colimit* object C, defined via morphisms $\iota_i : Pe_i \to C$, enabled the synthesis of a composite performance profile, which may serve as a design target in multi-objective optimization tasks.

This dual analysis captures generalisation (limit) and integration (colimit), offering a flexible modelling approach. The model supports the comparative approach through natural transformation between bacteria through functors. For example, transitions between *Pseudomonas fluorescens* and *Serratia marcescens* can be formalised as structure-preserving morphism families, highlighting how microbial adaptation maps across processing contexts. This indicates category theory's ability to enable reasoning and potential computational automation in material informatics.

4 Conclusion

With a brief case study of bioluminescent bacterial interactions on artificial surfaces, this paper offers a category-theoretic framework for modelling PSPP linkage in biomaterials. In order to allow compositional thinking, a formal categorical structure $\mathcal{C}_{\text{PSPP}}$ is designed to represent PSPP stages as objects and causal relationships as morphisms. The model captures both variability and unification across various experimental settings by using functors, natural transformations, and universal constructions like limits and colimits.

A case study of *Pseudomonas fluorescens* across three surface treatments shows the effectiveness of this approach in determining optimal input configurations (by limits) and composite performance targets (via colimits). The application of natural transformations furthers biological comparison across strains. The main contributions of this work are: (1) a formal mathematical structure for PSPP links based on category theory and (2) the integration of experimental design with universal structures like cones and cocones. However, there are some limits. The current model assumes discrete and predictable transitions between PSPP phases, does not account for uncertainty, dynamics, or probabilistic behavior, and lacks an implemented computational instance functor to real data.

Future work will focus on enriching the categorical model with quantitative structure (e.g., cost, confidence, time) via enriched categories; defining pullbacks and pushouts for constraints.

Acknowledgements

The research was supported by the BioLOG project: the second author is grateful for the support of the National Centre of Science (NCN) through grant DEC-2020/39/I/HS4/03533, the Slovenian Research and Innovation Agency (ARIS) through grant N1-0223, and the Austrian Science Fund (FWF) through grant I 5443-N. This work is supported by the ARIS research program P1-0404 and by the research program CogniCom (0013103) at the University of Primorska.

References

[1] Ankit Agrawal and Alok Choudhary. 2016. Perspective: materials informatics and big data: realization of the "fourth paradigm" of science in materials science. *Apl Materials*, 4, 5. DOI: 10.1063/1.4946894.

[2] Toshihiro Ashino. 2010. Materials ontology: an infrastructure for exchanging materials information and knowledge. *Data Science Journal*, 9, 54–61. DOI: 10.2481/dsj.008-041.

[3] Igor Baskin and Yair Ein-Eli. 2022. Electrochemoinformatics as an emerging scientific field for designing materials and electrochemical energy storage and conversion devices—an application in battery science and technology. *Advanced Energy Materials*, 12, 48, 2202380. DOI: 10.1002/aenm.202202380.

[4] Brendan Fong and David I Spivak. 2018. Seven sketches in compositionality: an invitation to applied category theory. *arXiv preprint arXiv:1803.05316*. DOI: 10.48550/arXiv.1803.05316.

[5] Wissam Ghach, M. Etienne, V. Urbanová, Frédéric P. A. Jorand, and A. Walcarius. 2014. Sol–gel based 'artificial' biofilm from pseudomonas fluorescens using bovine heart cytochrome c as electron mediator. *Electrochemistry Communications*, 38, 71–74. DOI: 10.1016/J.ELECOM.2013.11.001.

[6] D. Gu, Xinyu Shi, R. Poprawe, D. Bourell, R. Setchi, and Jihong Zhu. 2021. Material-structure-performance integrated laser-metal additive manufacturing. *Science*, 372. DOI: 10.1126/science.abg1487.

[7] Seyed Mahdi Hashemi, Soroush Parvizi, Haniyeh Baghbanijavid, Alvin TL Tan, Mohammadreza Nematollahi, Ali Ramazani, Nicholas X Fang, and Mohammad Elahinia. 2022. Computational modelling of process–structure–property–performance relationships in metal additive manufacturing: a review. *International Materials Reviews*, 67, 1, 1–46. DOI: 10.1080/09506608.2020.1868889.

[8] Surya R Kalidindi and Marc De Graef. 2015. Materials data science: current status and future outlook. *Annual Review of Materials Research*, 45, 1, 171–193. DOI: 10.1146/annurev-matsci-070214-020844.

[9] A. Kotlobay et al. 2018. Genetically encodable bioluminescent system from fungi. *Proceedings of the National Academy of Sciences of the United States of America*, 115, 12728–12732. DOI: 10.1073/pnas.1803615115.

[10] Wenqi Li and Jian Shi. 2023. Lignin-derived carbon material for electrochemical energy storage applications: insight into the process-structure-properties-performance correlations. *Frontiers in Bioengineering and Biotechnology*, 11. DOI: 10.3389/fbioe.2023.1121027.

[11] Zhanzhao Li, Te Pei, Weichao Ying, Wil V Srubar III, Rui Zhang, Jinyoung Yoon, Hailong Ye, Ismaila Dabo, and Aleksandra Radlińska. 2024. Can domain knowledge benefit machine learning for concrete property prediction? *Journal of the American Ceramic Society*, 107, 3, 1582–1602. DOI: 10.1111/jace.19549.

[12] Gregory B Olson. 1997. Computational design of hierarchically structured materials. *Science*, 277, 5330, 1237–1242. DOI: 10.1126/science.277.5330.1237.

[13] Rahman Rahmanpour and T. Bugg. 2015. Characterisation of dyp-type peroxidases from pseudomonas fluorescens pf-5: oxidation of mn(ii) and polymeric lignin by dyp1b. *Archives of biochemistry and biophysics*, 574, 93–8. DOI: 10.1016/j.abb.2014.12.022.

[14] Balashanmuga Priyan Rajamohan et al. 2025. Materials data science ontology (mds-onto): unifying domain knowledge in materials and applied data science. *Scientific Data*, 12, 1, 628. DOI: 10.1038/s41597-025-04938-5.

[15] Krishna Rajan. 2005. Materials informatics. *Materials Today*, 8, 10, 38–45. DOI: 10.1016/S1369-7021(05)71123-8.

[16] Z. Remeš, O. Babčenko, Vítězslav Jarý, and K. Beranová. 2024. Enhanced photoluminescence of plasma-treated recycled glass particles. *Nanomaterials*, 14. DOI: 10.3390/nano14131091.

[17] Francis W. M. R. Schwarze et al. 2024. Taming the production of bioluminescent wood using the white rot fungus desarmillaria tabescens. *Advanced Science*, 11. DOI: 10.1002/advs.202403215.

[18] David I Spivak. 2014. *Category theory for the sciences*. MIT press.

[19] David I Spivak, Tristan Giesa, Elizabeth Wood, and Markus J Buehler. 2011. Category theoretic analysis of hierarchical protein materials and social networks. *PloS one*, 6, 9, e23911. DOI: 10.1371/journal.pone.0023911.

[20] Aisha J Syed and James C. Anderson. 2021. Applications of bioluminescence in biotechnology and beyond. *Chemical Society reviews*. DOI: 10.1039/d0cs01492c.

[21] J. Trögl, G. Kuncová, and P. Kurán. 2010. Bioluminescence of pseudomonas fluorescens hk44 in the course of encapsulation into silica gel. effect of methanol. *Folia Microbiologica*, 55, 569–575. DOI: 10.1007/s12223-010-0091-9.

[22] Kewei Wang et al. 2024. Deep learning based inverse modeling for materials design: from microstructure and property to processing. In *2024 international conference on machine learning and applications (ICMLA)*. IEEE, 236–241. DOI: 10.1109/ICMLA61862.2024.00038.

[23] Yayun Wang, S. Naleway, and Bin Wang. 2020. Biological and bioinspired materials: structure leading to functional and mechanical performance. *Bioactive Materials*, 5, 745–757. DOI: 10.1016/j.bioactmat.2020.06.003.

[24] Tian Xie and Jeffrey C Grossman. 2018. Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties. *Physical review letters*, 120, 14, 145301. DOI: 10.1103/PhysRevLett.120.145301.

[25] Han Zhang, Runsheng Li, Junjiang Liu, Kaiyun Wang, Qian Weijian, Lei Shi, Liming Lei, Weifeng He, and Shengchuan Wu. 2024. State-of-art review on the process-structure-properties-performance linkage in wire arc additive manufacturing. *Virtual and Physical Prototyping*, 19, 1, e2390495. DOI: 10.1080/17452759.2024.2390495.

[26] Olga Zinovieva, Varvara Romanova, Ekaterina Dymnich, Aleksandr Zinoviev, and Ruslan Balokhonov. 2023. A review of computational approaches to the microstructure-informed mechanical modelling of metals produced by powder bed fusion additive manufacturing. *Materials*, 16, 19, 6459. DOI: 10.3390/ma16196459.

Cost-Sensitive Overview of Model Ensembling for Machine-Generated Text Detection

Abstract

In this paper, we compare different ways of combining encoder-based neural language models for machine-generated text detection from a cost-centric perspective. We tested five ensembling approaches: soft voting with separately fine-tuned models, training ensembles on disjoint training data subsets (for reducing training costs), snapshot ensembles, models with multiple classification heads, and merging models by averaging their weights. We evaluated each method based on how accurate they are (using macro F1 scores) and how much computing costs they require during training and prediction. Our findings show that while typical ensemble methods can boost accuracy, they come with high resource usage. In comparison, model merging achieves the highest accuracy (macro F1 of 0.832) without increasing the inference time. Although model merging requires higher training costs, this is often less of a concern where inference time costs outweigh the costs of model construction.

Keywords

ensemble methods, ai text detection, encoder models

1 Introduction

In recent years, improvements in generative models have made machine-generated text much more natural and realistic. Large volumes of AI-produced content are now being created and have even found their way into academic and scientific publications (Liang et al., 2024b). For example, previous studies indicate that as much as 6.5% to 16.9% of peer review submissions to AI conferences may be heavily influenced or rewritten by large language models, highlighting a growing dependence on LLMs in scholarly environments (Liang et al., 2024a). This trend is further reinforced by the widespread availability of powerful models, which can now be accessed by users with minimal technical expertise.

Because of this, there is a growing need for automated tools that can distinguish between human-written and machine-generated text. This need has been underscored by several recent shared tasks and competitions focused on this problem (Sarvazyan et al., 2023; Wang et al., 2024; Chamezopoulos et al., 2024; Wang et al., 2025).

Even with notable progress in the field, reliably separating machine-generated and human-written content remains difficult. Most state-of-the-art approaches for detecting AI-generated text rely on deep neural architectures, particularly transformer-based models fine-tuned on large annotated datasets (Sarvazyan et al., 2024; Marchitan et al., 2024). Recent research results have shown that the combination of multiple models can further improve the accuracy and robustness of machine-generated text detection (Gu and Meng, 2024; Kiss and Berend, 2025). However, these gains come at the expense of increased computational demands, both during training and inference, which can limit the practical adoption of such methods in large-scale, real-time applications.

To address these concerns, this paper provides a cost-centric overview of different strategies adapted for machine-generated text detection. For our experiments, we relied on a recent publicly available dataset: Task 1, Subtask A of the Workshop on Detecting AI-Generated Content at COLING 2025 (Wang et al., 2025), where the primary goal is to classify English text as human-written or AI-generated. We compare a range of approaches, including soft-voting ensembles, ensembles on disjoint training data subsets, snapshot ensembles, multiple classification heads model, and weight space model merging. By evaluating each method not only on predictive accuracy, but also on their training and inference costs, we aim to identify solutions that offer a practical balance between effectiveness and efficiency.

Our results show that, while traditional ensembling improves performance, it also results in significant computational overhead both during training and inference time. In contrast, model merging, where multiple models' weights are combined, achieves higher accuracy without increasing inference costs compared to a single model. Although training becomes more expensive, this is a one-time expense, making model merging an efficient and practical solution. Our source code is available at http://retracted.for.anonimity.

2 Related Work

Recent advancements in detecting machine-generated text have highlighted the efficacy of ensemble learning methods, particularly those that employ soft voting strategies (Gu and Meng, 2024; Kiss and Berend, 2025). In the SemEval-2024 Task 8 (Wang et al., 2024), Gu and Meng (2024) introduced a class-balanced soft voting system that fine-tuned transformer-based models (Gu and Meng, 2024). Their approach effectively addressed data imbalance and achieved state-of-the-art performance in multi-class classification tasks involving various text generators.

Due to the high computational cost of standard ensembles, which require training multiple independent models, several approaches have been proposed to reduce resource usage. One such method is the snapshot ensemble technique (Huang et al., 2017), which captures multiple model "snapshots" at different points during a single training run. Although this approach decreases training costs by avoiding the need to train separate models, it usually does not reach standard ensemble performance.

In the domain of multi-label and large-scale classification, Liang et al. (2025) proposed the Multi-Head Encoding (MHE) approach to tackle extreme label classification challenges(Liang et al., 2025). Their innovation involves substituting a conventional single classifier with multiple specialized classification heads, with each head responsible for a distinct subset of the overall label space. Their experiments across several extreme classification benchmarks underscore the potential of multiple classification head architectures in improving both efficiency and generalization, which aligns with our motivation for using models with multiple classification heads in the context of AI-generated text detection.

Information Society 2024, 7–11 October 2024, Ljubljana, Slovenia

3 Methodology

3.1 Base models

Dataset. The shared task dataset (Wang et al., 2025) we used provides more than 610,000 English texts for training and 261,000 for validation. Each entry is labeled either machine-generated by one of 40 different LLMs or human-written, yielding a 41-class classification task.

Single Model. For our ensemble experiments, we fine-tuned three separate DeBERTa-base models (He et al., 2021) using a 41-class setup. By adopting this multi-class approach, the models are better at capturing the diversity and differences among texts generated by different models, rather than treating all non-human written text as a single, heterogeneous category. Each DeBERTa-base model was trained independently using the AdamW optimizer (learning rate 2e-5, weight decay 0.01), with a batch size of 16, a 10% warmup schedule, and early stopping based on validation performance.

In our work, we intentionally chose not to rely on generative large language models (LLMs) as the core detection mechanism. This decision was motivated by practical considerations of cost efficiency and scalability. LLM-based solutions often have high computational costs and can be difficult to deploy at scale. As a result, our approach prioritizes methods that are both accurate and cost-effective.

3.2 Vanilla Ensemble

Traditional ensemble methods, also known as output-space ensembling, where several independently trained models contribute to the final prediction, are widely used to improve both accuracy and robustness in machine learning (Ganaie et al., 2022). It also works well for machine-generated text detection (Abburi et al., 2023; Gu and Meng, 2024; Kiss and Berend, 2025).

In our experiments, we used a standard soft voting approach, where each model was fine-tuned separately, and their predicted class probabilities were averaged to determine the final output. This typically led to better classification results than any single model.

However, this improvement comes with significant resource costs as all models must be trained and evaluated individually. For our study, we fine-tuned three models and combined their predictions using soft voting, then compared the ensemble's macro F1 score with the average score of the single models.

3.3 Disjoint Training Data (DTD) Ensembles

As we wanted to keep the total fine-tuning cost of the ensemble close to that of a single model, while still benefiting from the advantages of ensembling, we employed an ensemble method, where we trained the model on disjoint training data subsets. Knowing the large size of our available training data (approx. 611,000 examples), we divided the dataset into three disjoint subsets of equal size. Each subset was independently used to fine-tune a separate model, resulting in three models trained on disjoint portions of the dataset. The smaller training subsets shorten the individual training durations, making the approach more efficient overall.

After training, the three models were combined into an ensemble using soft voting. This setup uses the diversity of individually trained models while offering a cost-effective and time-efficient training alternative to standard ensemble strategies.

3.4 Snapshot Ensemble

For the snapshot ensemble approach (Huang et al., 2017), we generated several "snapshots" from a single model taken at different points throughout its fine-tuning. Unlike traditional ensemble methods, which require training multiple independent models from scratch, snapshot ensembling takes advantage of a cyclical learning rate schedule to encourage the model to converge to diverse regions of the parameter space within a single training run. This makes the method much more resource efficient, as it avoids the repeated cost of training from random initializations.

In our experiments, we used a cyclical learning rate schedule, as described in the original method. This approach periodically increases and decreases the learning rate during training. At the end of each cycle, we saved the model as a snapshot (see Figure 1). We decided to save six snapshots to achieve sufficient diversity.

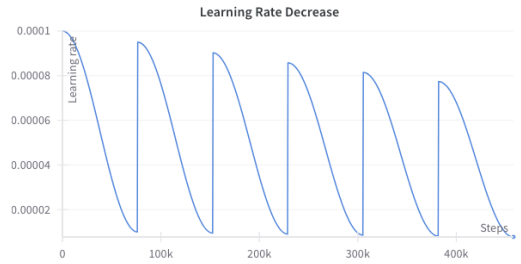


Figure 1: Learning rate schedule for snapshot ensemble

To construct the ensemble, we selected the three most diverse snapshots (based on validation performance) and combined their predictions via soft voting. This approach allowed us to obtain multiple strong models within a single fine-tuning process. However, since inference requires running each selected snapshot, the overall inference cost remains similar to that of standard ensembles. Although the improvement in predictive performance over a single model was modest, the snapshot ensemble presents a practical alternative when training resources are limited. Up to this point, the examined solutions mainly focused on reducing training costs, without lowering inference demands. From now on, we explore methods that aim to improve inference efficiency as well.

3.5 Model Merging

Model merging, also known as weight space ensembling or model soups (Wortsman et al., 2022), is a technique explored in machine learning as an alternative to traditional output space ensembling. Although assembling multiple independently trained models by averaging their predictions can improve performance, it comes at the cost of increased inference computation, as the output of several models must be calculated. Model merging addresses this by averaging the weights of several models to form a single, merged model. This merged model can match or exceed the performance of output space ensembles in many cases, but crucially, it incurs no additional inference costs compared to a single model.

The effectiveness of weight space merging is supported by two fundamental concepts: mode connectivity (Frankle et al., 2020) and sparsity. Mode connectivity suggests that independently trained neural networks, especially those derived from the same base model, can reside in regions of the loss landscape connected by paths of relatively constant performance. In this work, we applied a model merging approach using three distinct models.

After merging, we experimented with two configurations. First, we attached the original three classification heads from the source models to the merged encoder, resulting in three separate models sharing the same backbone but differing in their classification heads. Second, we also fine-tuned new classification heads on top of the merged encoder, which model variant we refer to as Model Merging (FT). This setup kept the efficiency of a single-model inference pipeline while allowing the new head to adapt to the merged weights.

3.6 Models with Multiple Classification Heads

In this architecture, multiple classification heads are attached to a shared backbone model (Chang et al., 2023; Lee et al., 2015). Each head is independently fine-tuned, allowing them to specialize and capture different aspects of the data distribution. During inference, the outputs of these heads are aggregated, typically by voting to produce the final prediction. This approach provides some of the benefits of ensembling, while maintaining a compact model structure and reducing deployment complexity. In our experiments, we implemented variants with 3 and 5 classification heads (referred to as MCH models). Each head independently produced a classification output, and the final label was determined by soft voting over them. The main advantage of the multiple classification headed (MCH) architecture is its low inference cost, equivalent to a single model since all heads share the same encoder. The only additional overhead comes from running the multiple classification heads, which is negligible compared to vanilla multi-model ensembles.

4 Results

We compared several ensembling approaches, focusing on both their Macro-F1 scores and their training and inference costs. The vanilla ensemble and the fine-tuned model merging approach are averaged over three independent runs, the rest are based on a single evaluation.

		Costs	
Experiment	Macro F1	Training	Inference
3 Classifier Heads	0.805	Low	Low
Single model	0.806	Low	Low
Snapshot ensemble	0.810	Low	High
DTD Ensemble	0.814	Low	High
5 Classifier Heads	0.817	Low	Low
Vanilla Ensemble	0.826	High	High
Model Merging (FT)	0.827	Highest	Low
Model Merging	**0.832**	High	Low

Table 1: Summary of results on test set with costs.

Our experimental results are summarized in Table 1, which compares the predictive performance (macro F1 score) and computational costs (training and inference) of each ensemble method. The Vanilla Ensemble achieved an improved macro F1 score of 0.826, which was a substantial increase over the single model. However, this performance gain incurred significantly higher costs for both training and inference.

The DTD ensemble approach resulted in a slightly lower macro F1 compared to the vanilla ensemble. However, it significantly reduced training costs, since each individual model was trained on only one-third of the training data, although inference costs remained high due to querying multiple models simultaneously. The Snapshot Ensemble offered moderate computational efficiency, yet resulted in a small improvement of macro F1 over the single model, while still retaining high inference costs due to combining predictions from multiple snapshots. MCH models demonstrated varying performance depending on the number of heads. The 3-head variant was matching single-model costs but did not outperform it. The 5-head variant improved F1 to 0.817 while maintaining low inference costs and moderate training expense. Model merging is the most promising approach. Directly merging pre-trained model weights showed the best overall performance, maintaining low inference cost equivalent to a single model. Further fine-tuning after merging increased training costs, but unlike ensembles, the merged model requires only a single model at inference. This greatly reduces inference time and resource usage. Considering both computational efficiency and classification performance, the model merging strategy without additional fine-tuning provided the best balance overall.

5 Conclusion

We explored multiple ensemble strategies for detecting machine-generated text, highlighting a cost-sensitive approach that balances computational efficiency with predictive performance. Our evaluation demonstrated that although ensemble methods consistently improve predictive accuracy, they typically require significantly higher training or inference resources, making them less practical for real-world scenarios where resources are limited. In particular, model merging was the most effective option among all the evaluated methods. Not only did it achieve the highest macro F1 score (0.832), but it did so without inflating inference costs, maintaining an efficiency comparable to running just a single model. This approach effectively captures the strengths of multiple models while avoiding the computational overhead commonly associated with traditional ensembling methods.

Overall, our results show that the use of ensembles does not always have to be very costly. By carefully choosing methods such as model merging, it is possible to get most of the performance benefits of ensembling without using too many resources for predictions, especially during inference time. This can make these methods a good fit for building fast and scalable systems to detect AI-generated text.

References

Harika Abburi, Kalyani Roy, Michael Suesserman, Nirmala Pudota, Balaji Veeramani, Edward Bowen, and Sanmitra Bhattacharya. 2023. A Simple yet Efficient Ensemble Approach for AI-generated Text Detection. arXiv:2311.03084 [cs.CL] https://arxiv.org/abs/2311.03084

Savvas Chamezopoulos, Drahomira Herrmannova, Anita De Waard, Drahomira Herrmannova, Domenic Rosati, and Yury Kashnitsky. 2024. Overview of the DagPap24 Shared Task on Detecting Automatically Generated Scientific Paper. In *Proceedings of the Fourth Workshop on Scholarly Document Processing (SDP 2024)*, Tirthankar Ghosal, Amanpreet Singh, Anita Waard, Philipp Mayr, Aakanksha Naik, Orion Weller, Yoonjoo Lee, Shannon Shen, and Yanxia Qin (Eds.). Association for Computational Linguistics, Bangkok, Thailand, 7–11. https://aclanthology.org/2024.sdp-1.2

Haw-Shiuan Chang, Ruei-Yao Sun, Kathryn Ricci, and Andrew McCallum. 2023. Multi-CLS BERT: An Efficient Alternative to Traditional Ensembling. In *Proceedings of the 61st Annual*

Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (Eds.). Association for Computational Linguistics, Toronto, Canada, 821–854. https://doi.org/10.18653/v1/2023.acl-long.48

Jonathan Frankle, Gintare Karolina Dziugaite, Daniel M. Roy, and Michael Carbin. 2020. Linear mode connectivity and the lottery ticket hypothesis. In *Proceedings of the 37th International Conference on Machine Learning (ICML'20)*. JMLR.org, Article 305, 11 pages.

M.A. Ganaie, Minghui Hu, A.K. Malik, M. Tanveer, and P.N. Suganthan. 2022. Ensemble deep learning: A review. *Engineering Applications of Artificial Intelligence* 115 (Oct. 2022), 105151. https://doi.org/10.1016/j.engappai.2022.105151

Renhua Gu and Xiangfeng Meng. 2024. AISPACE at SemEval-2024 task 8: A Class-balanced Soft-voting System for Detecting Multi-generator Machine-generated Text. In *Proceedings of the 18th International Workshop on Semantic Evaluation (SemEval-2024)*, Atul Kr. Ojha, A. Seza Doğruöz, Harish Tayyar Madabushi, Giovanni Da San Martino, Sara Rosenthal, and Aiala Rosá (Eds.). Association for Computational Linguistics, Mexico City, Mexico, 1476–1481. https://doi.org/10.18653/v1/2024.semeval-1.212

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. DeBERTa: Decoding-enhanced BERT with Disentangled Attention. In *International Conference on Learning Representations*. https://openreview.net/forum?id=XPZIaotutsD

Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E. Hopcroft, and Kilian Q. Weinberger. 2017. Snapshot Ensembles: Train 1, Get M for Free. In *International Conference on Learning Representations*. https://openreview.net/forum?id=BJYwwY9ll

Mihaly Kiss and Gábor Berend. 2025. SzegedAI at GenAI Detection Task 1: Beyond Binary - Soft-Voting Multi-Class Classification for Binary Machine-Generated Text Detection Across Diverse Language Models. In *Proceedings of the 1stWorkshop on GenAI Content Detection (GenAIDetect)*, Firoj Alam, Preslav Nakov, Nizar Habash, Iryna Gurevych, Shammur Chowdhury, Artem Shelmanov, Yuxia Wang, Ekaterina Artemova, Mucahid Kutlu, and George Mikros (Eds.). International Conference on Computational Linguistics, Abu Dhabi, UAE, 166–172. https://aclanthology.org/2025.genaidetect-1.15/

Stefan Lee, Senthil Purushwalkam, Michael Cogswell, David J. Crandall, and Dhruv Batra. 2015. Why M Heads are Better than One: Training a Diverse Ensemble of Deep Networks. *CoRR* abs/1511.06314 (2015). arXiv:1511.06314 http://arxiv.org/abs/1511.06314

Daojun Liang, Haixia Zhang, Dongfeng Yuan, and Minggao Zhang. 2025. Multi-Head Encoding for Extreme Label Classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 47, 3 (March 2025), 2199–2211. https://doi.org/10.1109/tpami.2024.3522298

Weixin Liang, Zachary Izzo, Yaohui Zhang, Haley Lepp, Hancheng Cao, Xuandong Zhao, Lingjiao Chen, Haotian Ye, Sheng Liu, Zhi Huang, Daniel Mcfarland, and James Y. Zou. 2024a. Monitoring AI-Modified Content at Scale: A Case Study on the Impact of ChatGPT on AI Conference Peer Reviews. In *Proceedings of the 41st International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 235)*, Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (Eds.). PMLR, 29575–29620. https://proceedings.mlr.press/v235/liang24b.html

Weixin Liang, Yaohui Zhang, Zhengxuan Wu, Haley Lepp, Wenlong Ji, Xuandong Zhao, Hancheng Cao, Sheng Liu, Siyu He, Zhi Huang, Diyi Yang, Christopher Potts, Christopher D Manning, and James Y. Zou. 2024b. Mapping the Increasing Use of LLMs in Scientific Papers. In *First Conference on Language Modeling*. https://openreview.net/forum?id=YX7QnhxESU

Teodor-george Marchitan, Claudiu Creanga, and Liviu P. Dinu. 2024. Team Unibuc - NLP at SemEval-2024 Task 8: Transformer and Hybrid Deep Learning Based Models for Machine-Generated Text Detection. In *Proceedings of the 18th International Workshop on Semantic Evaluation (SemEval-2024)*, Atul Kr. Ojha, A. Seza Doğruöz, Harish Tayyar Madabushi, Giovanni Da San Martino, Sara Rosenthal, and Aiala Rosá (Eds.). Association for Computational Linguistics, Mexico City, Mexico, 403–411. https://doi.org/10.18653/v1/2024.semeval-1.63

Areg Mikael Sarvazyan, José Ángel González, and Marc Franco-salvador. 2024. Genaios at SemEval-2024 Task 8: Detecting Machine-Generated Text by Mixing Language Model Probabilistic Features. In *Proceedings of the 18th International Workshop on Semantic Evaluation (SemEval-2024)*, Atul Kr. Ojha, A. Seza Doğruöz, Harish Tayyar Madabushi, Giovanni Da San Martino, Sara Rosenthal, and Aiala Rosá (Eds.). Association for Computational Linguistics, Mexico City, Mexico, 101–107. https://doi.org/10.18653/v1/2024.semeval-1.17

Areg Mikael Sarvazyan, José Ángel González, Marc Franco-Salvador, Francisco Rangel, Berta Chulvi, and Paolo Rosso. 2023. Overview of AuTexTification at IberLEF 2023: Detection and Attribution of Machine-Generated Text in Multiple Domains. *Proces. del Leng. Natural* 71 (2023), 275–288. http://journal.sepln.org/sepln/ojs/ojs/index.php/pln/article/view/6559

Yuxia Wang, Jonibek Mansurov, Petar Ivanov, Jinyan Su, Artem Shelmanov, Akim Tsvigun, Osama Mohammed Afzal, Tarek Mahmoud, Giovanni Puccetti, and Thomas Arnold. 2024. SemEval-2024 Task 8: Multidomain, Multimodel and Multilingual Machine-Generated Text Detection. In *Proceedings of the 18th International Workshop on Semantic Evaluation (SemEval-2024)*, Atul Kr. Ojha, A. Seza Doğruöz, Harish Tayyar Madabushi, Giovanni Da San Martino, Sara Rosenthal, and Aiala Rosá (Eds.). Association for Computational Linguistics, Mexico City, Mexico, 2057–2079. https://doi.org/10.18653/v1/2024.semeval-1.279

Yuxia Wang, Artem Shelmanov, Jonibek Mansurov, Akim Tsvigun, Vladislav Mikhailov, Rui Xing, Zhuohan Xie, Jiahui Geng, Giovanni Puccetti, Ekaterina Artemova, Jinyan Su, Minh Ngoc Ta, Mervat Abassy, Kareem Elozeiri, Saad El Dine Ahmed, Maiya Goloburda, Tarek Mahmoud, Raj Vardhan Tomar, Alexander Aziz, Nurkhan Laiyk, Osama Mohammed Afzal, Ryuto Koike, Masahiro Kaneko, Alham Fikri Aji, Nizar Habash, Iryna Gurevych, and Preslav Nakov. 2025. GenAI Content Detection Task 1: English and Multilingual Machine-generated Text Detection: AI vs. Human. In *Proceedings of the 1st Workshop on GenAI Content Detection (GenAIDetect)*. International Conference on Computational Linguistics, Abu Dhabi, UAE.

Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S. Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. 2022. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. arXiv:2203.05482 [cs.LG] https://arxiv.org/abs/2203.05482

Hybrid Reinforcement Learning Enhanced Genetic Algorithm for the Capacitated Vehicle Routing Problem with Split Deliveries and Heterogeneous Fleet

Ahmed Dabbous*
Luleå University of Technology, Department of Computer Science, Electrical and Space Engineering, Embedded Intelligent Systems Lab
Luleå, Sweden
ahmdab-1@student.ltu.se

András Bóta†
Luleå University of Technology, Department of Computer Science, Electrical and Space Engineering, Embedded Intelligent Systems Lab
Luleå, Sweden
andras.bota@ltu.se

Abstract

Vehicle routing is a classical field of combinatorial optimization with a multitude of real-life applications. Recent advances in machine learning, specifically reinforcement learning, promise improved performance compared to traditional heuristic approaches. Here, we propose a hybrid reinforcement learning enhanced genetic algorithm to solve a less common VRP variant: the Capacitated Vehicle Routing Problem with Split Deliveries and Heterogeneous Fleet. We test the performance of the approach on a custom dataset consisting of the road network and agricultural buildings on the island of Gotland. Our results show, that the reinforcement learning algorithm is able to improve the performance of the genetic algorithm by guiding operator selection.

Keywords

CVRP, split deliveries, heterogeneous fleet, reinforcement learning, genetic algorithm

1 Introduction

The vehicle routing problem (VRP) is a traditional combinatorial optimization problem, with the goal of finding the optimal set of routes for a fleet of vehicles visiting a set of locations. The problem has received widespread attention from the research community since its first appearance. Further contributing to its popularity, VRP has many applications in practice and has inspired many variants [6]. A common variant, the capacitated vehicle routing problem (CVRP) introduces a capacity constraint on the participating vehicles.

Here, we focus on two less-known variants of CVRP: 1. CVRP with Split Deliveries, where multiple vehicles may serve a single customer, and 2. CVRP with Heterogeneous Fleet, where a fleet of vehicles with different capacities may serve customers. Both of these variants model realistic situations, and as such, they are especially relevant in industry applications [14, 5].

Genetic algorithms are powerful metaheuristics frequently used to solve complex optimization tasks [1], including VRP and its variants [12]. Genetic algorithms work with a population of solutions, referred to as chromosomes. In each iteration of the algorithm, a set of genetic operators is applied to the chromosomes, to generate the next set of solutions. Both the operators and the algorithm itself may have a large set of hyperparameters. Finding the one that works the best for a specific problem can be difficult.

Reinforcement learning (also called as approximate dynamic programming) is an interdisciplinary branch of machine learning, where agents take actions in an environment aiming to maximize a reward function. Recent advances in reinforcement learning, such as deep Q learning, gave birth to applications in combinatorial optimization, including VRP [3].

In this proof-of-concept paper, we propose a reinforcement learning enhanced genetic algorithm to solve the Capacitated Vehicle Routing Problem with Split Deliveries and Heterogeneous Fleet. Our solution uses a novel chromosome representation, and introduces a pair of operators on them. We use reinforcement learning to dynamically learn genetic operator choice. We demonstrate the performance of our method on a custom dataset inspired by agricultural product collection on the island of Gotland.

2 Problem formulation

Let a complete graph $G = (V, D)$ with node set $V = \{0, 1, \ldots, n\}$, where node 0 denotes the depot and nodes $1, \ldots, n$ represent customers. Let d_{ij} denote the travel distance between nodes i and j. Each customer $j \in \{1, \ldots, n\}$ has a non-negative demand q_j. Let K be the set of vehicles, where each vehicle $k \in K$ has a non-negative capacity u_k.

The goal is to design a set of routes, each assigned to exactly one vehicle, such that every route starts and ends at the depot, and the total demand served on a route does not exceed the capacity u_k of its assigned vehicle. Multiple vehicles are allowed to visit the same node. The objective is to minimize the total distance, defined as the sum of all travel costs over all traversed pairs of nodes.

3 Method

Our proposed solution method is a genetic algorithm enhanced by a reinforcement learning algorithm, which aims to optimize genetic operator selection.

3.1 Genetic algorithm

Genetic algorithms (GA) are powerful metaheuristics inspired by the process of natural selection [1]. They work with a set (or population) of P solutions, also called chromosomes. Each chromosome represents a solution to the problem. The population is updated in an iterative manner, with the t-th iteration being referred to as the t-th generation. The first generation

*Both authors contributed equally to this research.
†Corresponding author.

Information Society 2025, 6–10 October 2025, Ljubljana, Slovenia

of solutions can be randomized, or be an output of a different method. Genetic algorithms have three main operators: 1. Selection, 2. Crossover, and 3. Mutation. The selection operator determines which solutions in the $t-1$-th generation progress to the t-th generation based on the fitness value. The fitness value evaluates the goodness of the solution. The crossover operator recombines the chromosomes of two (or potentially more) parent solutions to produce offspring solutions. The crossover should increase the general quality of the solutions in the population. The usually unary mutation operator introduces genetic diversity within individuals to prevent premature convergence. The mutation operation is intended to explore the local neighborhood of the current solution. In the rest of this section we describe the proposed algorithm.

3.1.1 Chromosome representation. We propose a new chromosome representation adapted to the specifics of heterogeneous fleet and split-delivery problems. Our solution represents both the vehicle assignment and the order of customers in the same chromosome, while allowing customers to be visited by multiple vehicles. The proposed representation is still fundamentally a permutation with repetition.

Two different sets of symbols are used to represent vehicles and customers. Here, we use letters to represent vehicles and numbers to represent customers. Furthermore, to allow for split deliveries, we split up the demand of each customer in the following way: if customer j had a demand of q_j, then the number corresponding to j will appear q_j times in the chromosome. Letters representing vehicles are always followed by at most a u_k amount of numbers representing customers, where u_k represents the capacity of the vehicle k.

On Figure 1, a chromosome with three unique customers and four vehicles is shown. Each of the three unique customers is repeated in the permutation according to their demands. The vehicles in the permutation are represented by letters. Each color indicates a unique route belonging to one of the vehicles.

Figure 1: A chromosome with $n_{\text{customers}} = 3$ unique customers and $n_{\text{vehicles}} = 4$ vehicles. Each color represents a unique vehicle route.

If we consider customer 3, then based on the representation, the customer has 2 units of demand. The delivery of their demands will be split between vehicles B and A, meaning this customer will be serviced twice in two separate deliveries.

3.1.2 Selection. We evaluate the fitness of the chromosomes by calculating the sum of all route lengths. In this work, we have a fixed amount of vehicles and assume all of them have the same operating costs. The vehicles may have different capacities.

The algorithm always works with a population of 100 chromosomes. We apply a limited elitism strategy with the 7 best chromosomes automatically copied over from the previous generation. The rest of the population is filled up with the best 93 solutions resulting after performing the crossover and mutation operators.

3.1.3 Crossover. We present a modified version of the standard two-point permutation crossover operator (OX1) [13] adapted to our chromosome representation. OX1 first selects two cut points on the parent chromosomes and copies the segment between these points from the first parent into the child. The remaining positions in the child are filled with the elements from the second parent in the order they appear, with the exception of elements already present. A second offspring is created by copying over the segment between the cut points from the second parent, with the remaining position filled up from the first parent. Since our chromosome representation allows for duplicates, repeated instances of the same element are treated as identical to each other. This ensures that the offsprings are valid permutations, preserving the relative order of elements from their parents.

We derive two crossover operators from OX1, one for the customer orders, and one for the vehicle assignment. For customer orders (COX1) we first save the positions of vehicles in the second chromosome. Then we apply the OX1 operator only on the customer orders. Finally, in the resulting new solution, we restore the position of the vehicles in the exact same order they were in the second chromosome.

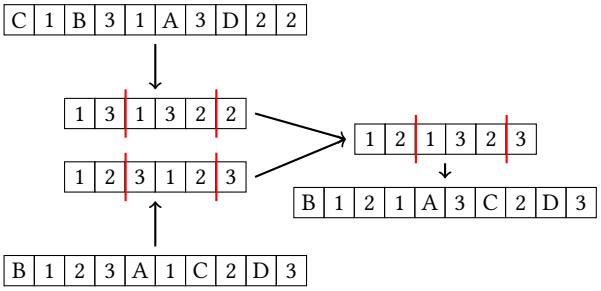


Figure 2: An example of applying the COX1 operator to two parent chromosomes in our defined representation. For brevity, only one offspring permutation is shown but a second offspring is also generated by swapping the parents.

Figure 2 provides a visual example of how the COX1 operator is applied to two parent chromosomes. The operator extracts the customer number elements from the permutation maintaining their ordering. This step generates two new permutations consisting solely of customer numbers. Two point OX1 crossover is applied on these new permutations to generate two new orderings of customers. The new customer orderings are reinserted into the parent permutations in the old customers' positions without altering the vehicle positions within the parent permutations.

The second operator works in a similar way, just for the vehicle assignment (VOX1). We first save the order of the customers from the first chromosome. Then we apply the OX1 operator only on the vehicles. Then, we insert the new vehicle assignment into the customer ordering sequentially, always filling up the vehicles to their maximum capacity.

3.1.4 Mutation. We use the inversion mutation operator [1]. It works by selecting two positions at random within the chromosome and reversing the order of all elements between these two positions. The operator preserves the permutation with repetition property of our chromosome representation.

3.1.5 Algorithm overview. Algorithm 1 summarizes the genetic algorithm.

3.2 Reinforcement learning

Reinforcement learning (RL) is a machine learning paradigm where an agent learns to make decisions by interacting with an

Algorithm 1 Genetic Algorithm (single-objective, generational, elitist)

Input: population size $|P|$, elite size m_{elites}, mutation rate p_m, crossover ratio p_c, generations T
$P \leftarrow \text{InitializePopulation}(P)$
$\text{EvaluateFitness}(P)$
$t \leftarrow 0$
while $t < T$ **do**
  $P^\star \leftarrow \text{SelectElites}(P, m_{\text{elites}})$ ▷ copy best E unchanged
  $M \leftarrow \emptyset$ ▷ mating pool
  $C^\star \leftarrow \emptyset$ ▷ Selected Crossover Offspring
  while $|M| < p_c^2|P|^2$ **do**
    $p_1 \leftarrow \text{SelectParent}(P)$ ▷ Roulette Wheel Selection
    $p_2 \leftarrow \text{SelectParent}(P)$
    $(p_1', p_2') \leftarrow \text{Crossover}(p_1, p_2)$
    $\text{Append}(M, p_1')$
    $\text{Append}(M, p_2')$
  end while
  $C^\star \leftarrow \text{SelectTop}(M, |P| - m_{\text{elites}})$
  for c_i `in` $C^\star$ **do**
    $c_i \leftarrow \text{Mutate}(c_i, p_m)$ ▷ Inversion
    $P^\star \leftarrow c_i$
  end for
  $P \leftarrow P^\star$
  $\text{EvaluateFitness}(P)$
  $t \leftarrow t + 1$
end while
return $\text{Best}(P)$

environment to maximize a reward function. The agent observes a state, takes an action to move to a new state, and potentially receives a reward, with the overall goal of finding a policy, mapping from states to actions, that maximizes expected discounted rewards. RL can be combined with genetic algorithms to create hybrid optimization methods combining the strengths of both paradigms. [8, 10, 11]

In this paper, we use Q-learning [4], a model-free value-based approach to enhance our genetic algorithm, inspired by the works of [8, 11].

The state space is defined by a pair of categorical values, which categorize the percentage change in the fitness value of the population's best member, and the diversity within the entire population [7] (the number of unique individuals in the population divided by the total population size). The best fitness percentage changes between generations are discretized into one of six categories ranging from high change to no change at all. Likewise, the diversity ratio is binned into one of five categories ranging from very high to very low diversity.

The action space is composed of 8 possible actions where each action is a triplet of 'C' or 'V' markers indicating whether a VOX1 or COX1 crossover operator should be used. For example, the action triplet 'CVV' indicates that the COX1 operator is to be used on the first generation, followed by the VOX1 operator on the second and third generations. Given that these action triplets apply to three consecutive generations, all model training steps including sampling the actions, states, and rewards as well as updating the Q table are only applied on every third generation.

The Q-learning agent aims to learn the optimal state-action value (Q-value) for each state-action pair. The Q-value measures the quality of state action pairs by estimating the expected sum of discounted future rewards. The policy chooses the action with the maximum Q-value in each given state. During the training process, a Q-table containing the Q-value of each state-action pair is maintained and updated according to equation (1) below.

$$Q_{\text{new}}(s_t, a_t) = Q_{\text{old}}(s_t, a_t) + \alpha(r_t + \gamma \operatorname*{argmax}_a Q(s_{t+1}, a) - Q_{\text{old}}(s_t, a_t)) \quad (1)$$

The learning rate α indicates the rate at which the agent learns a new behaviour. The discount rate γ indicates the relative significance of future rewards relative to the immediate reward r_t.

We employ a standard epsilon-greedy policy [4], where with a probability ϵ, a random action will be chosen in the given state. This allows for an exploration of the action space. With a probability of $1-\epsilon$, the action $\text{argmax}_a\, Q(s_t, a)$ with the maximal Q-value given the current state in the Q-table is chosen. In our setup, the ϵ-value is initialized to 1 and is exponentially decayed with every generation at a decay rate 0.0005 until it plateaus at $\epsilon_{\text{min}} = 0.1$. This allows for exploration of the action space in the beginning when the Q-values are not well estimated.

4 Data

We measure the performance of our proposed algorithm on a novel custom dataset based on open access data on agricultural buildings on the island of Gotland in Sweden. The dataset was originally inspired by livestock collection problem [9] but we use it in a different context in this paper. All data were downloaded from OpenStreetMaps https://www.openstreetmap.org/. The road network is the road network of Gotland. A $n = 250$ number of delivery locations were sampled from all locations marked as agricultural in the OpenStreetMaps database. Pairwise distances between customer nodes were computed with the OpenSourceStreetMap library https://www.openstreetmap.org while the customer-depot distances were computed with the Geopy open source library https://geopy.readthedocs.io/en/stable/. The demands for each customer were sampled from a uniform distribution in the interval $[1, 10]$ while the vehicle capacities were likewise sampled from a uniform distribution in the interval $[100, 200]$. The depot position was chosen to be the mean longitude and latitude of the customer positions.

5 Results

In this proof-of-concept work, our main goal is to measure the performance improvement reinforcement learning can provide for the genetic algorithm. We also conduct a limited amount of hyperparameter optimization for both the reinforcement learning and genetic algorithms.

Starting with the latter, we found that the following values provided the best fitness population in our experiments:

- population size $|P| = 100$
- crossover rate $p_c = 1$
- mutation rate $p_m = 0.10$
- number of elites $m_{\text{elites}} = 7$
- learning rate $\alpha = 0.3$
- discount rate $\gamma = 0.7$
- ϵ-greedy strategy as described in section 3.2

To measure the performance improvement of the RL enhanced algorithm, we ran experiments on the Gotland dataset. To provide a baseline, we used the genetic algorithm described in section

3.1 without any RL component, and where the two operators VOX1 and COX1 were alternating in subsequent generations. We compared the performance of the baseline algorithm with the RL enhanced version, which aims to learn the optimal operator triplet as described in section 3.2. Figure 3 shows the fitness value of the populations of both variants of the algorithm for the first 10000 generations.

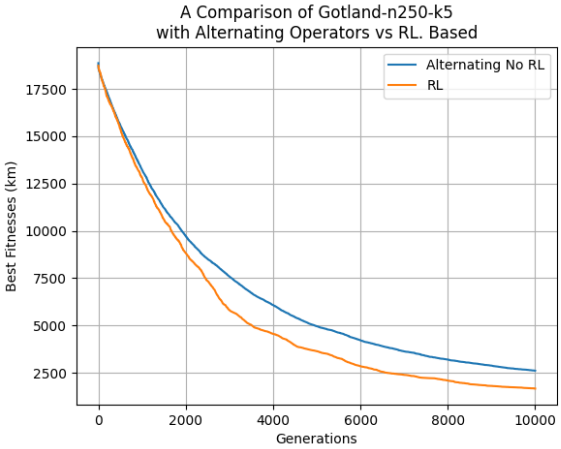


Figure 3: Fitness values for both baseline and RL enhanced algorithms on the Gotland dataset for the first 10000 generations.

Our results show, that the RL-enhanced algorithm maintains an advantage in allowing the agent to decide the genetic crossover operator triplets throughout the experiment. Up to the $1000th$ generation, both approaches follow approximately equal trajectories, attributed to the epsilon value $\epsilon > 0.6$ still being relatively high. This leaves the agent in an exploratory phase where it is more likely to choose random actions from the action space. During the evolution process, the RL agent picks up an advantage as it begins to learn the crossover triplet selection. At the end, the best fitness values of the baseline and RL-enhanced algorithms were 2616 km and 1671 km respectively, showing an improvement of 56.5 percent.

Further enhancing the usefulness of the algorithm the runtimes for the baseline and RL-enhanced algorithms were 45.1 minutes and 42.5 minutes respectively, showing only a slight increase for the enhanced variant. This is due to the computational overhead introduced by the RL algorithm, but we believe it is compensated by the improvement in the quality of the final solution.

6 Conclusions and future work

In this paper, we proposed a reinforcement learning enhanced genetic algorithm to solve the Capacitated Vehicle Routing Problem with Split Deliveries and Heterogeneous Fleet, and measured its performance on a custom dataset inspired by agricultural product collection on the island of Gotland. Our initial results show a significant, 56.5 percent performance boost of the RL-enhanced algorithm compared to a baseline genetic algorithm, that does not have a learning component. Furthermore, due to the lightweight nature of the learning algorithm, the performance boost comes with only a slight computational overhead.

We plan to perform a more thorough testing of the algorithm in the future. First, we plan to perform hyperparameter optimization for both the reinforcement learning and genetic algorithms to find the values providing the best performance. We will also apply our algorithm on other datasets, such as the Augerat benchmark [2].

We also plan to extend both parts of the algorithm. So far we have only experimented with the two genetic operators described in section 3.1. We plan to conduct experiments with both traditional operators, and potentially new ones. The reinforcement learning algorithm can be improved in multiple ways. It can be used to dynamically optimize the hyperparameters of the genetic algorithm during the evolution process, such as the number of elites and, the crossover and mutation rates. We can also experiment with more complex operator strategies, possibly adding more operators to the mix. Finally, in the context of deep Q learning, neural networks can be used to learn the hyperparameters of the operators, such as the cut points of OX1 or even a binary mask similar to the one used in the PBX operator [13]. These modification will likely further enhance the performance of our algorithm.

Acknowledgements

This work was supported by Sweden's Innovation Agency, grant number 2023-02595, which is gratefully acknowledged.

References

[1] Bushra Alhijawi and Arafat Awajan. 2024. Genetic algorithms: theory, genetic operators, solutions, and applications. *Evolutionary Intelligence*, 17, 3, 1245–1256.

[2] Philippe Augerat, D Naddef, JM Belenguer, E Benavent, A Corberan, and Giovanni Rinaldi. 1995. Computational results with a branch and cut code for the capacitated vehicle routing problem.

[3] Ahmad Bdeir, Simon Boeder, Tim Dernedde, Kirill Tkachuk, Jonas K Falkner, and Lars Schmidt-Thieme. 2021. Rp-dqn: an application of q-learning to vehicle routing problems. In *German conference on artificial intelligence (Künstliche Intelligenz)*. Springer, 3–16.

[4] Jesse Clifton and Eric Laber. 2020. Q-learning: theory and applications. *Annual Review of Statistics and Its Application*, 7, 1, 279–301.

[5] Simos Efthymiadis, Nikolaos Liapis, and George Nenes. 2023. Solving a heterogeneous fleet multi-compartment vehicle routing problem: a case study. *International Journal of Systems Science: Operations & Logistics*, 10, 1, 2190474.

[6] Burak Eksioglu, Arif Volkan Vural, and Arnold Reisman. 2009. The vehicle routing problem: a taxonomic review. *Computers & Industrial Engineering*, 57, 4, 1472–1483.

[7] Nguyen Thi Hien and Nguyen Xuan Hoai. 2006. A brief overview of population diversity measures in genetic programming. In *Proc. 3rd asian-pacific workshop on genetic programming, hanoi, vietnam*, 128–139.

[8] Eda Köksal Ahmed, Zengxiang Li, Bharadwaj Veeravalli, and Shen Ren. 2021. Reinforcement learning-enabled genetic algorithm for school bus scheduling. *Journal of Intelligent Transportation Systems*, 26, 3, 269–283. DOI: https://doi.org/10.1080/15472450.2020.1852082.

[9] Johan Oppen and Arne Løkketangen. 2008. A tabu search approach for the livestock collection problem. *Computers & Operations Research*, 35, 10, 3213–3229.

[10] Jose Quevedo, Marwan Abdelatti, Farhad Imani, and Manbir Sodhi. 2021. Using reinforcement learning for tuning genetic algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion* (GECCO '21). Association for Computing Machinery, Lille, France, 1503–1507. ISBN: 9781450383516. DOI: 10.1145/3449726.3463203.

[11] Jose Quevedo, Marwan Abdelatti, Farhad Imani, and Manbir Sodhi. 2021. Using reinforcement learning for tuning genetic algorithms. In *Proceedings of the genetic and evolutionary computation conference companion*, 1503–1507.

[12] A Serdar Tasan and Mitsuo Gen. 2012. A genetic algorithm based approach to vehicle routing problem with simultaneous pick-up and deliveries. *Computers & Industrial Engineering*, 62, 3, 755–761.

[13] Anant J Umbarkar and Pranali D Sheth. 2015. Crossover operators in genetic algorithms: a review. *ICTACT journal on soft computing*, 6, 1.

[14] Jiao Zhao, Hongxia Dong, and Ning Wang. 2023. Green split multiple-commodity pickup and delivery vehicle routing problem. *Computers & Operations Research*, 159, 106318.

Empiric results on the achievable performance gains by the inclusion of instance-specific information in a scheduling optimizer

Mate Hegyhati
hegyhati.work@gmail.com
University of Sopron
Sopron, Hungary
University of Pannonia
Veszprem, Hungary
University of Applied Sciences Wiener Neustadt
Wiener Neustadt, Austria

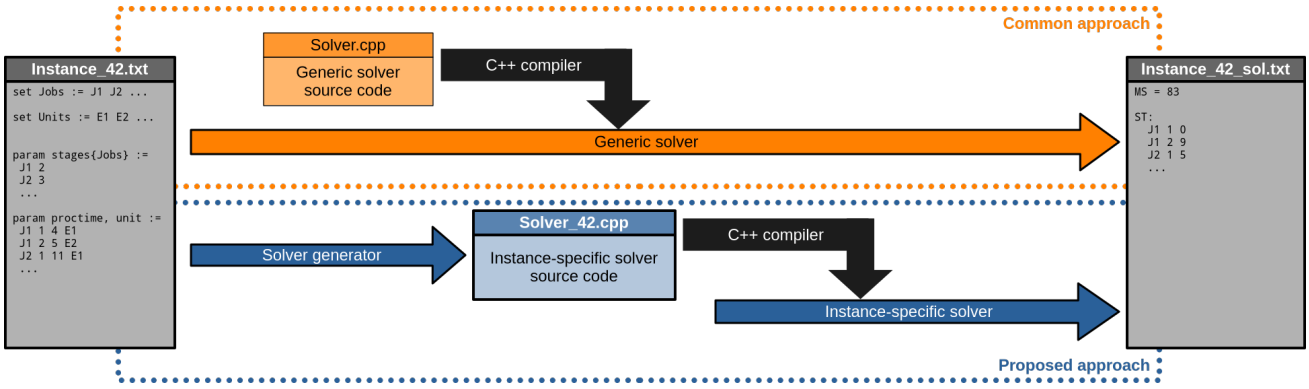


Figure 1: Proposed workflow with instance-specific binaries

Abstract

It is common in computational combinatorics, graph theory, and combinatorial optimization to develop custom algorithms. These algorithms usually address a whole problem class, i.e., an infinitely large set of instances. As a result, their implementation is also generic, and instance-specific information is only available in the execution phase of the solver, not at its compilation. There are reasonable arguments, why earlier inclusion of instance-specific information may result in better performance, i.e., compiling an instance-specific solver rather than relying on a generic binary. The goal of this work is to investigate the achievable performance gains on the C++ implementation of an S-graph based makespan-minimization algorithm that targets job-shop problems. Results show that instance-specific binaries can significantly outperform the generic solver, reducing the computational needs by 70-80%.

Keywords

optimization, scheduling, instance-specific implementation

1 Introduction

It is rather natural to have a clear distinction and separation in our mind between a computer program and the data it uses in a certain scenario. This is true for almost all software, let it be a spreadsheet application, restaurant menu page on a website, training advisor on a sport watch, or a simulation/optimization software. In each of the mentioned cases, the software executes generic logic on the data that is read from a database, file or received over the network, etc. The software itself is not specific to each use-case, which is obviously a big benefit in reusability, saving costs, and a reason, why the industry developed in this direction. Moreover, this phenomenon usually extends to multiple layers. For example: logic on a website is usually written in JavaScript, which is a (mostly) interpreted language, i.e., an even more generic code (JavaScript engine) is run, that reads the website code, whose execution reads the JSON retrieved from the backend.

However, rarely changing parameters and a need for efficiency can sometimes lead to code that has problem-specific data baked in for faster execution that ultimately leads to reduced code, better user experience, etc. Such examples are easier found in the early days of computing, where resources were expensive and more often used for a single purpose. To name an example, timetabling programs sometimes had railway network related information hardcoded in them[7]. Today, it is rather rare to see problem-specific binaries, though one could argue for example, that G-codes are problem-specific instructions on a CNC machine. In some very edge-cases, where performance is of utmost importance, baked-in data can be present, such as aerospace or avionics firmware, or a missile guidance computer. It is also common in the automotive industry to heavily rely on code-preprocessing macros, and generate specific binaries. Instance-specific precompiled data appears more often, e.g., BSP data and PVS in video games[9] or models in Finite

MATCOS 2025, 9–10 October 2025, Koper, Slovenia
2025.

Element Analysis[1], and one may even consider neural network weights as precompiled data as well.

Returning to the main topic of this paper from this nearly philosophical detour, optimization is a computation heavy area, where every reduction in cost is highly sought after, similar to the aforementioned simulation examples. Instance Specific Parameter Tuning Strategies are researched for heuristic approaches[2], where instance-specific data is taken into account to parametrize the execution of the solution software, which has a similar motivation to that of this paper, but differs in the approach.

This paper presents the results of an experiment about the performance consequences of using instance-specific information compiled into a graph-based scheduling solver as shown in Figure 1.

In Section 2 a brief technical background is given, why an instance-specific binary may have performance benefits. Section 3 provides the definition of the problem class for the experiment and the algorithm used by the implementation described in Section 4. Finally, Section 5 shows the test results, and conclusions are drawn in Section 6.

2 Brief technical background

While big-O notation is an excellent tool to compare the theoretical efficiency of algorithms, in practice, the actual running time matters the most. However, even on the same machine with the same operating system, the CPU time depends not only on the algorithm's design, but on how it is expressed in the computers language. Implementations of the same algorithm can vary in many ways, which can also lead to very significant performance differences.[3] It is a well known fact in software development, that *lower level languages* such as C, C++, Rust, etc. generally outperform interpreted languages (JavaScript, Python) or languages that are generally compiled to a virtual machine (Java, C#). Most of the performance difference can be attributed to heap vs. stack allocations, advantages of static typing, cache locality, and various compiler-backend optimizations. These factors can also lead to very different results for two implementations in the same language.

In this study, C++, a well-established language[8] is used for testing the possible performance benefits of instance-specific implementations. While it is impossible to detail every aspect writing efficient C++ code[4] here, one key factor is worth highlighting. The algorithm described in the next Section iterates over many partial schedules, whose data is mostly encoded by a 2-dimensional matrix of weights, that is copied from a parent node and altered in the child nodes.

A significant difference between a generic and an instance-specific solver is, that the size of this matrix is known at compilation time in the latter case, which allows much faster copying. While basically one `memcpy` call is enough to copy an `array<array<int,SIZE>SIZE>`, doing the same on `vector<vector<int>>` usually results in `SIZE`+1 heap allocations, and a syscall if the heap allocated to the process is insufficient. Moreover, things like tasks assigned to the same unit do not need to be looked up dynamically, they can be hardcoded into the binary, which may also result in faster execution.

3 Problem definition & solution algorithm

The scheduling algorithms of the S-graph framework[6] are great candidates for the planned experiment due to their graph model and custom branching rules. However, for the sake of keeping the descriptions brief, the problem class was reduced to job-shop scheduling with arbitrary job lengths, machine revisits, and non-negative integer processing times. The objective is to find the schedule with minimal makespan.

For the description of the algorithm, the terminology of batch process scheduling is used instead of shop scheduling, as it fits to an S-graph based approach better. The model is a directed graph, where each task and product (delivery) is represented by a vertex, and timing differences between the start of their execution are expressed by directed weighted arcs. In a fully scheduled graph, the longest path equals to the makespan, and it also provides the bounding function for partially scheduled cases. At the root of the B&B tree, the processing queue of each unit is empty, and only the arcs expressing the production sequence of products are included. In each branching step, a unit with non-complete queue is selected, and the next task is fixed for each child node, by inserting an arc between the previous task and the next task with the weight of the processing time of the previous task. If a cycle in the graph is created, the schedule is infeasible.

This is similar to the so-called EQ-based approach of the S-graph framework, however, it does not specify, which partial schedule in the B&B tree is investigated next, and how the unit is selected for branching. Fine tuning these strategies can also have a significant effect on performance [5]. To remove this factor from the comparisons, the following strategy is used:

- Partial schedules are explored in a depth-first-search manner.
- At each, the processing unit with smallest index is selected among those that have an unfinished production queue.

As a result, the top of the B&B tree contains decisions related to unit 0, the levels below that to unit 1, etc. This strategy allows a simple recursive implementation, with the following logic:

```
subroutine EXPLORE(case, unit, prev, curr):
  global best
  case = case.copy()
  if prev != NULL:
    if case.has_path(curr,prev): return
    case.add_arc(prev,curr,prev.proctime)
  if case.longest_path() >= best: return
  if case.is_complete():
    best = case.longest_path()
  else:
    if case.is_unit_complete(unit):
      unit = case.next_incomplete_unit()
      for task in unit.tasks():
        EXPLORE(case,unit,NULL,task)
    else:
      for task in unit.tasks():
```

```
if not case.is_scheduled(task):
    EXPLORE(case, unit, curr, task)
```

All implementations in Section 4 rely on this same exploration logic of the search space. Further algorithmic acceleration techniques of the S-graph framework are not utilized in any of the implementations. This adjusted S-graph algorithm may not be the most efficient job-shop scheduler, however, it still retains the same important characteristics, as the general solvers for the wider problem class with precedential recipes, mixed intermadiate storage policies, changeover times, etc.

4 Generic and instance-specific implementations

A shared feature of all of the S-graph implementations is to maintain a longest path matrix with a cached maximum instead of the schedule-graph itself. This removes the need for dynamic longest path calculations and cycle checks. Maintaining such a matrix when inserting an arc takes $O(|V|^2)$ time in the worst case, however, this technique was proven to be advantageous over the years. The recipe is always stored and accessible as constant global data.

Altogether, there are 5 implementations that will be compared in the next section.

G1 is the generic implementation, where a partial schedule is stored in such a structure:

```
struct partial_schedule {
  vector<vector<uint>> longest_paths;
  uint longest_path_max;
  vector<vector<uint>> unscheduled_tasks;
};
```

In **G2**, the list of unscheduled tasks for each unit is not stored by a `vector` in this struct, but as a managed global boolean array, which is possible due to the DFS-like strategy. Comparing these two solvers will show, how much time is needed for managing several relatively short dynamic arrays.

The implementations **I1** and **I2** are the problem scpecific equivalents, where fix-sized `array`s are employed instead of `vector`s, and the subroutine responsible for returning the root `partial_schedule` is `constexpr` with the return of a list-initialized structure instead of opening the input file, and reading the data dynamically. **I3** is a special variant of **I2**, where the branching subroutine is instantiated for all possible `unit`, `prev`, `curr` triplets in compile time, thus, some of the conditionals and loops can be removed in exchange for longer compilation time and a larger binary.

It is important to note, that all implementations traverse exactly the same DFS-like B&B tree with the same number of nodes, copies, etc. There are no algorithmic differences between them in this sense, and the C++ code for exploring a node is the same in **G1** and **I1**, and similarly the same in the remaining three.

5 Empiric results

Test cases

All of the test cases were generated randomly and solved by all of the aforementioned implementations on the same machine. A single C++ source file, however, can result very different binaries based on the compiler used, and the applied configurations. All binaries used were compiled with `g++` (GCC version 14.2). After testing various compiler configurations on a smaller set of examples, two binaries were generated for each implementation: one with optimization level 1, and another with optimization level 3. Initial tests showed, that `-O2` and `-Ofast` did not show significant difference compared tot `-O3`, and naturally, no optimization always provided worse results. Altogether 80 instances were tested with all of the 5 implementations and 2 compiling options. The size of the instances ranged from 3 to 8 products with 2-6 stages carried out by 5-8 units.

Overheads

Instance-specific solvers have an overhead for generating and compiling the source files. This is especially heavy for **I3**. The largest such overhead was less than 6 seconds for **I3** with `-O3`, for a problem, whose solution was measured in hours. For the same problem the overhad in case of **I2** with `-O1` was less then 1 second. On the other hand, constructing the root node is faster for the instance-specific binaries, but that only milliseconds and microseconds in the worst case for the generic and instance-specific solvers, respectively. As a conclusion, these overheads did not play a significant role, except for small problems, where the optimal solution can be found in a few seconds either way.

Optimization levels

A somewhat unexpected result between the `-O1` and `-O3` optimization option for the 5 implementations is shown on a box-plot in Figure 2.

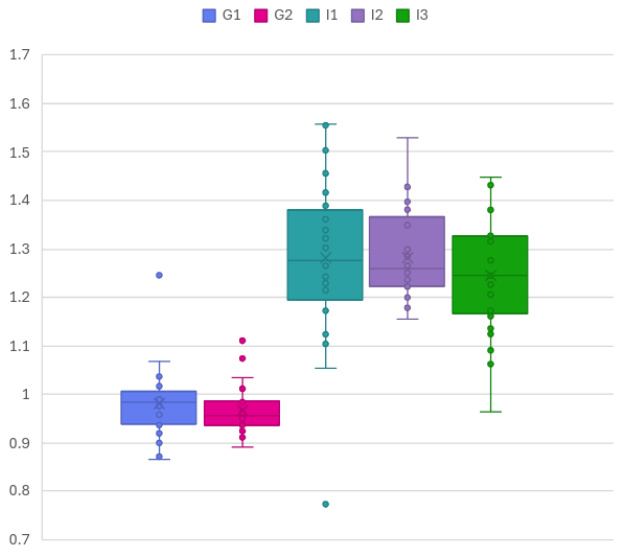


Figure 2: CPU time ratio of binaries with `-O3` and `-O1` for the 5 implementations

The figure shows the distribution of CPU time with `-O3` divided by that of with `-O1` of all the test cases for all implementations. While several outliers are cropped from the diagram, it is clear to see, that in case of the general solvers, more optimization had minor benefits, but the instance-specific solvers performed significantly better with

less optimization. Thus, in later comparisons results from the general solvers always refer to the `-O3` build, and to the `-O1` build for instance-specific solvers.

Instantiated functions

The overall performance of **I2** and **I3** were very close for all of the larger test cases. For the 5 largest ones, **I2** outperformed **I3** by 1,1,0,9,3 percent. **I3** is removed from further discussion, as function instantiation did not bring measurable results for this problem class, only slight performance decrease on top of the compilation overhead.

Unscheduled task set representation

Figure 3 shows the effect of storing the set of unscheduled tasks differently in case of the generic and instance-specific solvers.

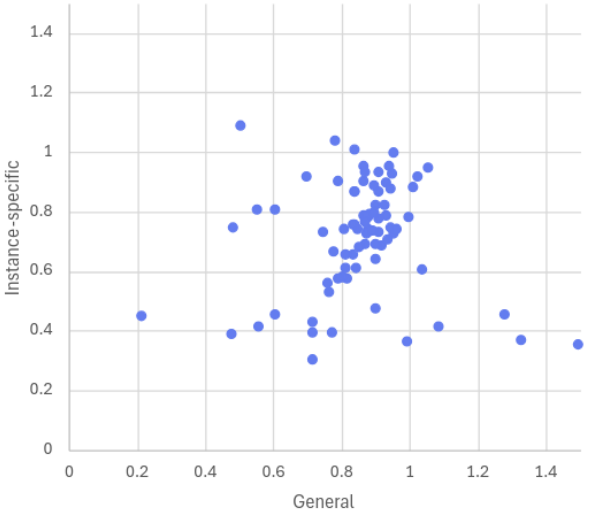


Figure 3: Effect of different unscheduled task set management

The Figure shows an XY plot, where the X and Y coordinates are CPU time ratios **G2/G1**, and **I2/I1**, respectively. For most of the instances, both coordinates are less than 1, the exceptions belong to smaller instances, i.e., maintaining a single boolean array was more beneficial, as expected. It can also be observed, that in general, this had a larger effect on instance-specific solvers, whose partial problem representation would not use vectors for other purpose.

Largest instances

Table 1 shows the overall results for several examples, including the largest cases and some medium-sized ones. The second column shows the number of tasks, which is the maximal depth of the B&B tree, thus the maximal number of function calls on the stack. Memory usage grows cubically with this in these DFS-like implementations. As is shown in the table, CPU becomes an issue much faster than RAM. Moreover, considering the 4 bytes for an unsigned int, the **I2** solver takes 256 kB to store all weights in the matrices in the call stack when a complete schedule is examined. This number is definitely higher for **G2**, but it is reasonable to assume, that syscalls for more heap space did not play a significant role for the larger examples.

Table 1: Overall results for some examples

case	# tasks	**G2**-O3	**I2**-O1	Reduction
66	40	11180 s	3246 s	71%
24	36	7392 s	2107 s	71%
45	30	62 s	14 s	77%
0	38	40 s	9 s	77%
1	39	19 s	4 s	79%
17	33	12 s	3 s	72%
69	37	8 s	2 s	78%

The last column of the table shows the reduction in CPU time by using insta-specific solvers, which are all in the 70-80% range.

6 Concluding remarks

While computer programs are meant to be generic for many reasons, performance focused applications may benefit from generating binaries that have instance-specific information encoded. In this work, this question was put to the test on an S-graph-based custom scheduler for job-shop problems, and results showed that a 70% reduction in computational needs is achievable mostly by fixing container sizes in compile time. This initial investigation motivates more exhaustive experiments on more complex real-life case-studies.

References

[1] Klaus-Jürgen Bathe. 2002. *Finite Element Procedures*. Prentice-Hall of India. ISBN: 8120310756. http://www.worldcat.org/isbn/8120310756.

[2] Yasemin Eryoldas and Alptekin Durmuşoğlu. 2021. A literature survey on instance specific algorithm configuration methods. *Proceedings of the International Conference on Industrial Engineering and Operations Management*. https://api.semanticscholar.org/CorpusID:240158129.

[3] M. Hegyhati and Zs. Tuza. 2023. Why implementation matters for graph algorithms? Presented at OPAL, Veszprém, Hungary. (June 2023).

[4] Scott Meyers. 2005. *Effective C++: 55 Specific Ways to Improve Your Programs and Designs*. (3rd ed.). Addison-Wesley Professional.

[5] Zsolt Ádám Nemes. 2015. *Heuristic Accelerations of the S-graph Solver*. Bachelor's thesis. University of Pannonia.

[6] E. Sanmartí, Luis Puigjaner, Tibor Holczinger, and Ferenc Friedler. 2002. Combinatorial framework for effective scheduling of multipurpose batch plants. *AIChE Journal*, 48, 2557–2570, 11. doi:10.1002/aic.690481115.

[7] G. R. Stibbs. 1958. Railway timetabling and early computers. *Proceedings of the IEE*.

[8] Bjarne Stroustrup. 2013. *The C++ Programming Language*. (4th ed.). Addison–Wesley, Boston, MA. ISBN: 978-0321958327.

[9] Seth J. Teller and Carlo H. Séquin. 1991. Visibility preprocessing for interactive walkthroughs. *SIGGRAPH Comput. Graph.*, 25, 4, (July 1991), 61–70. doi:10.1145/127719.122725.

Bi-Level Routing and Scheduling

Alain Quilliot and Hélène Toussaint
LIMOS Lab. UCA, CNRS and EMSE
Clermont-Ferrand, France
Email: alain.quilliot@uca.fr

***Abstract*—Decentralized renewable energy platforms promote *self-consumption*, a single agent becoming able to simultaneously act as a producer and a consumer of power. We deal here with the design of routing strategies for electric vehicles relying on a local photovoltaic (PV) platform. We propose an exact Mixed-Integer Linear Programming (MILP) setting solved through Branch and Cut, along with a heuristic approach involving an approximation of the behavior of the PV platform.**

I. Introduction

Renewable energy sources are promoting the emergence of the *self-consumption* paradigm [6], according to which consumers become energy producers. This paradigm raises its own issues, operational ones (synchronization of production and consumption) and tactical (pricing).
We consider here a *Vehicle Routing under Energy Production Constraints* **VR_EPC** problem which consists in routing electric vehicles (EV) while synchronizing their activity with the time dependent production and purchase of power. Numerous routing models (see *Green Vehicle Routing*, *Pollution Routing* models [2], [7], [9],[3]) have been designed, involving refueling transactions subject to time windows or shared access constraints, as well as CO2 emissions (see Kuo [4], Lajunen [5]). Nevertheless, few studies simultaneously dealt with both energy production and consumption, which requires integrating heterogeneous routing and scheduling processes (see [1]). Related models are complex ones, involving highly heterogeneous variables and relying on the existence of a single central decider, which may not fit most real life contexts. In any case, designing efficient algorithms remains a challenge.

Because of the collaborative features of our problem, we adopt here the point of view of the EV and propose:

- An exact MILP model that considers the *vehicle* as leader and includes *Recharge Decomposition* constraints requiring the application of a separation time-polynomial procedure. We solve it through branch-and-cut.
- Heuristic algorithms, that approximate the behavior of the power producer.

II. The VR_EPC Problem

We consider a photovoltaic (PV) platform, referred to as **PVP**, along with an electric vehicle (EV) in charge of visiting a set of customers. These two players interact through *recharge events* when the vehicles return to **PVP** to recharge. We suppose that the system is deterministic . Its main components are:

- **A Photo-Voltaic platform PVP**
  The time horizon of **PVP** is divided into N periods $i = 0, \ldots, N-1$, all with a same duration p. Thus period i starts at time $p \cdot i$ and ends at time $p \cdot (i+1)$. During this period, **PVP** is expected to produce R_i power units. It may also buy an additional amount y_i of power, that cannot exceed a threshold C^{Ch}. The cost of y_i may be written $\Phi_i(y)$, where Φ_i is a piecewise linear increasing convex function, convexity meaning that marginal power purchase prices are usually increasing. **PVP** is provided with a macro-battery, with storage capacity $C^{PVP} \geq C^{Ch}$ and initial load H_0^{PVP}. It must meet vehicle's demand, ending with a charge at least equal to H_0^{PVP} under minimal purchase cost $PCost$.

- **An Electric Vehicle EV and a Set of Customers J**
  EV is initially located at a *Depot*. It must visit, within the time horizon $[0, N.p]$, a set $\mathbf{J} = \{1, \ldots, M\}$ of customers according to a TSP (*Traveling Salesman*) route Γ, before coming back to *Depot*. Moving from a customer j to a customer k requires $E_{j,k}$ energy units and $T_{j,k}$ time units. Vectors T and E define distances on the set defined by $Depot$, **PVP** and **J**, $Depot$ and **PVP** being considered as customers identified as $Depot = 0$ at the beginning of the process, $Depot = M+1$ at the end and **PVP** $= -1$. **EV** is provided with a battery, with storage capacity C^{EV} and initial load H_0^{EV}. It must end with a charge at least equal to H_0^{EV} while minimizing the time EV_Time when it is back to $Depot$.

- **Recharge Events**
  EV must periodically move to **PVP** in order to recharge. A *recharge event* requires a single period i and takes place between two customers j, k consecutive according to Γ: **EV** moves from j to **PVP** before time $p \cdot i$, receives an amount $m \leq Inf(C^{Ch}, C^{EV})$ of power, and leaves **PVP** at time $p \cdot (i+1)$. This recharge event, denoted by $\omega = (i, j, k, m)$, induces a cost Ψ_i which depends on i and is independent on m (infrastructure resource cost). Purchasing power is forbidden during the period i. Recharge event ω may force **EV** to wait at **PVP** until the beginning of period i in case it arrives before time $p \cdot i$. If we denote by τ_j the time when **EV** arrives at j, by

V_j^{EV} its energy load and by V_i^{PVP} the energy load of the PV-plant at the beginning of i, then:

- $p \cdot i \geq \tau_j + T_{j,-1}$; $\tau_k = p \cdot (i+1) + T_{-1,k}$;
- $V_{i+1}^{PVP} = V_i^P - m \geq 0$;
- The load of **EV** at the end of i is equal to $V_j^{EV} - E_{j,-1} + m \leq C^{EV}$;
- The load of **EV** at the beginning of i is equal to $V_j^{EV} - E_{j,-1} \geq 0$.

Given *time versus money* coefficient α, we set **VR_EPC** as a bi-level mono-objective problem, with **EV** a leader.

VR_EPC: ***Vehicle Routing under Energy Production Constraints***: {Compute the route Γ followed by **EV** together with the recharge events linking **PVP** and **EV** in such a way that:

- All customers are visited once within the time horizon.
- Vehicle storage capacity constraints and energy requirements are satisfied.
- Extended cost $PCost + \alpha.EV\_Time$ is minimized, under the **PVP** constraints:
  - y meets the demand related the recharge events;
  - It meets **PVP** storage and charge capacities as well as the constraint about the final charge of **PVP**;
  - $PCost = \sum_i (\Psi_i \cdot \delta_i + \Phi_i(y_i))$.}

III. A MILP Model Solved through Branch&Cut

Since we adopt here the point of view of the vehicle, our master variables are 2 $\{0,1\}$-valued vectors $Z = (Z_{j,k}, k \neq j \in \{-1, 0, \ldots, M+1\})$ and $X = (X_{j,k}, k \neq j \in \{0, \ldots, M+1\})$, describing the routes followed by the vehicle with and without the recharge detours respectively. Those main **EV** variables are completed by time and charge **EV** variables, by **PVP** variables and by *Recharge Event* variables.

- **EV** variables:
  - $\{0,1\}$-valued $Z_{j,k}, k \neq j \in \{-1, 0, \ldots, M+1\}$: $Z_{j,k} = 1$ iff **EV** moves from j to k: (-1 = **PVP**).
  - $\{0,1\}$-valued $X_{j,k}, k \neq j \in \{0, \ldots, M+1\}$: $X_{j,k} = 1$ iff **EV** moves either from j to k or from j to **PVP**, and next from **PVP** to k.
  - Non negative $L_j^{EV}, j = 0, \ldots, M$: L_j^{MP} is the power transferred to **EV** just after j if $Z_{j,-1} = 1$;
  - Non negative $V_j^{EV}, j = 0, \ldots, M+1$: V_j^{EV} is the power stored by **EV** when it arrives at j;
  - Non negative $\tau_j, j = 0, \ldots, M+1$: τ_j is the time when **EV** arrives at j;
  - Non negative $\tau_j^*, j = 0, \ldots, M+1$: τ_j^* is the time when **EV** starts recharging after j if $Z_{j,-1} = 1$.
- **PVP** variables:
  - Non negative $y_i, i = 0, \ldots, N-1$: y_i means the power bought by **PVP** during period i;
  - $\{0,1\}$-valued $\delta_i, i = 0, \ldots, N-1$: $\delta_i = 1$ means that some recharge event takes place at i;
  - Non negative $V_i^{PVP}, i = 0, \ldots, N-1, N$: V_i^{PVP} is the charge of **PVP** at the beginning of i;
  - Non negative $L_i^{PVP}, i = 0, \ldots, N-1$: L_i^{PVP} is the power transferred at i if $\delta_i = 1$.
- **Recharge Event** variables:
  - $\{0,1\}$-valued $U_{i,j}, i = 0, \ldots, N-1, j = 0, \ldots, M$: $U_{i,j} = 1$ means that some recharge event involving j occurs at period i;
  - Non negative $m_{i,j}, i = 0, \ldots, N-1, j = 0, \ldots, M$: $m_{i,j}$ means related amount of power.

A. Structural *EV* Constraints

Z and X describe the full route followed by **EV**. They must clearly meet the standard *vehicle routing* constraints:

- $Z_{M+1,0} = 1; \quad \forall j : Z_{j,j} = 0;$ (VR1)
- $\forall j = 0, \ldots, M+1$: $\sum_{k=-1,\ldots,M+1} Z_{j,k} = 1 = \sum_{k=-1,\ldots,M+1} Z_{k,j}$; (VR2)
- $\sum_{j=0,\ldots,M+1} Z_{j,-1} = \sum_{j=0,\ldots,M+1} Z_{-1,j} \geq 1$; (VR3)
- $\forall j, k \in \{0, \ldots, M+1\}: \quad X_{j,k} \geq Z_{j,k}$; (VR4)
- $\forall j \in \{0, \ldots, M+1\}$: $\sum_{k=0,1,\ldots,M+1} X_{j,k} = 1 = \sum_{k=0,1,\ldots,M+1} X_{k,j}$; (VR5)

(VR1, ..., VR5) do not ensure that Z, X defines a route, consistent with the power requirements. We reinforce them by noticing that if **EV** spends W energy inside or at the border of some customer subset A which does contain **PVP**, then it must move at least $\lceil \frac{W}{C^{EV}} \rceil$ times towards **PVP** in order to recharge. Let us set:

- For any such a subset A of $\{-1, 0, \ldots, M, M+1\}$:
  - $Cl(A) = \{(j,k)$ s.t at least j or k is in $A\}$;
  - $\delta(A) = \{(j,k)$, s.t $j \notin A$ and $k \in A\}$.
- For any (j,k):
  - $\Pi_{j,k} = E_0$ if $(j,k) = (M+1, 0)$ and $\Pi_{j,k} = C^{EV}$ else.
  - $\Pi_{j,k}^* = C^{EV} - E_0$ if $(j,k) = (M+1, 0)$ and $\Pi_{j,k}^* = C^{EV}$ else.

Then we derive the *Recharge Decomposition* constraints:

- For any $A \subseteq \{0, \ldots, M+1\}$, $\sum_{(j,k) \in \delta(A)} \Pi_{j,k} \cdot Z_{j,k} \geq \sum_{(j,k) \in Cl(A)} E_{j,k} \cdot Z_{j,k}$ (VR6)
- For any $A \subseteq \{0, \ldots, M+1\}$, $\sum_{(j,k) \in \delta(J-A)} \Pi_{j,k}^* \cdot Z_{j,k} \geq \sum_{(j,k) \in Cl(A)} E_{j,k} \cdot Z_{j,k}$ (VR6-Bis)

Lemma 1: *The VRP constraints (VR1, ..., VR6-Bis) hold if and only if the arcs (j,k) such that $Z_{j,k} = 1$ define a collection γ of sub-tours $\gamma_s, s = 0, \ldots, S$ such that:*

- *γ_0 starts from Depot $= 0$, ends into* **PVP** $= -1$*, and spends less than H_0^{EV} power.* (SUB1)
- *γ_S starts from* **PVP** $= -1$*, ends into Depot $= 0$, and spends less than C^{EV}–H_0^{EV} power.* (SUB2)
- *For any $s = 1, \ldots, S-1$, γ_s starts from* **PVP** $= -1$*, ends into -1 and does not require more than C^{EV} power.* (SUB3)
- *Every customer $j = 1, \ldots, M$ is visited once.* (SUB4)

Remark 1: Constraints (VR6, VR6-Bis) do not order the sub-routes $\gamma_s, s = 0, \ldots, S$.

Separating the Recharge Decomposition Constraints: Given 2 possibly non integral vectors (Z, X), separating the constraints (VR6, VR6-Bis) means checking that all those constraints are satisfied by (Z, X) and, in case they are not, computing a contradicting subset $A \subseteq \{0, \ldots, M+1\}$.

Theorem 1: *The Recharge Decomposition constraints can be separated in polynomial time, via a min cost flow algorithm.*

Principle of the Proof: We construct an auxiliary multi-graph $G = (X, A)$ with $X = \{Source = -1, 0, \ldots, M+1, M+2 = Sink\}$ and A is defined as the set of all *simple-arc* $(j, k), j \neq k \in \{-1, 0, \ldots, M+1\}$, augmented, for every j, with *copy-arcs* $(j, M+1)^k, k \neq j \in \{-1, 0, \ldots, M+1\}$, connecting j to $Sink = (M+2)$ and provided with label k. Every *copy-arc* $a = (j, M+1)^k$ is provided with a weight w_a equal to $E_{j,k} \cdot Z_{j,k}$. Every *simple-arc* $a = (j, k)$ is provided with a weight w_a equal to $\Pi_{j,k} \cdot Z_{j,k} - E_{j,k} \cdot Z_{j,k}$. Then computing A that contradicts VR6 (VR6-Bis) means computing some cut B which separates $Source = -1$ from $Sink = M+2$ in G and is such that $\sum_{a \ s.t. \ origin(a) \in B', destination(a) \notin B'} w_a$ does not exceed some threshold. This can be done in polynomial time through a Max-Flow algorithm. **End-Proof**.

*B. The **VR_EPC_MILP** MILP formulation*

We get it while distinguishing 3 main groups of constraints:

- The **PVP Constraints**: They involve the variables related to the purchase of power and express the evolution along the periods of the load V_i^{PVP} of the **PVP** battery.
- The **EV Constraints**: They contain VR1, ..., VR6-Bis, together with constraints related to the time and power values when the vehicle visit the customers or to **PVP**.
- The **Recharge Event Constraints**: They link together the **PVP** periods and the time horizon $[0, p \cdot N]$ of **EV**.

Those constraints come as follows:

VR_EPC Constraints and Objective Function:

- **Objective**: Minimize $\sum_i (\Psi_i.\delta_i + \Phi_i(y_i) + \alpha.\tau_{M+1}$
- **PVP Constraints**
  - $\forall i = 1, \ldots, N-1$: $y_i \leq C^{Ch} \cdot (1 - \delta_i)$; (PC1)
  - $\forall i = 0, \ldots, N$: $V_i^{PVP} \leq C^{PVP}$; (PC2)
  - $V_0^{PVP} = H_0^{PVP}$; $V_N^{PVP} \geq H_0^{PVP}$; (PC3)
  - $\forall i = 1, \ldots, N$: $V_i^{PVP} = V_{i-1}^{PVP} + y_i - L_i^{PVP}$; (PC4)
- **EV Constraints**
  - (VR1, ..., VR6-Bis) involved in Lemma 1;
  - $V_0^{EV} = H_0^{EV}$; $V_{M+1}^{EV} \geq H_0^{EV}$; (VR7)
  - $\forall j = 0, \ldots, M+1$: $E_{j,-1} \leq V_j^{EV} \leq C^{EV}$; (VR8)
  - $\forall j, k = 0, \ldots, M+1$: $X_{j,k} \rightarrow (V_k^{EV} + E_{j,k} + (Z_{j,k} - 1) \cdot (E_{j,-1} + E_{-1,k} - E_{j,k})) \leq (V_j^{EV} + L_j^{EV})$; (VR9)

TABLE I
BEHAVIOR OF **VR_EPC_MILP**

Id	(N, M, p)	LBG_LP	UBG_LP
3	(160, 10, 1)	51,6	171
3-2	(160, 10, 2)	51,7	174
5	(240, 20, 1)	43,8	148
6	(240, 20, 1)	59,5	176
5-2	(240, 20, 2)	43,9	148
6-2	(240, 20, 2)	59,1	201
8	(320, 30, 1)	58,0	151
9	(320, 30, 1)	51,7	117
8-2	(320, 30, 2)	58,0	153
9-2	(320, 30, 2)	51,2	127

  - $\forall j = 0, \ldots, M$: $Z_{j,-1} \rightarrow (V_j^{EV} \geq E_{j,-1})$; (VR10)
  - $\forall j = 0, \ldots, M$: $Z_{j,-1} \rightarrow (V_j^{EV} + L_j^{EV} \leq E_{j,-1} + C^{EV})$; (VR10-Bis)
  - $\tau_0 = 0$; $\tau_{M+1} \leq p \cdot N$; (VR11)
  - $\forall j, k = 0, \ldots, M+1$: $Z_{j,k} \rightarrow (\tau_j + T_{j,k} \leq \tau_k)$; (VR12)
  - $\forall j, k = 0, \ldots, M+1$: $(X_{j,k} - Z_{j,k} = 1) \rightarrow (\tau_j^* + p + T_{-1,k} \leq \tau_k)$; (VR13)
  - $\forall j = 0, \ldots, M+1$: $(Z_{j,-1}) \rightarrow (\tau_j + T_{j,-1} \leq -\tau_j^*)$; (VR13-Bis)
- **Synchronization Constraints**
  - $\forall j = 0, \ldots, M$: $\sum_{i=0,\ldots,N-1} U_{i,j} = Z_{j,-1}$; (SY1)
  - $\forall j = 0, \ldots, M$: $\sum_{i=0,\ldots,N-1} m_{i,j} = L_j^{EV}$; (SY1-Bis)
  - $\forall i = 0, \ldots, N-1$: $\sum_{j=0,\ldots,M} U_{i,j} = \delta_i$; (SY2)
  - $\forall i = 0, \ldots, N-1$: $\sum_{j=0,\ldots,M} m_{i,j} = L_i^{PVP}$; (SY2-Bis)
  - $\forall j = 0, \ldots, M$: $\sum_{i=0,\ldots,N-1} p \cdot i \cdot U_{i,j} = \tau_j^*$; (SY3)
  - $\forall i = 0, \ldots, N-1, j = 0, \ldots, M$: $m_{i,j} \leq Inf(C^{PVP}, C^{EV}) \cdot U_{i,j}$; (SY4)

Theorem 2: *Above **VR_EPC_MILP** MILP model solves **VR_EPC** in an exact way.*

We handle **VR_EPC_MILP** through Branch and Cut.

C. Numerical Experiments

Purpose: Evaluating the **VR_EPC_MILP** MILP model. **Technical Context:** The experiments are performed on an AMD EPYC 7H12 64-Core processor, under Gnu/linux Ubuntu 20.04.2CPLEX 12.10 is used in single-thread mode. **Instances:** The main parameters of every instance are the customer number M (from 5 to 30), the period number N (from 40 to 320), and the period length $p \in \{1, 2, 4\}$. We derive instances with $p = 2$ $(Id - 2)$ from instances with $p = 1$ (Id) by merging the periods, $p \cdot N$ remaining the same.

Results: For every instance, Table II displays the lower bound LBG_LP and the upper bound UBG_UB computed through Branch and Cut by CPLEX library in 2 CPU hours.

IV. Heuristic Handling of VR_EPC

We drive the leader route Γ (the vector X of the MILP setting) while relying on the correlation which exists between the optimal value of **VR_EPC**(Γ) and some key features of Γ. Let us $W(\Gamma)$ be the optimal value **VR_EPC**(Γ) and $L^T(\Gamma), L^E(\Gamma)$ be the lengths of Γ in the sense of T and E respectively. Standard TSP *2_Opt* and *Reloc* operators act on any route Γ through 2 parameters j_1, j_2 in $\mathbf{J} + \{Depot\}$:

- *2_Opt*(Γ, j_1, j_2) replaces the moves from j_1 (j_2) to its successor $\bar{j}_1$ ($\bar{j}_2$) by moves from j_1 ($\bar{j}_1$) to j_2 ($\bar{j}_2$);
- *Reloc*(Γ, j_1, j_2) relocates j_1 between j_2 and $\bar{j}_2$.

A. Approximating $W(\Gamma)$ through Path Search

Once Γ is fixed, a full solution is determined by the sequence of recharge events linking **EV** and **PVP**, augmented with the amounts of power bought between 2 consecutive recharge transactions. Thus, we design a *PDYN_EPC* algorithm that searches for a path in a *transition* graph:

- A state in the sense is a 4-uple $S = (j, i, V_j^{VEV}, V_i^{PVP})$, where j is a customer, i a period, V_j^{EV} is the power load of **EV** when it leaves j and V_i^{PVP} the power load of **PVP** at the end of i, with the implicit meaning that a recharge transaction involving i and j has just been performed.
- A decision is a 4-uple $(j_1, i_1, m_1, \bar{y})$, where (j_1, i_1, m_1) means the next recharge event and $\bar{y}$ means the power bought by **PVP** during the periods $i+1, \ldots, i_1 - 1$. Related transition cost is $p \cdot (i_1 - i)$ augmented with the purchase cost of $\bar{y}$, which corresponds to the optimal value of some convex optimization program and is computed and stored as part of a pre-process.
- Initial state S_0 is a 4-uple $(Depot, -1, H_0^{EV}, H_0^{PVP})$. A final state is any 4-uple $(Depot, N-1, V_j^{EV} \geq H_0^{EV}, V_j^{PVP} \geq H_0^{PVP})$.

We speed the algorithm by fixing an upper bound $NDec$ on the number of feasible decisions and introduce filtering devices.

B. Two Simple Heuristic Algorithms

Experiments show that the evolutions of $W(\Gamma)$, $L^T(\Gamma)$, $L^E(\Gamma)$ and the number NRT during a decent loop performed on $W(\Gamma)$ via the *2_Opt* and *Reloc* operators are strongly correlated. So we partially short-cut $W(\Gamma)$ and rely on $L^T(\Gamma)$ and $L^E(\Gamma)$ in order to drive Γ towards good solutions.

The GRASP Algorithm *GRASP_VR_EPC*:It considers a replication parameter Q and for any $q = 1, \ldots, Q$:

1) It randomly generates parameters $\omega^T, \omega^E \geq 0$ together with an initial $\Gamma(q)$;
2) It applies *2_Opt* and *Reloc* until $\Gamma(q)$ becomes a local optimum with respect to $\omega^T \cdot L^T(\Gamma) + \omega^E \cdot L^E(\Gamma)$;
3) It computes $W\_Aux(\Gamma(q))$ via*PDYN_EPC* and updates the best current route $\Gamma(q_{Best})$.

TABLE II
Results for *GRASP_VR_EPC*

Id	$GRASP\_150$	$Desc+LS$	T_Desc	T_GRASP
3	173	165	122,7	145.6
3-2	176	171	35,6	47.6
5	156	168	356,2	327.2
6	175	176	565,7	845.1
5-2	158	173	109,8	134.1
6-2	181	181	203,6	227.2
8	121	121	1998,6	542.5
9	119	122	2351,0	578.9
8-1	132	134	773,0	246.7
9-1	123	130	629,1	189.3

The Pseudo-Descent Algorithm *Descent_VR_EPC*: We allow *2_Opt* and *Reloc* to slightly deteriorate $L^T(\Gamma)$ and $L^E(\Gamma)$. Given 2 parameters $\delta^T > 0, \delta^E > 0$ and 2 routes Γ_1, Γ_2, we say that Γ_2 deteriorates Γ_1 by no more than (δ^T, δ^E) if $L^T(\Gamma_2) - L^T(\Gamma_1) \leq \delta_1$ or $L^E(\Gamma_2) - L^E(\Gamma_1) \leq \delta_2$. Then the *Descent_VR_EPC*(δ^T, δ^E) algorithm works as follows:

1) Initialize Γ by applying *GRASP_SVR_EP*(1);
2) At any iteration do
   a) Generate all the *2_Opt* and *Reloc* parameters (j_1, j_2) such that resulting route deteriorates Γ by no more than (δ^T, δ^E);
   b) For any (j_1, j_2) selected this way and any resulting route Γ_1 compute $W\_Aux(\Gamma_1)$ through *PDYN_EPC*; If improving Γ is possible, then do it according to a *Best Descent* strategy else stop.

C. Numerical Experiments

Table III involves the same instances as Table II. It displays:

- The value $GRASP\_150$ obtained by *GRASP_VR_EPC*(150) and related CPU time T_GRASP.
- The value $Desc+LS$ by pipelining: *GRASP_VR_EPC*(1) (1 replication) $\rightarrow$ *Descent_VR_EPC*(4, 4), together with CPU time T_Desc.

Comments: *GRASP_VR_EPC* with 150 replications most often reaches quasi-optimality. The pipe-line *GRASP_VR_EPC*(1) $\rightarrow$ *Descent_VR_EPC*(4, 4) improves resulting value in 2 among the 10 instances.

References

[1] Drexl M.: Synchronization in vehicle routing–a survey; *Transportation Science* 46, pp. 297-316, (2012)
[2] Erdelic T., Caric T., Lalla-Ruiz E.: A Survey on the Electric Vehicle Routing Problem.*Journal of Advanced Transportation* (2019).
[3] Koç C., Jabali O., Mendoza J., Laporte G.. The electric vehicle routing problem with shared charging stations. *ITOR*, 26, (2018).
[4] Kuo Y.. Simulated annealing to minimize fuel consumption for the time-dependent VRP. *Comp. Indust. Eng.*, 59 :157–165, (2010).
[5] Lajunen A.: Energy consumption and cost analysis of electric city buses. *Transportation Research Part C*, 38 :1–15, (2014).
[6] Luthander R., Widen J., Nilsson D., Palm J.: Photovoltaic self-consumption in buildings. *Applied Energy* 142, p 80-94, (2015).
[7] Macrina G., Di Puglia L., Guerriero F.: The Green-Vehicle Routing Problem: A Survey. *Model. and Optim. in Logistics* p 1-26 (2020).
[8] Verma A.: Electric vehicle routing with recharging stations. *Euro Journal of Transport. Logistics* 7, p 415-451, (2018).

A discrete event simulation model for analyzing the wood waste reverse supply chain

Nikola Kovačević
nikola.kovacevic@famnit.upr.si
UP FAMNIT, University of Primorska
Koper, Slovenia

Črtomir Tavzes
crtomir.tavzes@innorenew.eu
InnoRenew CoE, UP IAM, University of Primorska
Koper, Slovenia

Balázs Dávid
balazs.david@innorenew.eu
InnoRenew CoE, UP IAM and UP FAMNIT, University of Primorska
Koper, Slovenia

Abstract

This paper presents a discrete event simulation model for analyzing reverse supply chain operations of wood waste materials. The model enables comprehensive evaluation of storage management, processing capabilities and resource allocation in reverse logistics networks, tracking the flow and transformation of resources from generation to its final destination either as recovered and reused products of landfilled waste. The proposed model is capable of managing different inventory policies and stock strategies. The efficiency of the proposed model is shown on a network generated based on statistical data of waste wood in Slovenia.

Keywords

reverse supply chain, discrete event simulation, inventory management, wood waste

1 Introduction

Wood waste represents a growing environmental and economic challenge worldwide, driven by expanding construction, manufacturing, and demolition activities. If not managed properly, it constitutes both a lost resource and a significant burden, contributing to greenhouse gas emissions, pollution, and rising landfill costs. Recent years have witnessed increased wood waste generation due to expanding construction sectors, including substantial increases in renovations and refurbishments for structural and energy improvements of building stock, alongside growing demand for wood-based packaging [14].

The environmental and economic consequences of wood waste mismanagement are profound. Conventional disposal methods, such as landfilling and incineration, release harmful greenhouse gases like methane and carbon dioxide, while slow decomposition under anaerobic conditions creates long-term environmental hazards [11, 10]. Beyond ecological damage, improper disposal poses public health risks due to associated pollution. Conversely, wood waste remains a vastly underutilized resource, with substantial volumes that could be repurposed into valuable raw materials or energy sources if managed effectively [2, 9].

To fully realize these opportunities, effective reverse supply chains are essential. Unlike traditional forward supply chains, which move products from raw materials to consumers, reverse supply chains focus on recovering value from used products through collection, reprocessing, and redistribution [5, 12]. Wood waste presents unique logistical and environmental challenges, including uncertain return volumes, variable product conditions, and complex processing requirements, but also significant potential for energy recovery and impact reduction.

Reverse supply chains for wood waste operate within the framework of cascading use principles, where reclaimed wood undergoes successive down-cycling through high-value applications—from structural reuse to particleboard feedstock—before final energy recovery, thereby maximizing carbon storage while reducing virgin material demand [3].

Wood waste reverse supply chains involve diverse stakeholders spanning sawmills, panel producers, demolition firms, recyclers, transport providers, and municipal waste collectors. These actors coordinate complex logistical processes including collection, transportation, quality control, sorting, and processing operations. However, this complexity introduces substantial challenges: seasonal and spatial dispersion of biomass sources, wide variability in material form and contamination levels, high processing costs, and insufficient information systems for post-consumer wood recovery [7].

Discrete Event Simulation (DES) has emerged as a particularly effective tool for modeling these complex reverse logistics systems. Unlike analytical methods that often require simplifying assumptions, DES can capture the stochastic nature of return flows, processing variability, and resource constraints that characterize real-world reverse logistics operations [4].

Discrete-event simulation has proven effective for studying supply chain dynamics and transport logistics, particularly in complex systems where uncertainty and stochastic elements play critical roles [8]. By representing a supply chain as a sequence of discrete events, DES enables detailed representation of operational policies and aids in evaluating alternative strategies and system responses to uncertainty. Recent advances demonstrate DES capability to analyze complex multi-level resilience relationships that traditional analytical methods struggle to capture [6].

Current research predominantly employs deterministic optimization models that fail to capture the inherent stochasticity of wood waste systems. While existing research has advanced understanding of individual system components, there remains a gap in DES applications to wood waste reverse supply chains. The bidirectional flows, uncertain material quality, and variable availability patterns characteristic of these systems require simulation approaches capable of representing stochastic interactions and multi-stakeholder coordination mechanisms.

This paper presents a DES model designed for analyzing reverse supply chain operations in wood waste management contexts. The model enables evaluation of processing capabilities, resource allocation strategies, and operational policies through a dual-axis inventory management framework that combines push/pull strategies with different stock management approaches. Using Slovenia as a case study, we demonstrate how the simulation framework supports evidence-based decision-making in reverse logistics network design and management.

MATCOS 2025, 9–10 October 2025, Koper, Slovenia
2024.

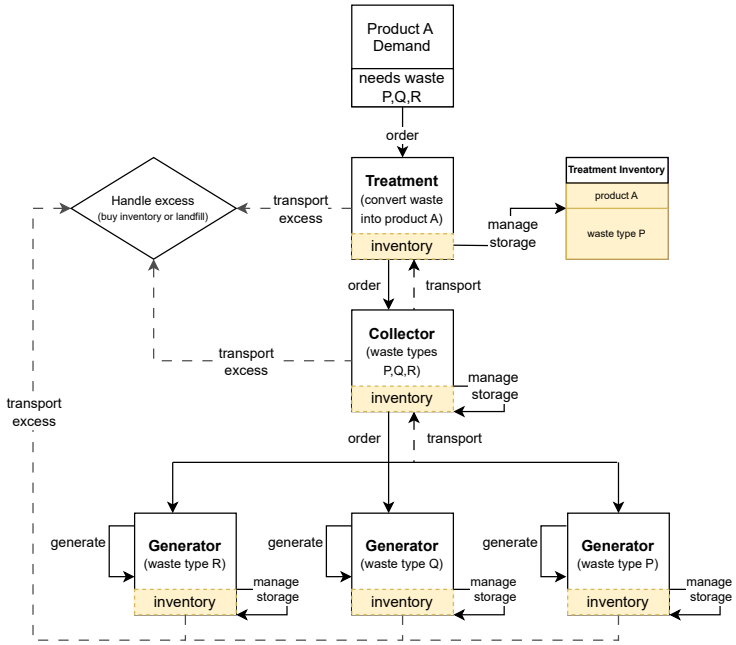


Figure 1: Overview of the model.

2 Model Development

A discrete-event simulation model was developed for the representation of the wood waste reverse supply chain. The proposed framework models waste flows from generation through collection to final processing across connected regions, with each region containing generators, collectors, and treatment facilities. The system tracks three primary entity categories that interact through event-driven communication protocols, responding to state changes and resource availability.

Figure 1 provides an overview of this system, presenting all model entities, their connections and the possible decisions available to them.

Table 1: Policy Configuration Framework

Configuration	Description
PUSH ON DEMAND	Continuous forecast-driven production
PUSH REORDER 50	Production triggered at 50% capacity
PUSH REORDER 90	Production triggered at 90% capacity
PULL ON DEMAND	Demand-triggered operation only
PULL REORDER 50	Demand-driven with 50% threshold
PULL REORDER 90	Demand-driven with 90% threshold

2.1 Entity Modeling

Waste Generators represent real-world actors that produce wood waste, serving as material sources introducing specified quantities and types over time. Generator behavior incorporates variability through random seasonal fluctuations reflecting market and operational uncertainties. Generators produce waste proactively regardless of system policy.

Collector Companies serve as commercial intermediaries operating as midstream buffers managing bidirectional material flows. Collected material is stored in finite-capacity collection centers that decouple generation events from treatment scheduling. The transport system operates at two levels: local collection from generators to collection centers with capacity and distance constraints, and long-haul transport from collection centers to treatment facilities using priority-based scheduling.

Treatment Operators represent industrial facilities that serve as final waste customers and value-adding stages. These entities consume waste as raw material, performing transformation according to predefined recipes for products.

2.2 Decision strategies

All entities operate under a dual-axis operational paradigm that collectively dictates production and procurement logic throughout the reverse supply chain. The **Inventory Policy** axis defines strategic philosophy: PUSH policies operate using forecast-driven approaches, managing operations based on internal state projections, while PULL policies implement lean, demand-driven approaches aligned with Just-in-Time principles. The **Stock Strategy** axis defines tactical execution rules: ON DEMAND represents continuous operation under PUSH or operation only upon signal receipt under PULL, while buffer-based strategies (REORDER 50 and REORDER 90) trigger actions when inventory levels cross 50% or 90% capacity thresholds respectively. This framework generates six distinct operational models (2×3) that enable systematic empirical comparison.

Table 1 summarizes the six operational configurations evaluated in this study. Each configuration represents a unique combination of inventory policy (PUSH vs PULL) and stock strategy (ON DEMAND, REORDER 50, REORDER 90), enabling analysis of different supply chain management approaches.

The framework incorporates realistic capacity constraints through an overflow management system. When entities exceed storage capacity, they choose between capacity expansion or landfill disposal using cost-minimization logic with dynamic pricing that escalates with repeated use, reflecting real-world constraints and discouraging reactive management.

2.3 Stochastic Elements

The model incorporates several stochastic components to capture real-world uncertainty. Entity failures use uniform recovery durations with failure probabilities varying by scenario severity. Waste generation includes daily variability factors from clipped normal distributions, while treatment conversion efficiency is normally distributed around base values. The model ensures reproducibility through hierarchical random number generation with deterministic seeding for consistent behavior across runs.

3 Results

The simulation study analyzes six operational models across a 365-day simulation period representing one operational year. Each configuration underwent 100 Monte Carlo replications with unique random seeds to capture stochastic variability.

System performance is evaluated through five complementary metrics that capture operational effectiveness across the complete material flow trajectory.

Service level quantifies system reliability by measuring the percentage of generated waste successfully converted into final products, representing the fundamental waste diversion capability. Operational efficiency provides hierarchical assessment through three interconnected ratios: collection efficiency (waste collected relative to generated), processing efficiency (waste converted relative to collected), and overall system efficiency (final products relative to initial generation), thereby identifying bottlenecks and resource utilization patterns throughout the supply chain. Storage utilization examines capacity management by tracking current storage usage against maximum capacity across collection centers and processing facilities, revealing both potential constraints and the effectiveness of demand-supply synchronization under different inventory policies. Environmental impact quantifies the emissions in $kgCO_2e$ equivalent generated through transportation, processing operations, and landfill disposal activities. Finally, landfill overflow measures the volume of waste diverted to disposal due to system capacity limitations or processing constraints, representing both environmental burden and foregone resource recovery opportunities.

Collectively, these metrics provide comprehensive assessment of reverse supply chain performance across operational, environmental, and resource recovery dimensions.

The simulation framework was implemented with SimPy as the core simulation engine [13].

3.1 Setup

The analytical framework employed Slovenia as the case study, with each of the 12 statistical regions of the country represented by a single instance of each system entity (generator, collector and treatment). The system incorporated 14 waste codes and 3 product types.

These types were selected to be engineered wood products: Particle Board (accepting five waste types), OSB (four waste types), and MDF (four waste types). Production prioritization follows ABC analysis based on total biogenic carbon impact, with OSB demonstrating the highest carbon storage potential ($-1213.60\,kgCO_2e/m^3$).

While not all waste codes contribute to product creation, unused codes were retained to maintain representation of Slovenia's wood-related waste streams, and resources of these waste codes still contributed to decisions about inventory management and landfilling. Initial generation rates derived from empirical data by the Slovenian Environment Agency (ARSO) datasets [1] and underwent dynamic adjustment throughout the 365-unit simulation period.

3.2 Inventory Management Performance

Storage utilization analysis reveals distinct patterns across entity types and strategies. PUSH strategies maintain consistently high storage utilization with frequent saturation events, particularly in the simulation's latter half, reflecting proactive inventory prepositioning regardless of downstream demand. PULL strategies demonstrate more dynamic, demand-responsive behavior with lower average utilization and greater temporal variability.

At the generation level, PUSH policies show progressive storage accumulation leading to widespread saturation across most regions, with particularly high utilization (80-100%) sustained throughout the simulation period. PULL strategies maintain more balanced levels with moderate utilization (20-60%) and better demand synchronization, avoiding the excessive accumulation patterns observed in PUSH configurations.

Collection storage exhibits similar patterns, with PUSH policies creating sustained high utilization across all regions and stock strategies, while PULL policies show significantly lower average utilization with more efficient turnover. The temporal analysis reveals PUSH strategies experience early saturation that persists throughout the simulation, whereas PULL strategies maintain responsive storage levels that adapt to demand fluctuations.

3.3 Cost-Environmental Impact Analysis

The cost-environmental impact analysis, based on single simulation runs for each configuration, reveals significant trade-offs across supply chain configurations. PULL REORDER 90 emerges as the best-performing strategy for environmental performance, achieving the lowest environmental impact (approximately 800k $kgCO_2e$) at competitive costs (around 30 million €). PULL ON DEMAND demonstrates comparable performance with slightly higher emissions (approximately 970k $kgCO_2e$) at lower cost levels. PULL REORDER 50 occupies an intermediate position within the PULL solution space, demonstrating suboptimal efficiency relative to alternative configurations due to higher costs despite moderate environmental performance.

PUSH strategies generally show higher environmental impacts, with PUSH REORDER 90 representing the least favorable option, generating approximately 2m $kgCO_2e$ emissions at costs around 58 million €.

Figure 2 illustrates the total costs and environmental impacts of each strategy, demonstrating PULL policy as the dominant policy across both cost and environmental dimensions.

3.4 Monte Carlo Analysis Results

The statistical analysis based on 100 Monte Carlo replications reveals different performance patterns when accounting for stochastic variability. All strategies achieved high service levels

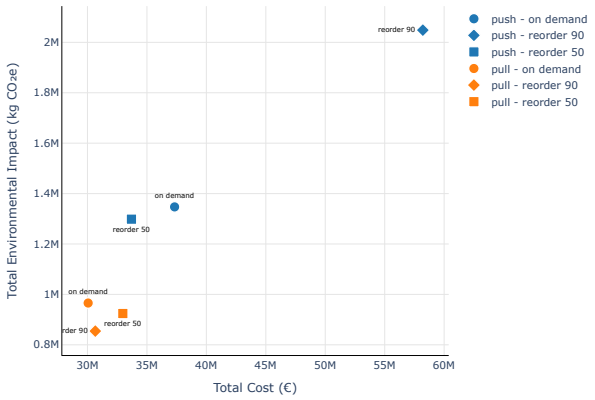


Figure 2: Cost versus environmental impact analysis across supply chain configurations.

above 97%, with PULL REORDER 50 reaching the highest average (99.9%), followed closely by PULL REORDER 90 (99.7%). PUSH strategies showed slightly lower but still strong service levels, ranging from 97.8% (PUSH ON DEMAND) to 98.6% (PUSH REORDER 90).

Environmental impact analysis reveals PULL REORDER 50 as the most environmentally friendly option (278k $kgCO_2e$), followed by PULL REORDER 90 (323k $kgCO_2e$). PUSH REORDER 90 generated the highest emissions (629k $kgCO_2e$), more than double that of the best-performing configuration.

Landfill overflow analysis demonstrates superior waste management performance of PULL strategies. PULL REORDER 50 achieved zero landfill overflow, while PULL ON DEMAND generated minimal overflow (16.1 m^3). In contrast, PUSH REORDER 90 produced substantial landfill overflow (2,014 m^3), indicating significant waste management inefficiencies.

Efficiency comparison reveals nuanced performance patterns across configurations. Collection efficiency ranges from 39.3% (PULL REORDER 50) to 58.1% (PUSH REORDER 90), while processing efficiency shows PULL strategies generally outperforming PUSH approaches, with PULL REORDER 50 achieving the highest processing efficiency (38.2%). Overall system efficiency demonstrates PUSH REORDER 90 as the top performer (19.0%), followed by PULL REORDER 90 (16.6%) and PULL REORDER 50 (14.9%).

4 Conclusion & Future Work

In this paper, we developed a DES framework for analyzing waste and material flow and transformation in reverse supply chains. The proposed model was tested on a simplified network of wood waste flows built on statistical data by ARSO. The model is capable of evaluating different operational strategies through a dual-axis framework that combines inventory policies (PUSH vs PULL) with stock management approaches (ON DEMAND, REORDER 50, REORDER 90), enabling systematic comparison of supply chain configurations under realistic stochastic conditions.

The simulation results demonstrate that PULL-based inventory strategies offer superior environmental performance while maintaining competitive service levels. The analysis reveals a fundamental trade-off between aggressive inventory accumulation and environmental sustainability, with PULL strategies achieving better demand synchronization and reduced waste overflow compared to forecast-driven PUSH approaches.

These findings suggest that waste management practitioners should prioritize demand-driven operational models over forecast-based strategies. PULL REORDER configurations emerge as particularly effective, balancing high service levels with minimal environmental impact through responsive inventory management that avoids excessive accumulation and associated disposal costs.

The framework provides valuable insights for reverse logistics network design and demonstrates the importance of aligning inventory policies with sustainability objectives in waste management systems. Future work should extend the model to incorporate multi-product interactions, dynamic pricing mechanisms, and regional policy variations to enhance applicability across diverse waste management contexts.

Acknowledgements

The research was supported by the BioLOG project: the authors are grateful for the support of National Center of Science (NCN) through grant DEC-2020/39/I/HS4/03533, the Slovenian Research and Innovation Agency (ARIS) through grant N1-0223 and the Austrian Science Fund (FWF) through grant I 5443-N. Balázs Dávid is also grateful for the support of the Slovenian Research and Innovation Agency (ARIS) through grant J1-50000. Balázs Dávid and Črtomir Tavzes gratefully acknowledge the Slovenian Research and Innovation Agency (ARIS) and the Ministry of the Economy, Tourism and Sport (MGTŠ) for the grant V4-2512.

References

[1] Agencija Republike Slovenije za okolje (ARSO). 2025. Poročila in publikacije o odpadkih. (2025). Retrieved Aug. 20, 2025 from https://www.arso.gov.si/varstvo%20okolja/odpadki/poro%C4%8Dila%20in%20publikacije/.

[2] Peter Akhator, Albert Obanor, and Anthony Ugege. 2017. Nigerian Wood Waste: A Potential Resource for Economic Development. *Journal of Applied Sciences and Environmental Management*, 21, 2, 246. DOI: 10.4314/jasem.v21i2.4.

[3] Michael Burnard, Črtomir Tavzes, Aleksandar Tošić, Andrej Brodnik, and Andreja Kutnar. 2015. The role of reverse logistics in recycling of wood products. In *Environmental Implications of Recycling and Recycled Products*. Subramanian Senthilkannan Muthu, editor. Springer Singapore, 1–30. DOI: 10.1007/978-981-287-643-0_1.

[4] George S. Fishman. 2001. *Discrete-Event Simulation*. Springer New York, New York, NY. ISBN: 978-1-4419-2892-4 978-1-4757-3552-9. DOI: 10.1007/978-1-4757-3552-9.

[5] S.M. Gupta, ed. 2013. *Reverse Supply Chains: Issues and Analysis*. (1st ed.). CRC Press. DOI: 10.1201/b13749.

[6] Dmitry Ivanov. 2025. Comparative analysis of product and network supply chain resilience. *International Transactions in Operational Research*, itor.13612. DOI: 10.1111/itor.13612.

[7] Arkadiusz Kawa. 2023. REVERSE SUPPLY CHAIN OF RESIDUAL WOOD BIOMASS. *Logforum*.

[8] C. Kögler and P. Rauch. 2018. Discrete event simulation of multimodal and unimodal transportation in the wood supply chain: a literature review. *Silva Fennica*, 52. DOI: 10.14214/sf.9984.

[9] Dorin Maier. 2023. A Review of the Environmental Benefits of Using Wood Waste and Magnesium Oxychloride Cement as a Composite Building Material. *Materials*, 16, 5, 1944. DOI: 10.3390/ma16051944.

[10] J. Owoyemi, H. Zakariya, and I. Elegbede. 2016. Sustainable wood waste management in nigeria. *Environmental & Socio-Economic Studies*, 4, 3, 1–9. DOI: 10.1515/environ-2016-0012.

[11] Kemal Parlak, Nural Yilgor, and Atakan Öngen. 2024. Hydrogen-rich syngas production from wood waste and wood waste pellet via gasification in updraft circulating fixed bed reactor. *Research Square*, (May 2024). Preprint. DOI: 10.21203/rs.3.rs-4343729/v1.

[12] Carol Prahinski and Canan Kocabasoglu. 2006. Empirical research opportunities in reverse supply chains. *Omega*, 34, 6, 519–532. DOI: 10.1016/j.omega.2005.01.003.

[13] Team SimPy. 2025. Simpy. Accessed: 2025-05-15. (2025). https://simpy.readthedocs.io/en/latest/.

[14] Statistical Office of the Republic of Slovenia. 2024. Več komunalnih odpadkov predvsem zaradi povečanja količine kosovnih odpadkov. Stat.si. Accessed: 2025-05-15. (2024). https://www.stat.si/StatWeb/News/Index/12770.

Robust (re)Design of Material Flow in Circular Networks – a Scientific Approach

Ádám Szaller*
HUN-REN Institute for Computer Science and Control
Budapest, Hungary
adamszaller@sztaki.hu

Balázs Dávid
InnoRenew CoE, UP IAM and UP FAMNIT, University of Primorska
Koper, Slovenia
balazs.david@innorenew.eu

Péter Egri
HUN-REN Institute for Computer Science and Control
Budapest, Hungary
egri@sztaki.hu

Miklós Krész
InnoRenew CoE, UP IAM and UP FAMNIT, University of Primorska
Koper, Slovenia
miklos.kresz@innorenew.eu

József Váncza
HUN-REN Institute for Computer Science and Control
Budapest, Hungary
vancza@sztaki.hu

Abstract

As sustainability concerns grow, the conventional linear supply chain model — where products move in a single direction toward consumers — is proving insufficient. In Circular Networks (CNs), forward and reverse flows of resources coexist and should be jointly optimized. Yet, CNs face significant challenges, including uncertainty in the availability, quality, and usability of end-of-life products, as well as unpredictable demand for recovered materials. These factors increase risks in processing and limit adaptability, since traditional supply chains are rigid and rely on stable partnerships. While resilience strategies and optimization methods such as stochastic or robust models have been explored, they often oversimplify real-world dynamics and neglect circular flows. To address these gaps, this paper introduces a block-based value chain model and a 5-step scientific approach.

Keywords

robust material flow planning, circular networks

1 Introduction

Flow of products and provision of services between suppliers and consumers is an integral part of modern-day business processes. This has traditionally been the domain of supply chain management, in which different actors work together to improve operational efficiency by sharing information and linking their activities. As environmental awareness became an increasingly important topic, an alternative was needed to the traditional forward product flows that only point towards the consumer. According to Dekker et al. [8], more emphasis had to be put on end-of-life recovery and the reuse of resources in the supply chain due to different driving forces (economic, legislative, and corporate citizenship). The field of reverse logistics is the integration of all activities that aim to recover resources from their final destination either to produce further value or to dispose of them properly. This can be done either in a direct manner (resale/reuse/redistribution) or through a more complex recovery process (repair, reuse, refurbish, remanufacture, retrieve, recycle, incinerate/landfill), which are described in the 10 R model of the levels of circularity [7][28]. Reverse logistics methods can be integrated into the conventional process of supply chains, forming a circular or closed-loop supply chains that account for both forward and reverse flows of resources [19]. As sustainability is becoming a key issue, the main arguments for a transition towards circular value chains are linked to the increasing scarcity of natural resources and the ecological impact of human activities. Such a sustainable mindset has become crucial for the supply chains of several significant materials, such as steel and aluminium, plastics [23], bottles [12], textiles, paper or wood. Despite the increasing need of circularity, there are still numerous research and practical gaps in literature [27]. However, there are several issues that make the optimal design of material flows in such Circular Networks (CNs) exceptionally challenging. Besides the hard-to-predict demand for products and components, the potential reuse of end-of-life products and production residues introduces new sources of uncertainty on the supply side. Not only is the available quantity for end-of-life products unknown in advance, but their quality and usability are also questionable, introducing yield uncertainty to the waste processing. The same is true for production residues; while types and quantities can be estimated for each production process, their ultimate quality will depend on the original input materials to their corresponding processes. The recovery of the used products is also a challenging process with difficult, complex and expensive steps. For example, recovered wooden products could require sorting, removing of tacks, nails, paint and even bugs. There might be possible different ways to reuse these products which degrade the material quality to a different extent, such as using as solid wood, particle board, fiber-based product, chemical product or even incineration as an energy source [15]. Moreover, supply chains are traditionally linear and static, and their entities usually cooperate with their well-established connections, always relying on the same pool of actors and processes [29]. Because of this, their adaptation to rapid changes is inefficient, and disruptions either on supply- or demand-side, or in the processes of the system itself can cause issues.

2 Research objectives

Research of dynamic and resilient supply chains has started in recent years [16], and it has been identified that dynamics control can be classified into proactive (designing in a redundant and flexible way) and reactive (recovery methods in case of disruptions) stages. While there are several different categorizations of supply chain disruptions, the ones that appear consistently in every group are process risks, demand risks and supply risks [18]. Such uncertainties in circular supply chain planning are usually tackled with stochastic and robust optimization approaches, such as chance constraints, scenario-based optimization, data-driven robust optimization and maximin optimization [2][20][25][30]. In addition to the various sources of uncertainty, there are multiple, often contradicting objectives of the supply chain planning problem. Supply chain planning traditionally aims at satisfying the demand at minimal cost, but this does not necessarily meet other objectives, such as an increased proportion of recycled materials or a minimal value loss during the recovery process (the so-called biomass cascading principle [15]). Furthermore, conflicting objectives can also arise from the presence of multiple stakeholders in the circular chain [13].

Due to these complexity and uncertainty issues, the performance of the circular business models must be extensively explored before implementation, i.e., with simulation models [21]. Despite the growing need for models and tools for assessment of circularity, simulation approaches for circular supply chains are still scarce in the literature and in practice. According to a recent systematic review, most models apply optimization or multi criteria decision making models which greatly outnumber simulation approaches [22].

In our previous joint work, we have focused on robust design of waste wood collection and transportation networks [10], which we intend to continue with the modeling, planning and validating robust supply chains considering multiple optimization criteria. We have also developed a robust two-stage model for planning and adapting the supply chain material flow to the disturbances [11]. On the level of mathematical abstraction, supply chains can be regarded as multi-commodity flow networks where the flow (resources) is transformed into the nodes of the system. While the theory of robust and adaptive network flows has been studied in the past [4][5], and they have been proven to be NP-hard problems [9], the application of this theory to actual supply chain networks has been limited. Even recent studies [1][14] simplify the real-world aspects and dynamics of supply chains significantly to a level where their results—while being important theoretical contributions—cannot be used directly for real-world networks. Moreover, none of these studies consider reverse or circular flows to our knowledge, they only deal with the traditional forward case.

Based on the above-mentioned research challenges, the objectives to be solved with the scientific approach presented in this paper are the following:

- Address shortcomings of existing approaches for circular supply chains.
- Design a novel, general approach for modeling material flow in reverse and circular networks.
- Develop a robust design method for material flow in circular supply chains.
- Analyze the impact of disruptions (on demand, supply, or process side) on these robust flows.
- Create efficient methods for disruption management and material flow redesign in the network.
- Ensure general applicability of the methods developed across various circular industrial use-cases.

3 Scientific approach

3.1 Block-based model representation

First, this paper proposes a *modular modeling concept* that can be applied to represent any supply chain in general, incorporating both forward and reverse material flows. With the help of this concept, a multi-commodity flow network can be constructed using the existing entities of multiple independent value chains, and material flow and transformation can be optimized in this network. This representation is also capable of accommodating cascade or reuse possibilities and residual material flows. The value chains are deconstructed into independent entities called *blocks*. A block is a universal representation of a series of different activities inside a value chain, defining the required material inflow and outflow, as well as the material transformation happening during the corresponding processes. Material flows are described by a *vector* of features, and outflow(s) can be directed to be inputs of another block if the features of the flow matches the input requirements of the block. This feature vector can include all relevant material characteristics (depending on the application), as well as economic, social and environmental costs and parameters. A general outline of this modeling approach is given in Figure 1.

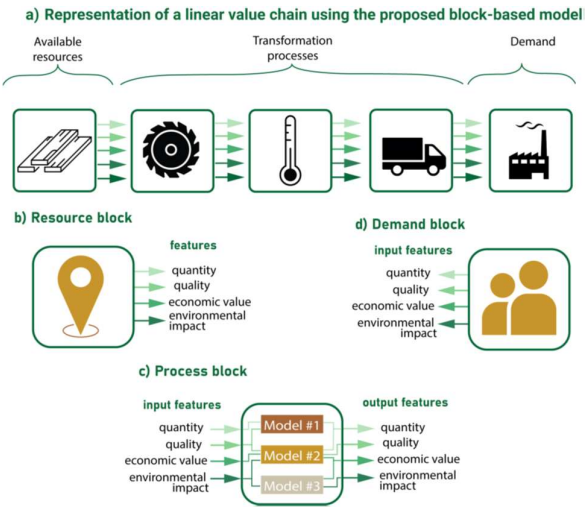


Figure 1: Graphical representation of the block-based model elements

The modeling concept distinguishes between three different block types: resource, process, and product/demand blocks. This corresponds to most value chains, where initial resources are transformed by a sequence of processes and yield products that

satisfy some customer demand. *Resource blocks* (Figure 1.b) provide data about the initially available resources in the network and can represent both virgin and reused materials. *Process blocks* (Figure 1.c) can represent any activity inside the chain (production, transportation, storage, etc.) and also define the transformation of the material properties during the activities they represent, as well as any additional outputs (residuals, etc.) they might have. The chain ends with *demand blocks* (Figure 1.d), which represent the needs in material quantity/quality of the end-users (customers, factories, etc.). With this modeling approach, existing value chains (Figure 1.a) can be de-linearized and connected into a common supply chain networks by combining all the possible directions of resource flow (both primary and residual) and transformation processes. Using the above modeling approach, a supply network can be constructed using the entities (processes) of multiple existing value chains as nodes, where all possible material flow options between these actors can be represented. Figure 2 shows an example for four different linear value chains (I-IV), with the potential reverse flow options of residual materials from the processes of one value chain used as inflow in the processes of other chains.

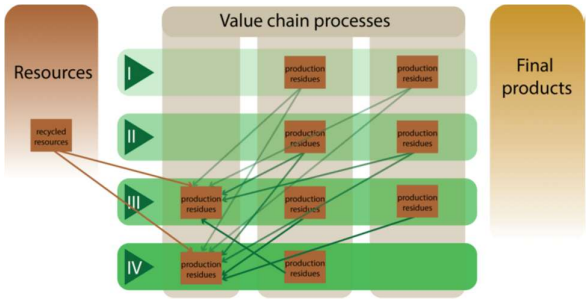


Figure 2: Circular resource flow in the supply network

3.2 Research steps

Based on the above approach, the following research steps are proposed.

First, a novel and universally applicable *mixed-integer linear programming model* must be formulated for circular material flow optimization using the flexible block-based network representation. The major challenge of this step is incorporating flow transformation functions within the (process) nodes of the network, corresponding to the production processes of the given entities. These possible production choices for flow transformation must be reflected in the decision variables managing material flow between the nodes of the network, as well as their connected constraints. The model must consider multiple objectives (e.g. operational costs and environmental impacts).

Next, a method for *optimal design of robust material flow optimization* must be created for this network, e.g. by adapting a selected method from literature. The extension of the robust modelling approach proposed by Bertsimas and Sim [6], and the two-stage robust network flow approaches such as the one by Atamturk and Zhang [3] and especially by Simchi-Levi et al. [26] are considered as promising starting points. The developed method must take into account possible uncertainties on the supply side (both for virgin and reused materials), demand side, and connected to the yield uncertainty with regards to residual materials of the processes in the network.

Following the design of the material flow, a novel method must be developed for *disruption management and recovery* after unforeseen events to redesign the flow to still satisfy the original constraints as fully as possible. These unforeseen events can include changes in the original supply (both for virgin and reused materials), demand, or in the production processes of the system, leading to a lower yield of reverse materials to be reused. The three types of disruptions must be examined independently, as well as simultaneously, studying their combined effect on the system. Literature for optimization methods in reactive disruption management (re-planning after an unforeseen event) for supply chains is quite limited and instance-based even for traditional forward chains [17]. Metaheuristic approaches such as Tabu search or Simulated Annealing, as well as Large Neighborhood Search are promising approaches in this topic. Various new neighborhood structures can be constructed for these algorithms and new diversification methods can be implemented for exploring new regions of the neighborhood. The hybridization possibilities of metaheuristic and exact approaches must also be studied, such as aiding the branch and bound solution of the mathematical models with a rounding heuristic or solving the subproblems of a large neighborhood search by exact methods. The applicability of Genetic Algorithms and Ant Colony Optimization must also be investigated to find a suitable and efficient solution.

To validate and further develop the above-mentioned methodologies, a *real-world use case* is proposed using e.g. the wood industry as a basis. Wood is an ideal example for a circular material, as both production residues and recovered end-of-life products can be reused as inputs for other lower value wood products. Actors and processes from the value chains of multiple wooden products (e.g. furniture, panels, packaging) could be modeled using the block-based network, and the developed models and methods could be validated on this use-case.

As a final step, simulation methods could be applied to further evaluate the efficiency of the disruption management methods on large amount of realistic, generated data (for which the related literature and experience from industry can give a basis).

4 Conclusion and future work

In the paper, a scientific approach regarding robust (re)design of material flow in circular networks was introduced. First, a block-based representation – that takes into account several aspects of forward and reverse material flows by defining resource, process, and product/demand blocks, and the vector of features connected to each block – was presented to model the circular value chain. Then, a 5-step scientific approach was suggested for the (re)design process: 1) creating mixed-integer linear programming model, 2) designing and optimizing robust material flow, 3) developing a methodology for disruption management and recovery, 4) validating on a real-world use case, 5) creating a simulation model and testing with generated data.

Acknowledgements

This research has been supported by the TKP2021-NKTA-01 NRDIO grant on "Research on cooperative production and logistics systems to support a competitive and sustainable economy". The research was supported by the BioLOG project: the authors are grateful for the support of National Center of

Science (NCN) through grant DEC-2020/39/I/HS4/03533, the Slovenian Research and Innovation Agency (ARIS) through grant N1-0223 and the Austrian Science Fund (FWF) through grant I 5443-N. Balázs Dávid and Miklós Krész gratefully acknowledge the Slovenian Research and Innovation Agency (ARIS) and the Ministry of the Economy, Tourism and Sport (MGTŠ) for the grant V4-2512. Balázs Dávid is grateful for the support of the Slovenian Research and Innovation Agency (ARIS) through grant J1-50000.

References

[1] Heiner Ackermann, Erik Diessel, and Sven O. Krumke. “Robust flows with adaptive mitigation”. In: *EURO Journal on Computational Optimization* 9 (2021), p. 100002. issn: 2192-4406.

[2] Mohammad Saeid Atabaki, Mohammad Mohammadi, and Bahman Naderi. “New robust optimization models for closed-loop supply chain of durable products: Towards a circular economy”. In: *Computers & Industrial Engineering* 146 (2020), p. 106520.

[3] Alper Atamtürk and Muhong Zhang. “Two-Stage Robust Network Flow and Design Under Demand Uncertainty”. In: *Operations Research* 55.4 (2007), pp. 662‑673.

[4] Dimitris Bertsimas, Ebrahim Nasrabadi, and Sebastian Stiller. “Robust and Adaptive Network Flows”. In: *Operations Research* 61.5 (Oct. 2013), pp. 1218‑1242.

[5] Dimitris Bertsimas and Melvyn Sim. “Robust discrete optimization and network flows”. In: *Mathematical Programming* 98.1‑3 (Sept. 2003), pp. 49‑71.

[6] Dimitris Bertsimas and Melvyn Sim. “The Price of Robustness”. In: *Operations Research* 52.1 (Feb. 2004), pp. 35‑53.

[7] Jacqueline Cramer. “The Raw Materials Transition in the Amsterdam Metropolitan Area: Added Value for the Economy,Well-Being, and the Environment”. In: *Environment: Science and Policy for Sustainable Development* 59.3 (2017), pp. 14‑21.

[8] Rommert Dekker, Moritz Fleischmann, Karl Inderfurth, and Luk N. Van Wassenhove, eds. *Reverse Logistics: Quantitative Models for Closed-Loop Supply Chains*. Springer Berlin Heidelberg, 2004. doi: 10.1007/978- 3-540-24803-3. url: https://doi.org/10.1007/978-3-540-24803-3.

[9] Yann Disser and Jannik Matuschke. “The complexity of computing a robust flow”. In: *Operations Research Letters* 48.1 (2020), pp. 18‑23.

[10] Péter Egri, Balázs Dávid, Tamás Kis, and Miklós Krész. “Robust facility location in reverse logistics”. In: *Annals of Operations Research* 324.1-2 (2021).

[11] Péter Egri and Tamás Kis. “Robust two-stage optimisation in biomass supply chains”. In: *International Journal of Production Economics 285, 109623 (2025)*

[12] Carmen Liping Fernández-Arribas, Borja Ponte, and Isabel Fernández. “Shaping closed-loop supply chain dynamics: Mitigating the bullwhip effect and improving customer satisfaction in production systems with material reuse”. In: *Computers & Industrial Engineering* 195 (2024), p. 110407.

[13] Eilidh J. Forster, John R. Healey, Gary Newman, and David Styles. “Circular wood use can accelerate global decarbonisation but requires cross-sectoral coordination”. In: *Nature Communications* 14.6766 (2023).

[14] Supriyo Ghosh and Patrick Jaillet. “An iterative security game for computing robust and adaptive network flows”. In: *Computers & Operations Research* 138 (2022), p. 105558. issn: 0305-0548

[15] Benjamin Hague, Jana Kozáková, and Andrea Veselá. *Closing the loop on wood ‑ Circular Bioeconomy Opportunities in the Value Chain for Forest Products and Wood in Czechia*. Tech. rep. Institute of Circular Economy (INCIEN), 2023.

[16] Dmitry Ivanov. *Structural Dynamics and Resilience in Supply Chain Risk Management*. Vol. 265. International Series in Operations Research & Management Science. Springer, 2018.

[17] Dmitry Ivanov, Alexandre Dolgui, Boris Sokolov, and Marina Ivanova. “Literature reviewon disruption recovery in the supply chain”. In: *International Journal of Production Research* 55.20 (2017), pp. 6158‑6174.

[18] Korina Katsaliaki, Panagiota Galetsi, and Sameer Kumar. “Supply chain disruptions and resilience: a major review and future research agenda”. In: *Annals of Operations Research* 319 (2022), pp. 965‑1002.

[19] Nima Kazemi, Nikunja Mohan Modak, and Kannan Govindan. “A review of reverse logistics and closed loop supply chain management studies published in IJPR: a bibliometric and content analysis”. In: *International Journal of Production Research* 57.15-16 (2019), pp. 4937‑4960.

[20] Alireza Khalili-Fard, Fatemeh Sabouhi, and Ali Bozorgi-Amiri. “Data-driven robust optimization for a sustainable steel supply chain network design: Toward the circular economy”. In: *Computers & Industrial Engineering* 195 (2024), p. 110408.

[21] Kasper P.H. Lange, Gijsbert Korevaar, Inge F. Oskam, Igor Nikolic, and Paulien M. Herder. “Agent-based modelling and simulation for circular business model experimentation”. In: *Resources, Conservation & Recycling Advances* 12 (2021), p. 200055.

[22] Haitham A. Mahmoud, Sarah Essam, Mohammed H. Hassan, and Arafa S. Sobh. “Modeling circular supply chains as an approach for waste management: A systematic review and a conceptual framework”. In: *Journal of Engineering Research* in press (2025).

[23] Yasmine Morjéne, Nadia Ndhaief, and Nidhal Rezg. “Optimization of production batches in a circular supply chain under uncertainty”. In: *IFAC PapersOnLine* 55.10 (2022), pp. 1752‑1757.

[24] Gianfranco Pedone, József Váncza, and Ádám Szaller. “Exploring hidden pathways to sustainable manufacturing for cyber-physical production systems”. In: *Heliyon* 10.8 (2024), e29004. issn: 2405-8440.

[25] Youngchul Shin, Gwang Kim, and Yoonjea Jeong. “Robust closed-loop supply chain model with return management system for circular economy” . In: *Computers & Industrial Engineering* in press (2025)

[26] David Simchi-Levi, He Wang, and Yehua Wei. “Constraint Generation for Two-Stage Robust Network Flow Problems”. In: *INFORMS Journal on Optimization* 1.1 (Winter 2019), pp. 49‑70.

[27] Emilia Taddei, Claudio Sassanelli, Paolo Rosa, and Sergio Terzi. “Circular supply chains theoretical gaps and practical barriers: A model to support approaching firms in the era of industry 4.0”. In: *Computers & Industrial Engineering* 190 (2024), p. 110049. issn: 0360-8352.

[28] Tharaka De Vass, Alka Ashwini Nand, Ananya Bhattacharya, Daniel Prajogo, Glen Croy, Amrik Sohal, and Kristian Rotaru. “Transitioning to a circular economy: lessons from the wood industry”. In: *The International Journal of Logistics Management* 34.3 (2023), pp. 582‑610.

[29] Weiqi Yan, Nan Li, and Xin Zhang. “Enhancing supply chain management in the physical internet: a hybrid SAGA approach”. In: *Scientific Reports* 13.1 (2023), p. 21470.

[30] Yuchen Zhao, Mohsen Roytvand Ghiasvand, and Babak Mohamadpour Tosarkani. “Balanced Uncertainty Sets for Closed-Loop Supply Chain Design: A Data-Driven Robust Optimization Framework with Fairness Considerations”. In: *Expert Systems with Applications* in press (2025)

Harvest plan generation in precision agriculture

Štefan Horvat*
stefan.horvat@um.si
Faculty of Electrical Engineering and Computer Science
Maribor, Slovenia

Damjan Strnad
Faculty of Electrical Engineering and Computer Science
Maribor, Slovenia
damjan.strnad@um.si

Domen Mongus
Faculty of Electrical Engineering and Computer Science
Maribor, Slovenia
domen.mongus@um.si

Matej Brumen
Faculty of Electrical Engineering and Computer Science
Maribor, Slovenia
matej.brumen@um.si

Abstract

One of many practical applications of precision agriculture is planning and optimization of harvest times, which requires the knowledge of crop maturity from which harvest time can be predicted. We present a simple procedure where we first obtain the Green Normalized Difference Vegetation Index (GNDVI) index values from Sentinel-2 imagery for a certain time period before the crop enters the last growth cycle. The timeseries is analysed, and a simple linear and polynomial regression model are fitted. Extrapolation is used to calculate the intersection with the time axis, which acts as the harvest date prediction. In the next step we rearrange the fields based on crop maturity and create a harvesting plan that utilizes the combine harvester that maximizes the harvest area. We validated the results using actual harvest dates, and found that the polynomial regression results matched closer to the actual harvest dates than those of the linear regression. This approach offers a scalable solution to harvest date prediction and plan generation, but further adjustments and a larger dataset are needed to improve the performance for practical application.

Keywords

precision agriculture, harvest maps, linear regression, polynomial regression

1 Introduction

The development of Earth observation technologies, such as Sentinel-2 satellite network, remote sensing and GPS, has enabled their widespread integration in to new areas. One of them is precision agriculture, which enables site specific interventions without physical presence and makes agriculture more sustainable, cost effective and increases the quality of the produce. This aligns well with the common agricultural policy (CAP) of the EU [2, 5, 13]. There are numerous interesting applications in precision agriculture such as creating optimal fertilization plans [15], monitoring crop health and development [10], estimating yield [17], creating harvest plans [11] and most recently the popular trend of artificial intelligence integration.[7, 17].

The problem of optimal harvest time deals with determining the exact time of crop maturity. It enables mass detection of fields or parts of fields that are ready or will be ready for harvest in the near future. This enables efficient planning and maximum utilization of resources, such as agricultural equipment and labor. In this work, we present a method that utilises remote sensing data, such as Sentinel-1 and Sentinel-2 imagery. Sentinel-1 uses synthetic aperture radar (SAR) and enables constant monitoring regardless of the weather conditions and the time of day. The resulting images usually contain a lot of backscatter that can be hard to interpret and analyse [3]. Sentinel-2 imagery is multispectral and its main advantage over Sentinel-1 is the ability to capture spectral reflectance, which is a lot more useful for large scale crop analysis [4]. Most existing works focus on detecting and monitoring specific growth cycles of crops to detect and predict harvest dates using Sentinel-1. In [1, 9, 14] authors analyse SAR backscatter signatures to detect crop growth cycles. In [12, 17] machine learning models are first trained and than used to predict the harvest dates. When using Sentinel-2 imagery the most straightforward approach is to gather timeseries for an observed area and analyse the time series for any abrupt changes. These can signify significant events happening on the field, which is especially useful for post analysis [16].

In this paper, we present a simple algorithm for harvest time prediction and creation of a simple harvesting plan which simulates the anticipated harvest. Section 2 describes the methodology that was used in the approach, section 3 evaluates the results and section 4 concludes with potential future research directions.

2 Methodology

The process of harvest time optimization and harvest map generation consists of two steps. In the first step, the field data is gathered and predictions of optimal time for harvesting are made. In the second step, an optimal harvest plan is created that aims to maximize the combine harvester load based on the field maturity from the previous step.

2.1 Prediction of harvest times

The *HarvestPrediction* algorithm accepts three inputs, a list of fields in GeoPackage (GPKG) format, which we denote as F. The package contains metainformation about the fields, such as crop type and field area. We only keep the fields that contain wheat or barley. The second input is the date (we will refer to it as the starting date), for which the harvest plan is being created. A 30-day window preceding the starting date is used to obtain the historic samples for each field in F from Sentinel-2 data. All available Sentinel-2 images from the historic window are first cropped to field dimensions, and only images with cloud cover lower than 15% are retained. For each field image, the GNDVI was calculated as:

$$GNDVI = \frac{NIR - G}{NIR + G} \tag{1}$$

Information Society 2025, 6–10 October 2025, Ljubljana, Slovenia

where NIR represents the near-infrared band and G represents the green band. The GNDVI vegetation index is sensitive to chlorophyll content, which helps at identifying late growth stages [6]. The GNDVI values were clamped to be between 0 and 1. For each valid timestamp, the mean of GNDVI over the field was calculated and saved to a corresponding data structure. After preprocessing, each field had a time series of mean GNDVI values, which were used in the forecasting step. An example of a GNDVI time series can be seen in Figure 1 .

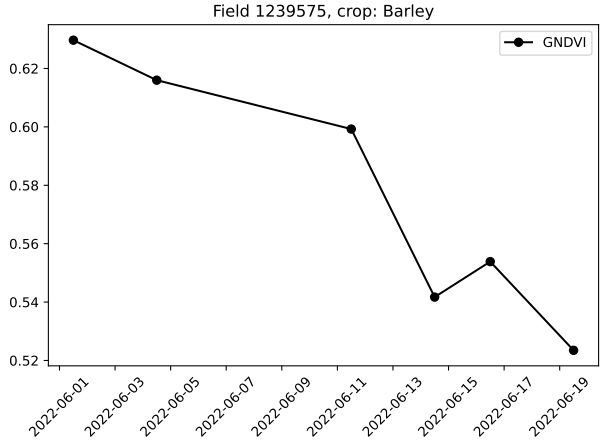


Figure 1: Example of a GNDVI timeseries for a field with barley.

To determine the optimal harvest time, we used two approaches. The first approach is a simple linear regression, and the second is a polynomial regression with the degree as a parameter.

2.2 Predictions with linear regression

Because there exists significant correlation between GNDVI values at consecutive timestamps, linear regression is the most natural approach. The dependent variable is the value of GNDVI, the independent variable is time. The timestamps were converted to POSIX format for easier handling of calculations. The relationship between dependent and independent variable can be defined using the following formulation:

$$y_t = \beta_0 + \beta_1 x_t + \epsilon_t \tag{2}$$

where y_t denotes the dependent variable, x_t denotes the independent variable, coefficient β_0 denotes the intercept, β_1 denotes the slope and ϵ_t denotes the deviation from the fitted line [8]. The main application here is to determine the optimal time to harvest. Once the model is fitted, we can use extrapolation to determine, where the regression line intersects the line of user-specified threshold value y_f (which we will refer to as crop maturity threshold):

$$x_f = \frac{y_f - \beta_1}{\beta_0} \tag{3}$$

2.3 Predictions with polynomial regression

Sometimes the changes in values between consecutive timestamps exhibit a curved pattern. To capture such curvature in the data, we used a polynomial regression model, defined as:

$$y_t = \beta_0 + \beta_1 x_t + \beta_2 x_t^2 + \ldots + \beta_n x_t^n \tag{4}$$

where n denotes the degree of the polynomial. After fitting the polynomial to the data, the next step is to calculate, where the polynomial intersects the maturity threshold y_f, which means that we need to solve the polynomial for that value and find the real roots. The downside here is the need to choose the polynomial degree and the right root values in case of multiple solutions. After experimenting with n in the range from 2 to 4 we found n of 2 to be sufficient. An example of fitting both models to a different GNDVI timeseries can be seeen in Figure 2. In the bottom Figure, the quadratic polynomial regression was used with two solutions for the root. In our methodology, we select the solution that is the nearest to the starting date.

Algorithm 1 Harvest plan generation

```
function HarvestPrediction(F, StartingDate, y_f)
        ▷ F: GeoPackage which contains a list of fields
    ▷ StartingDate: Date for which the harvest plan will be created
                                        ▷ y_f: Maturity threshold

    n ← 2
    for f_i in F do
        MetaInfo ← GetMetainfo(f_i)
        TS ← GetSentinel2History(StartingDate)
        LinRegInt ← LinearRegression(TS, y_f)
        PolyRegInt ← PolyRegression(TS, n, y_f)
        UpdateGeoPackageMetainfo(f_i)
    end for
end function

function HarvestPlan(F, HarvestDate, Z)
        ▷ F: GeoPackage which contains a list of fields
            ▷ Z: Maximum area to harvest per day
        ▷ HarvestDate: Date when the harvest will begin

    SortByMaturity(F)
    tempDate ← HarvestDate
    while F <> empty do
        MetaInfo ← GetMetainfo(f_i)
        Q ← CreateAndFillQueue(tempDate,F)
        tempDate ← tempDate + 1
    end while
end function
return F
```

2.4 Creating the harvest plan

The second step is to create an actual harvest plan that uses a single combine harvester for all fields in the list. The *HarvestPlan* algorithm accepts three inputs. The first input is a GPKG of fields with updated metainformation from the previous step, the second input is the harvest date, when the combine is going to start the actual harvesting, and the third is Z, which denotes total harvest area a combine harvester can do every day. The fields in F are first sorted by maturity. For every day the combine harvester is available, we add the fields from F to a queue Q as long as their total area is smaller or equal to Z. The process can also tie-break fields, so it is not necessary that the whole field is harvested in one day. If fields remain in F, we create new queue for the next day and repeat the process until F is empty. After assigning the harvest dates to the fields, we store the fields back to the initial

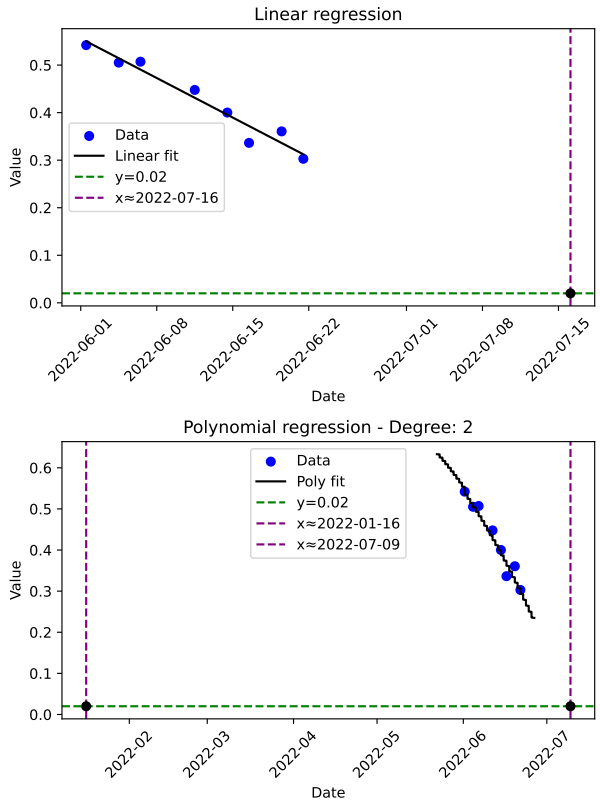


Figure 2: An example of fitting a linear model (top) and a quadratic polynomial model (bottom) for a field with wheat. Black dots mark the prediction which can be found on the intersection between the maturity threshold $y_f = 0.02$ and the time axis.

GeoPackage F, populate the fields with new metainformation and return the result to the user. The algorithms used in the process are described in 1.

3 Results

A collection of eighteen fields located in northwestern Slovenia were used to evaluate the process. There were total of ten samples with barley and eight samples with wheat. Sometimes the actual harvests happened before or after the optimal time for harvesting because of bad weather or disruptions in resources (farming equipment breaks, organisational problems due to labor shortage). The algorithm was intended to be used near the start of the actual harvest season, which is usually determined by numerous exogenous factors. Our method assumes perfect conditions so it does not take those limitations into account.

The vegetation index, the historic window length and maturity threshold y_f were all chosen in conjunction with agricultural experts. We should point out that the historic window can be increased, but then the preprocessing step takes more time. When doing extrapolation, both linear and quadratic polynomial regression diverge far from the expected results or in the case of quadratic regression there can be no real solutions. To avoid this, we clamp the harvest dates to the period between the 20th of June and 31st of July, which are reasonable bounds for barley and wheat.

Table 1: Mean difference and std. deviation between predicted and actual harvest times in days for linear regression.

Crop	Mean	Std. deviation	Accuracy
Barley	15.10	12.51	5/10
Wheat	23.10	20.79	2/8

Table 2: Mean difference and std. deviation between predicted and actual harvest times in days for quadratic polynomial regression.

Crop	Mean	Std. deviation	Accuracy
Barley	5.70	15.71	6/10
Wheat	5.75	11.12	6/8

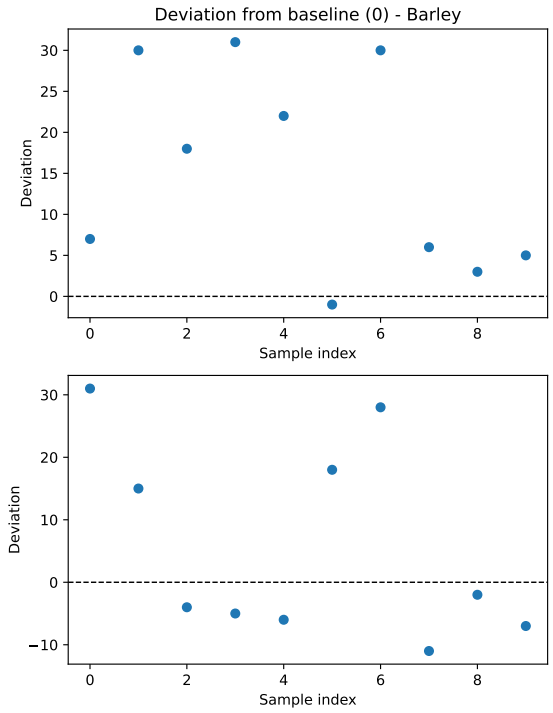


Figure 3: Differences (in days) between predicted and actual harvest dates on barley fields for linear (top) and quadratic polynomial (bottom) model.

The comparison of harvest dates predicted with linear and polynomial regression compared to actual harvest dates can be seen in Tables 1 and 2. The polynomial regression has a lower average error than the linear regression for both crops, so its predictions were closer to the actual harvest dates in both cases. Figures 3 and 4 show the residuals, i.e. the differences in days between actual and predicted harvest days, for barley and wheat fields, respectively. The opinion of agricultural experts is that all forecasts, that are within a ten day window from the actual

harvesting date can be considered acceptable, which is represented with the accuracy column. The polynomial regression outperforms linear regression, as it scored an additional correct sample with barley and four additional samples with wheat. We can see that some predictions are off by multiple weeks. One of the reasons can be attributed to the small quantity of real samples.

4 Conclusion

The described procedure is a simple and scalable solution for prediction of harvest dates and simple harvest plan generation. It utilizes Sentinel-2 imagery to obtain a timeseries before the nearing harvests and predicts harvest dates based on timeseries patterns. We found that polynomial regression did a better job at making those predictions and that the generated harvest plan can be organised as a simulation of how the harvest could be performed. The current method does not take into account real world limitations, such as weather events, which could prove vital if included in the harvest plan generation. Additional data would also be of great importance because with a large enough dataset a supervised machine learning model could be build that could predict the harvest dates with greater accuracy.

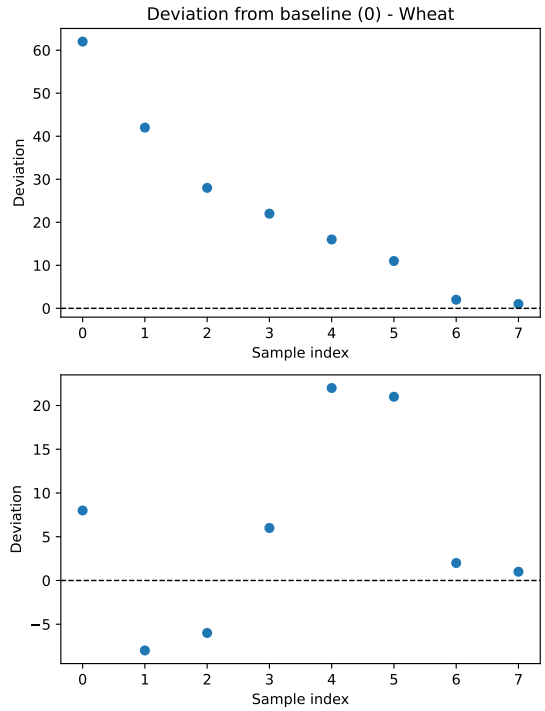


Figure 4: Differences (in days) between predicted and actual harvest dates on wheat fields for linear (top) and quadratic polynomial (bottom) model.

5 Acknowledgments

This research has received funding from the HORIZON Europe programme through project PrAEctiCe under grant agreement no. 101084248 and Research Programme P2-0041. I would also like to thank ITC for providing the actual harvest dataset.

6 References

[1] Arturo G. Cauba, Roshanak Darvishzadeh, Michael Schlund, Andrew Nelson, and Alice Laborte. 2025. Estimation of transplanting and harvest dates of rice crops in the philippines using sentinel-1 data. *Remote Sensing Applications: Society and Environment*, 37, 101435. DOI: https://doi.org/10.1016/j.rsase.2024.101435.

[2] European Commission. 1962. Retrieved August 21, 2025 from https://agriculture.ec.europa.eu/common-agricultural-policy_en.

[3] ESA. 2014. Sentinel-1. Retrieved August 21, 2025 from https://dataspace.copernicus.eu/data-collections/sentinel-data/sentinel-1.

[4] ESA. 2015. Sentinel-2. Retrieved August 21, 2025 from https://www.esa.int/Applications/Observing_the_Earth/Copernicus/Sentinel-2.

[5] Robin Gebbers and Viacheslav I. Adamchuk. 2010. Precision agriculture and food security. *Science*, 327, 5967, 828–831. eprint: https://www.science.org/doi/pdf/10.1126/science.1183899. DOI: 10.1126/science.1183899.

[6] Anatoly A. Gitelson, Yoram J. Kaufman, and Mark N. Merzlyak. 1996. Use of a green channel in remote sensing of global vegetation from eos-modis. *Remote Sensing of Environment*, 58, 3, 289–298. DOI: https://doi.org/10.1016/S0034-4257(96)00072-7.

[7] Garima Gupta. 2025. Applications of ai in precision agriculture. *Discover Agriculture*. DOI: 10.1007/s44279-025-00220-9.

[8] Rob Hyndman and G. Athanasopoulos. 2021. *Forecasting: Principles and Practice*. English. (3rd ed.). OTexts, Australia.

[9] Olena Kavats, Dmitriy Khramov, Kateryna Sergieieva, and Volodymyr Vasyliev. 2019. Monitoring harvesting by time series of sentinel-1 sar data. *Remote Sensing*, 11, 21. DOI: 10.3390/rs11212496.

[10] James Y. Kim. 2024. Open-source software for satellite-based crop health monitoring. *Journal of Biosystems Engineering*. DOI: 10.1007/s42853-024-00242-z.

[11] Emogine Mamabolo, Makgabo Johanna Mashala, Ephias Mugari, Tlou Elizabeth Mogale, Norman Mathebula, Kabisheng Mabitsela, and Kwabena Kingsley Ayisi. 2025. Application of precision agriculture technologies for crop protection and soil health. *Smart Agricultural Technology*, 12, 101270. DOI: https://doi.org/10.1016/j.atech.2025.101270.

[12] Gordan Mimić, {Amit Kumar} Mishra, Miljana Marković, Branislav Živaljević, Dejan Pavlović, and Oskar Marko. 2025. Machine learning-based harvest date detection and prediction using sar data for the vojvodina region (serbia). English. *Sensors*, 25, 7, (Apr. 2025). Publisher Copyright: © 2025 by the authors. DOI: 10.3390/s25072239.

[13] David J. Mulla. 2013. Twenty five years of remote sensing in precision agriculture: key advances and remaining knowledge gaps. *Biosystems Engineering*, 114, 4, 358–371. Special Issue: Sensing Technologies for Sustainable Agriculture. DOI: https://doi.org/10.1016/j.biosystemseng.2012.08.009.

[14] Ali Nasrallah, Nicolas Baghdadi, Mohammad El Hajj, Talal Darwish, Hatem Belhouchette, Ghaleb Faour, Salem Darwich, and Mario Mhawej. 2019. Sentinel-1 data for winter wheat phenology monitoring and mapping. *Remote Sensing*, 11, 19. DOI: 10.3390/rs11192228.

[15] Dorijan Radočaj, Mladen Jurišić, and Mateo Gašparović. 2022. The role of remote sensing data and methods in a modern approach to fertilization in precision agriculture. *Remote Sensing*, 14, 3. DOI: 10.3390/rs14030778.

[16] Jan Verbesselt, Rob Hyndman, Achim Zeileis, and Darius Culvenor. 2010. Phenological change detection while accounting for abrupt and gradual trends in satellite image time series. *Remote Sensing of Environment*, 114, 12, 2970–2980. DOI: https://doi.org/10.1016/j.rse.2010.08.003.

[17] George Worrall, Jasmeet Judge, Kenneth Boote, and Anand Rangarajan. 2023. In-season crop phenology using remote sensing and model-guided machine learning. *Agronomy Journal*, 115, 3, 1214–1236. eprint: https://acsess.onlinelibrary.wiley.com/doi/pdf/10.1002/agj2.21230. DOI: https://doi.org/10.1002/agj2.21230.

Reducing #SAT to k-clique enumeration

Sándor Szabó
sszabo7@hotmail.com
University of Pécs
Pecs, Hungary

Bogdán Zaválnij
bogdan@renyi.hu
HUN-REN Alfred Renyi Institute of Mathematics
Budapest, Hungary

Abstract

Complexity theory states that some complex problem can be reduced to another problem. The reduction from SAT to k-CLIQUE was already introduced by Karp. Here we propose an alternative graph reformulation, that can be efficiently used for solving the #SAT problem.

Keywords

model counting, combinatorial optimization, k-clique, SAT, reformulation

1 Introduction

That is actually the base of complexity theory that some complex problem can be reduced to another problem. The reduction from SAT to k-CLIQUE was already introduced by Karp [3]. Thus most textbooks that include chapter on NP-completeness include reduction from 3-SAT to k-CLIQUE, as for example [2, pp. 1087–1089]. A small textbook like example pictured in Figure 1. Here the SAT example consists of three clauses, each consisting of three literals. Namely

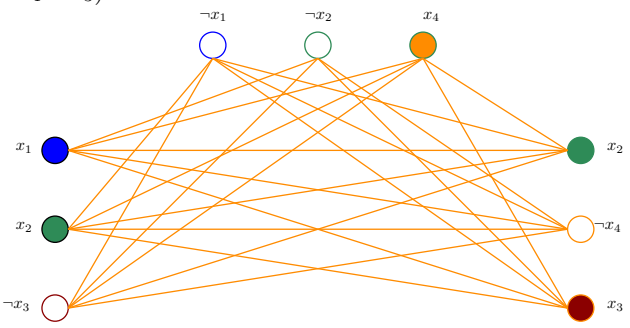
$$\Phi = C_1 \wedge C_2 \wedge C_3,$$
$$C_1 = x_1 \vee x_2 \vee \neg x_3,$$
$$C_2 = \neg x_1 \vee \neg x_2 \vee x_4,$$
$$C_3 = x_2 \vee \neg x_4 \vee x_3.$$

In short, we construct a graph. The number of the nodes is equal to number of the literals in the SAT problem. Each node associated with a certain literal is not adjacent to any node adjacent to a literal in the same clause; and two nodes associated with literals l_1, l_2 from different clauses are not adjacent if $l_1 \wedge l_2$ is a contradiction. All other pairs are adjacent. The constructed graph is k-partite where k is the number of clauses, and each k-clique is giving us a solution of the proposed SAT problem.

There are obviously several ways to accomplish such reformulation. We will introduce one, that may possibly occurred elsewhere before. There is a small problem with the textbook way, and that causes two different drawbacks. The problem is basically that two different k-cliques can encode the same solution of the SAT problem. The reason behind another reformulation is twofold. First, for the SAT reformulation use a symmetry breaking and produce a graph with less edges and less k-cliques. Second, the textbook reformulation is hardly usable for #SAT but the proposed method is. This short paper is focusing only on the #SAT problem.

Information Society 2024, 7–11 October 2024, Ljubljana, Slovenia

Figure 1: $\Phi = (x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_4) \wedge (x_2 \vee \neg x_4 \vee x_3)$



2 Proposed reformulation

We will describe a reduction of the SAT problem to the k-clique problem which differs from the commonly encountered reduction. The new k-clique reformulation of the SAT problem has the added benefit that all the possible solutions of the SAT problem and all the possible solutions of the k-clique problem are in a well defined connection. Consequently, the k-clique problem can be used to list all possible solutions of the SAT problem.

Let $C_1, \ldots, C_k$ be clauses over the propositional variables $x_1, \ldots, x_n$. For our considerations we need a more detailed description of the clauses. Namely, for each i, $1 \leq i \leq k$ set

$$C_i = l_{i,1} \vee l_{i,2} \vee \cdots \vee l_{i,s(i)},$$

where $l_{i,1}, l_{i,2}, \ldots, l_{i,s(i)}$ are literals of the variables $x_1, \ldots, x_n$.

As a first step we assign the expression

$$L_{i,j} = l_{i,j} \wedge (\neg l_{i,1} \wedge \cdots \wedge \neg l_{i,j-1})$$

to the literal $l_{i,j}$ for each i, j, $1 \leq i \leq k$, $1 \leq j \leq s(i)$.

For our previous small textbook like example the described expressions will be the following:

$$L_{1,1} = x_1, L_{1,2} = x_2 \wedge \neg x_1, L_{1,3} = \neg x_3 \wedge \neg x_1 \wedge \neg x_2.$$
$$L_{2,1} = \neg x_1, L_{2,2} = \neg x_2 \wedge x_1, L_{2,3} = x_4 \wedge x_1 \wedge x_2.$$
$$L_{3,1} = x_2, L_{3,2} = \neg x_4 \wedge \neg x_2, L_{3,3} = x_3 \wedge \neg x_2 \wedge x_4.$$

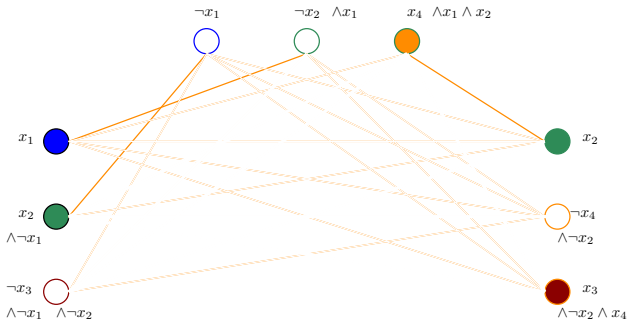
Altogether nine L_{ij} formulas are associated with the SAT instance. We would like to point out that the L_{ij} formulas are not clauses but it will not cause any problem in the course of the clique reformulation of the problem. So the equivalent formula for the original expression is:

$$\Phi = K_1 \wedge K_2 \wedge K_3,$$
$$K_1 = L_{1,1} \vee L_{1,2} \vee L_{1,3},$$
$$K_2 = L_{2,1} \vee L_{2,2} \vee L_{2,3},$$
$$K_3 = L_{3,1} \vee L_{3,2} \vee L_{3,3}.$$

The reader can easily verify that the two formulas are in fact equivalent. We skip the proof, as we do use this fact in our reasoning.

As a second step we define a finite simple graph G. The nodes of the graph G are the conjunctions $L_{i,j}$ for each i, j, $1 \leq i \leq k$, $1 \leq j \leq s(i)$. Two distinct nodes $L_{u,v}$, $L_{p,q}$ of the graph G are not adjacent in G if the formula $L_{u,v} \wedge L_{p,q}$ is a contradiction (always false). See as a small example the new graph for the same textbook problem we described previously in Figure 2.

Figure 2: $\Phi = (x_1 \vee (x_2 \wedge \neg x_1) \vee (\neg x_3 \wedge \neg x_1 \wedge \neg x_2)) \wedge (\neg x_1 \vee (\neg x_2 \wedge x_1) \vee (x_4 \wedge x_1 \wedge x_2)) \wedge (x_2 \vee (\neg x_4 \wedge \neg x_2) \vee (x_3 \wedge \neg x_2 \wedge x_4))$



LEMMA 2.1. *The set of nodes $\{L_{i,1}, \ldots, L_{i,s(i)}\}$ is an independent set in the graph G for each i, $1 \leq i \leq k$. This means that the nodes of the graph G can be well-colored using k colors, the nodes $L_{i,1}, \ldots, L_{i,s(i)}$ all can receive color i.*

PROOF. In order to verify our observation let us consider two distinct nodes $L_{i,p}$, $L_{i,q}$ of the graph G such that $p < q$. The formula $L_{i,p}$ contains the literal $l_{i,p}$ and the formula $L_{i,q}$ contains the literal $\neg l_{i,p}$. Consequently, $L_{i,p} \wedge L_{i,q}$ is a contradiction. Therefore, the nodes $L_{i,p}$, $L_{i,q}$ of the graph G are not adjacent in G.

LEMMA 2.2. *If the clauses $C_1, \ldots, C_k$ can be satisfied simultaneously, then the graph G has a k-clique.*

PROOF. Let us assume that there is an assignment of the truth values of the propositional variables $x_1, \ldots, x_n$ that makes each of the clauses $C_1, \ldots, C_k$ true. In particular for each i, $1 \leq i \leq k$ there is a literal $l_{i,u(i)}$ such that the assignment of the truth values makes $l_{i,u(i)}$ true and makes $l_{i,1}, \ldots, l_{i,u(i)-1}$ false. We simply pick the first among the literals $l_{i,1}, l_{i,2}, \ldots, l_{i,s(i)}$ which is made true by the truth assignment of the variables. In the $u(i) = 1$ particular case the set of indices $\{1, \ldots, u(i) - 1\}$ is empty. Obviously, this assignment of the truth values of the variables makes the conjunction

$$L_{i,u(i)} = l_{i,j} \wedge (\neg l_{i,1} \wedge \cdots \wedge \neg l_{i,u(i)-1})$$

true for each i, $1 \leq i \leq k$. From this follows that $L_{p,u(p)} \wedge L_{q,u(q)}$ is true for each p, q, $1 \leq p < q \leq k$. In other words, the formula $L_{p,u(p)} \wedge L_{q,u(q)}$ cannot be a contradiction. As a consequence the nodes

$$L_{1,u(1)}, L_{2,u(2)}, \ldots, L_{k,u(k)}$$

of the graph G are the nodes of a k-clique in G. □

LEMMA 2.3. *If there is a k-clique in the graph G, then there is an assignment of the truth values of the variables $x_1, \ldots, x_n$ that makes the clauses $C_1, \ldots, C_k$ true simultaneously.*

PROOF. Let us suppose that the graph G contains a k-clique Δ. We know that the nodes of the graph G can be wel-colored using k colors. The nodes of the k-clique Δ must receive k pair-wise distinct colors. It means that the nodes of the k-clique Δ receive the colors $1, 2, \ldots, k$. There are integers $u(1), u(2), \ldots, u(k)$ such that

$$L_{1,u(1)}, L_{2,u(2)}, \ldots, L_{k,u(k)} \tag{1}$$

are the nodes of the k-clique Δ.

Note that the expression $L_{i,u(i)}$ is true when each of the literals

$$l_{i,u(i)}, \neg l_{i,1}, \neg l_{i,2}, \ldots, \neg l_{i,u(i)-1}$$

is true. Using this information we can set the values of the corresponding propositional variables $x_1, \ldots, x_n$ to be true or false. Note, that it is not possible to assign true and false values at the same time to a given variable, since the nodes (1) pair-wise adjacent in the graph G. If a literal in the expression $L_{u,v}$ forces us to set the value of the variable x_i to be true and a literal in the expression $L_{x,y}$ forces us to assign the variable x_i the false value, then the expression $L_{u,v} \wedge L_{x,y}$ contains $x_i \wedge \neg x_i$ and consequently $L_{u,v} \wedge L_{x,y}$ is a contradiction. This violates the fact that the nodes $L_{u,v}$, $L_{x,y}$ of the graph G are adjacent in the k-clique Δ.

Of course, it well may happen that for some variables among $x_1, \ldots, x_n$ we are not forced to assign any of the true or false values. In this case the value of such variable is not restricted and we are free to choose between the truth or false values. In other words a k-clique in the graph G may give rise to more than one truth assignments of the truth values of the variables that satisfy the clauses $C_1, \ldots, C_k$ simultaneously. Namely there will be 2^z of such assignments if the number of unsigned variables was z. □

Remember, that in the graph G two distinct nodes $L_{u,v}$, $L_{x,y}$ are not adjacent if the formula $L_{u,v} \wedge L_{x,y}$ is a contradiction. Let γ be an assignment of truth values to the propositional variables $x_1, \ldots, x_n$. Using γ we color certain edges of the graph G. The edge connecting the nodes $L_{u,v}$, $L_{x,y}$ receives red color if the truth value assignment γ makes the expression $L_{u,v} \wedge L_{x,y}$ true. If each edge of a k-clique Δ in the graph G is red, then we call the k-clique Δ a red k-clique in the graph G. We call the assignment γ a satisfying assignment if γ makes each of the clauses $C_1, \ldots, C_k$ true simultaneously. Lemma 2.2 can can be restated in the following way.

LEMMA 2.4. *If γ is a satisfying assignment of the truth values of the variables $x_1, \ldots, x_n$, then there is exactly one red k-clique in the graph G.*

PROOF. Lemma 2.2 proved that there is at least one such clique, as it is constructed one.

Assume on the contrary that there are two non-identical red k-cliques Δ_1, Δ_2 in the graph G. Since the red k-cliques Δ_1, Δ_2 are not identical, there is a color class, say the i-th color class, such that the nodes $L_{i,p}$, $L_{i,q}$ of Δ_1, Δ_2 in the i-th color class are not identical.

As $L_{i,p}$ is an end point of a red edge in the red k-clique Δ_1, the assignment γ makes the expression $L_{i,p}$

true. Similarly, γ makes the expression $L_{i,q}$ true. But from Lemma 2.1 we know that $L_{i,p} \wedge L_{i,q}$ is a contradiction, so $L_{i,p}$, $L_{i,q}$ cannot be true at the same time, so Δ_1, Δ_2 must be identical. □

THEOREM 2.5. *The proposed reformulation along with the k-clique enumerating method, where for each clique one adds up numbers 2^z, where z is the number of unset variables, solves the #SAT problem.*

PROOF. Concludes from Lemmas 2.2,2.3 and 2.4.

3 Discussion

One can notice, that if the ordering of the literals are fixed in the same way in each clause then the proposed method is very similar to the DPLL method adjusted for #SAT in [1]. The dedicated reader will realize that in the proposed method rearranging the literals among each other in a clause will lead to essentially different graphs in the clique reformulation. These graphs not only have different number of edges, but the number of k-cliques can also differ. At the first glance this may look as a disadvantage. However, there is another way to look at this phenomenon. The sizes of the search spaces vary with permuting the literals in the original SAT instance. Thus it opens up an avenue reducing the size of the search tree by tactically choosing among the possible rearrangements of the literals. It gives a certain flexibility, as one can use different methods (nowadays for example artificial intelligence) to find out which ordering and thus reformulation would give the best approach in terms of solution speed.

Also, one can apply preconditioning methods for the resulting graph based on graph theoretical considerations, like proposed in [5]. It may be much easier to enumerate all k-cliques in the resulting graph, this is a future goal of our research.

We also did a tiny computational experiment. We took two simple examples from [4], namely from Section 4, with parameters $t = 2, 3$, where the SAT formula has $8t + 22$ clauses and each of these clauses has length three, and the formula is unsatisfiable, that is the number of solution is zero. So our first 3-SAT example has 38 clauses, and the second example has 46.

The textbook reformulation of the first problem gave a graph with 114 nodes and 6 218 edges. The clique search algorithm we used lead to a search tree of size **15 414 191**. After an edge dominance preconditioning from [5] we got a simplified graph of 5 967 edges, and the clique search lead to a search tree of size **28 474**. The proposed reformulation of the first problem gave a graph with 114 nodes and 5 953 edges. The clique search algorithm we used lead to a search tree of size **92 296**. After an edge dominance preconditioning the simplified graph had 5 619 edges, and the clique search lead to a search tree of size **18 202**.

The textbook reformulation of the second problem gave a graph with 138 nodes and 9 174 edges. The clique search algorithm lead to a search tree of size **2 497 099 319**. After an edge dominance preconditioning we got a simplified graph of 8 730 edges, and the clique search lead to a search tree of size **142 399**. The proposed reformulation of the second problem gave a graph with 138 nodes and 8 805 edges. The clique search algorithm we used lead to a search tree of size **922 996**. After an edge dominance preconditioning the simplified graph had 8 233 edges, and the clique search lead to a search tree of size **72 810**.

It is clear from these examples, that both preconditioning and the proposed reformulation may greatly aid the calculation in some cases.

We close this section with some clarifying notes. The preconditioning rules used in connection with k-clique problems can be sorted into two broader classes. Namely, k-clique preserving and k-clique loosing rules. The k-clique preserving rules reduces the given graph G to a new graph G' in such a way that each k-clique in G is also a k-clique in the reduced graph G'. On the other hand, the k-clique loosing rules reduces the given graph G to a new graph G'' such that some k-clique in G may not be a k-clique in the reduced graph G'' but at least one of the k-cliques of the original graph G still will be a k-clique in the reduced graph G''. Therefore, when our purpose is to enumerate all the possible k-cliques of G, then we cannot simply apply k-clique loosing preconditioning rules. However, when our purpose is to verify that the given graph G does not have any k-clique, then we may apply k-clique loosing preconditioning rules.

The edge doninance preconditioning rule is a k-clique loosing preconditiong rule. In the same time it was known that the examples from [4] are unsatisfiable. Therefore using k-clique loosing rules are fully justified in this case.

On the other hand, surprisingly, even when there are several k-cliques present and the goal is to count them, one can use clique loosing preconditioning like dominance in the following way. In such a case, one need to keep a list of these transformations, and during the enumeration each solution needs to be check against that list. This is justified by the fact that such transformations always delete a node (edge) while pointing to another one as a possible substitution. That is, if node u dominates node v and we deleted node v, then during the solution listing process we need to check each solution if it contains (the remaining) node u, and if substituting back node v instead of node u also gives a valid solution.

This approach however is not developed yet, and certainly needs a more elaborate program for listing solutions.

References

[1] Elazar Birnbaum and Eliezer L. Lozinskii. The good old davis-putnam procedure helps counting models. *Journal of Artificial Intelligence Research*, 10(1):457–477, 1999.

[2] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, third edition, 2009.

[3] Richard M. Karp. Reducibility among combinatorial problems. In R. E. Miller, J. W. Thatcher, and J.D. Bohlinger, editors, *Complexity of Computer Computations*, pages 85–103. New York: Plenum, 1972.

[4] Ming Ouyang. How good are branching rules in dpll? *Discrete Applied Mathematics*, 89(1):281–286, 1998.

[5] Sándor Szabó and Bogdán Zaválnij. Clique search in graphs of special class and job shop scheduling. *Mathematics*, 10(5), 2022.

Modularity aware graph clustering for exploratory tasks with a case study of the biomass supply chain

Sylvert Prian Tahalea
sylvert@inf.u-szeged.hu
University of Szeged, Hungary
Universitas Pembangunan Nasional Veteran Yogyakarta, Indonesia

Arkadiusz Kawa
Poznań School of Logistics, Department of Logistics, Poznań
Poznań University of Economics and Business, Department of Business Relationships and International Marketing, Poznań
Poland

Balázs Dávid
balazs.david@innorenew.eu
InnoRenew CoE, UP IAM and UP FAMNIT, University of Primorska, Slovenia

Abstract

This paper proposes a novel modularity-aware graph clustering algorithm that combines label propagation principle and global modularity. The algorithm consists of two phases: (1) the first phase is to quickly form clusters using LPA and approximate modularity to evaluate their quality, and (2) the second phase focuses on refining the cluster structures using global modulation. The proposed algorithm is later evaluated using several metrics with several datasets and applied to a real-world use case. The results reveal a clear trade-off between internal and external cluster quality, which is useful for exploratory tasks. The modularity-aware graph clustering algorithm performed well in the experimental and the real-world cases, producing reliable clusters for each case.

Keywords

graph clustering, modularity, LPA, exploratory data analysis

1 Introduction

Graph clustering has arisen as a critical issue in network research, data mining, and machine learning due to its potential to reveal hidden structures within complex systems. Real-world phenomena such as social interactions, biological processes, communication networks, citation graphs, and transportation systems can be naturally modeled as graphs, with nodes representing things and edges representing their relationships. Identifying cohesive groups of nodes, also known as clusters or communities, allows researchers to get a better understanding of the modular organisation of networks and simplify large-scale data for analysis.

Over the past decades, various algorithmic paradigms have arisen, ranging from modularity optimisation, spectral approach, and random walks to label propagation, statistical inference, and most recently, utilising graph neural networks. Each technique strikes a balance between accuracy, scalability, interpretability, and applicability to overlapping or hierarchical communities. Despite these achievements, there are some problems remaining, such as the low modularity for the fast algorithms, efficient methods for large-scale graphs, adaptive strategies for temporal graphs, and the lack of universally acknowledged ground truth.

The Label propagation algorithm (LPA) is considered to be one of the fastest graph clustering techniques. However, it has low modularity [20, 17]. Modularity measurement objectively measures the quality of the communities, where the value near one indicates well-structured communities, and a value near zero indicates a bad community structure. There are several LPA extensions to increase the modularity utilizing its fast runtime, namely LPAm [1], LPAm+ [13], LPA-MNI [12], and FLPA [18].

This paper proposes a novel modularity-aware graph clustering algorithm utilising LPA, local modularity, and global modularity. This algorithm consists of two phases: fast label propagation and global modularity refinement. In the first phase, utilising LPA for the fast propagation and local modularity to regain its initial structure. The second phase focuses on increasing the modularity using global modularity refinement. Finally, the result will provide higher-quality partitions compared to the original LPA. The algorithm is later evaluated using different metrics with several datasets. Moreover, in order to invetsigate the quality of the methodology in real-world scenario, a use case of biomass supply chains is studied.

The remainder of the paper is organised as follows. **Section 2** provides the methodology and evaluation techniques. **Section 3** provides the main results and highlights the main findings. **Section 4** provides the use case on a real-world scenario. **Section 5** concludes the contribution, limitations, and future research.

2 Methodology

2.1 Modularity

Modularity is one of the most influential principles in graph clustering, which is evaluated by the strength of a network's partition by comparing the density of edges inside communities [15]. High modularity value indicates that clusters have significantly more internal connections than external ones. Modularity optimisation has become a benchmark for measuring cluster quality, but the direct optimisation of modularity is challenging due to its NP-hard nature. The modularity measurement [15] is as follows.

$$Q = \frac{1}{2m} \sum [A_{ij} - \frac{k_i k_j}{2m}] \delta(c_i, c_j), \tag{1}$$

where A_{ij} represents the actual connection between nodes i and j, the term $\frac{k_i k_j}{2m}$ is the expected number of edges between nodes i and j, and c_i is the community assignment.

To avoid recomputing modularity for every possible partition, Blondel et al. [2] proposed a local measurement that evaluates the modularity change if a node were moved from its original cluster to the neighboring cluster and also provides the approximate modularity. The approximate modularity is as follows.

$$\Delta Q(v, l) \approx \frac{k_{v,in}}{m} - \frac{k_v \cdot \text{sum\_tot}(l)}{2m^2} \tag{2}$$

Information Society 2024, 7–11 October 2024, Ljubljana, Slovenia

Algorithm 1 Fast propagation phase

Require: Graph $G = (V, E)$
Ensure: Node labels $L(v)$ for all $v \in V$
Assign each node $v \in V$ a unique label $L(v) \leftarrow v$
repeat
  for all nodes $v \in V$ in random order **do**
    Compute label frequencies among neighbors of v
    Select top-k frequent labels (optional)
    for all candidate labels l **do**
      Estimate local modularity gain $\Delta Q(v, l)$
    end for
    Select label l^* with maximum gain (if gain > threshold)
    Update $L(v) \leftarrow l^*$
  end for
until no label changes or maximum iterations reached

where $k_{v,in}$ is the number of edges from node v carrying label l; k_v is the degree of node v; sum_tot(l) is the total degree of nodes with label (l); and m is the number of edges in the network.

2.2 Label Propagation Algorithm

The label propagation algorithm (LPA) was introduced as one of the simplest yet most scalable methods for graph clustering [16]. The algorithm begins by assigning a unique label to each node, then iteratively updates the node's label based on the most frequent label among its neighbors. The process is performed in random sequential order to avoid bias, and a label is chosen at random when there are multiple labels that have the same frequency. The iterations continue until every node holds a label that is the majority among the neighbors, leading to convergence in the network. The clusters are formed by grouping the nodes that share the same label. LPA has near-linear time complexity, ability to uncover clusters without prior knowledge of their number, and is scalable for large networks.

2.3 Modularity-Aware Graph Clustering

The modularity-aware graph clustering utilizes LPA's ability to quickly cluster the nodes and modularity refinement to produce better cluster structures. This proposed method is divided into two phases: (1) The first phase is to quickly form clusters using LPA and approximate modularity to evaluate their quality, and (2) the second phase focuses on refining the cluster structures using global modularity measurement.

2.3.1 Phase 1: Fast Propagation Phase. The objective of the first phase is to obtain the local approximation of the modularity gain by rapidly propagating through the network. This step helps to form a coarse graph for further refinement. The initial label for each node is its own node ID; then the algorithm iterates over all the nodes in a randomized order and updates the labels using the local approximation of modularity gain as in Equation (2). This step also leverages the Top-K label filtering to reduce the chance of unstable changes while a minimum gain threshold is applied to prevent weak label changing. The iteration process will continue until there is no label change or the maximum number of iterations is reached. The fast propagation algorithm is presented in Algorithm 1.

2.3.2 Phase 2: Modularity Refinement. The objective of the second phase is to refine the approximation performed in the first phase. Whenever the label is spread across the network, maximal modularity is gained from the overall structure. Therefore, the second phase performs the refinement for each label using global modularity, as in Equation (1), for the entire network. The modularity refinement is presented in Algorithm 2.

Algorithm 2 Global Modularity Refinement

Input: Label map L, Graph G, stability threshold R
Initialize stability counter $S(v) \leftarrow 0$ for all
repeat
  for all nodes $v \in V$ not marked stable **do**
    Compute label frequencies among neighbors
    Select Top-k candidate labels (optional)
    for all labels l **do**
      Temporarily assign $L'(v) \leftarrow l$
      Compute new global modularity Q'
    end for
    Select label l^* that maximizes Q'
    if $Q' > Q$ **then**
      Update $L(v) \leftarrow l^*$, $S(v) \leftarrow 0$
    else
      $S(v) \leftarrow S(v) + 1$
      if $S(v) \geq R$ **then**
        Mark v as stable
      end if
    end if
  end for
until no label changes or improvement $< \epsilon$

2.4 Evaluation

The proposed algorithm is evaluated using several common metrics that are widely accepted for graph clustering, such as the modularity measurement, normalized mutual information (NMI), Adjusted Rand Index (ARI), and F1-score. The modularity measurement measures the internal structures of the cluster using Equation (1), while NMI measures the similarity between the result cluster and the ground truth of the dataset as follows [4].

$$NMI(A, B) = \frac{2 \cdot I(A, B)}{H(A) + H(B)} \tag{3}$$

where $I(A, B)$ is the mutual information between clusters A and B; and $H(A)$ and $H(B)$ are the entropies of the clusters.
The Adjusted Rand Index (ARI) is used to measure similarity between generated clusters and ground truth [6], while F1-score evaluates the precision and recall for each cluster [14, 3]. The ARI is measured using Equation (4) and F1-score is measured using Equation (5).

$$\text{ARI} = \frac{\sum_{ij} \binom{n_{ij}}{2} - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}\right] / \binom{n}{2}}{\frac{1}{2}\left[\sum_i \binom{a_i}{2} + \sum_j \binom{b_j}{2}\right] - \left[\sum_i \binom{a_i}{2} \sum_j \binom{b_j}{2}\right] / \binom{n}{2}} \tag{4}$$

where n is the number of nodes; $n_{i,j}$ is the number of nodes in both the predicted cluster i and the ground truth cluster j; a_i is the number of nodes in the predicted cluster i; b_j is the number of nodes in the ground truth cluster j; and $\binom{x}{2}$ is a binomial coefficient to count the number of pairs between clusters.

$$F_1 = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} = \frac{2 \cdot TP}{2 \cdot TP + FP + FN} \tag{5}$$

where TP are correctly predicted positive cases; FP are incorrectly predicted as positive; and FN are missed positive cases, which are predicted as negative.

3 Experiments and results

.

3.1 Experimental Datasets and results

There are four real-world datasets used in this research, such as Zachary's Karate Club Network [5], Football [5], Polbooks [8], and Email [19, 10, 11] from SNAP datasets. The synthetic datasets are also used in this research, utilizing the LFR framework [9].

The summary of the datasets is presented in Table 1, where c is the number of clusters, k is the average degree, μ is the mixing parameter. The measurement is compared with several algorithms such as LPA, LPAm+, and Louvain. This is meant to compare the proposed algorithm with its predecessors and one of the top graph clustering algorithms.

Table 1: Comparative metadata of benchmark datasets

Dataset	Nodes	Edges	c	k	μ
Karate	34	78	2	-	-
Football	115	613	3	-	-
Polbooks	105	441	12	-	-
Email	1005	25571	42	-	-
LFR1	1000	10455	44	20.91	0.10
LFR2	2000	15495	32	15.49	0.15
LFR3	5000	93743	56	37.50	0.30
LFR4	10000	113290	61	22.66	0.10
LFR5	10000	120047	75	24.01	0.20
LFR6	10000	190440	84	38.09	0.25

3.1.1 Modularity and NMI. The modularity evaluation is meant to evaluate the structure of the clusters. The larger value, closer to 1, indicates a densely connected structure within the results, but it does not indicate the accuracy of the community detection. The results presented in Table 2 show that the modularity aware label propagation algorithm has the best performance across the datasets, from the small one to large datasets. The modularity aware label propagation performed at the same level as Louvain in the LFR4-LFR6 with the highest modularity score, showing that it can produce high-quality clusters.
The normalized mutual information measures the similarity between clusters generated by the algorithms and the ground truth. Based on the results presented in Table 2, the LPA is the best performer, reaching the highest score for almost all the datasets. The modularity aware label propagation algorithm is the second best for NMI evaluation, with notably one highest score for the Email network dataset, having a huge gap with LPA, which is the best overall NMI performer.

3.1.2 ARI and F1-score. The Adjusted Rand Index (ARI) quantifies the similarity between the generated clusters and the ground truth, considering all pairs of samples and evaluating whether they are assigned to the same cluster. The results presented in Table 3 show that only LPA is the best performer. The modularity aware label propagation only has the highest score once, for the Email network case, while becoming the second best for most other cases.
The F1-score serves as an evaluation metric for assessing the quality of cluster assignments. The results presented in Table 3 show that LPA produced high-quality clusters, reaching the highest score for most of the real-world and synthetic datasets. The modularity-aware label propagation algorithm performs as the second best and joins the LPA as the best performer for the three largest datasets in this research.

3.2 Discussion

The results highlight a clear distinction between internal and external clustering quality. The modularity-aware label propagation consistently maximizes the modularity, as presented in Table 2, indicating densely connected clusters in the graph topology. However, the standard LPA attains the strongest argument with ground truth across most datasets, suggesting that it works better

Table 2: Comparison of algorithms in terms of modularity and NMI across datasets

Dataset	Modularity			
	LPA	LPAm+	MA-LPA	Louvain
Karate	0.309	0.418	**0.445**	0.427
Polbook	0.481	0.493	**0.528**	0.519
Football	0.583	0.557	**0.604**	0.596
Email	0.089	0.418	**0.432**	0.431
LFR1	0.807	0.809	**0.810**	0.808
LFR2	0.479	0.514	**0.515**	0.511
LFR3	0.569	0.735	**0.742**	0.731
LFR4	0.829	0.824	**0.830**	**0.830**
LFR5	0.839	0.840	**0.841**	**0.841**
LFR6	0.611	0.609	**0.613**	**0.613**
	NMI			
	LPA	LPAm+	MA-LPA	Louvain
Karate	**1.000**	0.565	0.607	0.483
Polbooks	**1.000**	0.611	0.759	0.697
Football	**1.000**	0.919	0.937	0.939
Email	0.180	0.593	**0.663**	0.575
LFR1	0.932	**1.000**	0.964	0.993
LFR2	**1.000**	0.831	0.854	0.823
LFR3	0.301	0.602	**0.615**	0.598
LFR4	**1.000**	0.986	0.989	0.964
LFR5	**1.000**	0.979	0.994	0.995
LFR6	**1.000**	0.961	0.975	0.963

Table 3: Comparison of algorithms in terms of ARI and F1-score across datasets

Dataset	ARI			
	LPA	LPAm+	MA-LPA	Louvain
Karate	**1.000**	0.398	0.597	0.483
Polbooks	**1.000**	0.509	0.710	0.579
Football	**1.000**	0.833	0.842	0.892
Email	0.011	0.332	**0.440**	0.315
LFR1	0.778	**1.000**	0.905	0.976
LFR2	**1.000**	0.400	0.411	0.387
LFR3	0.177	0.471	**0.487**	0.462
LFR4	**1.000**	0.911	0.966	0.959
LFR5	**1.000**	0.955	0.980	0.979
LFR6	**1.000**	0.918	0.899	0.903
	F1-score			
	LPA	LPAm+	MA-LPA	Louvain
Karate	**1.000**	0.807	0.956	0.850
Polbooks	**1.000**	0.688	0.897	0.665
Football	**1.000**	0.869	0.986	0.745
Email	0.078	0.128	**0.261**	0.130
LFR1	0.546	**1.000**	**1.000**	0.710
LFR2	**1.000**	0.907	0.998	0.855
LFR3	0.185	0.491	**0.512**	0.498
LFR4	**1.000**	0.661	**1.000**	0.877
LFR5	**1.000**	0.759	**1.000**	0.0.993
LFR6	**1.000**	0.796	**1.000**	0.837

for datasets with known labels. The ARI and F1-score results in Table 3 show the consistent trade-offs of modularity-aware label propagation, which is optimizing the global structural objective not always gives the best intended partition against known labels.

This means that the algorithm choice should reflect the evaluation goal. When the task is exploratory and the ground truth is not available, maximizing internal structure is attractive for discovering cohesive clusters; thus, modularity-aware label propagation algorithms are better. When the aim is to recover known categories or enable downstream tasks, LPA is safer given its superior NMI, ARI, and F1-score.

4 Case study

4.0.1 Dataset. This case study draws on a quantitative survey conducted in late 2022 within the BioLog project, aimed at developing a model of the reverse supply chain of residual wood biomass. The survey targeted enterprises across the wood sector—sawmills, furniture and joinery producers, recyclers, and logistics providers—using a stratified random sampling method based on PKD (The Polish Classification of Activities) industry codes and a project-specific list. Data were collected through 300 Computer Assisted Telephone Interviews (CATI) with company representatives knowledgeable about wood residue origins and management. The questionnaire gathered company profile data (value chain position, employment size, revenue) and explored types of wood residues used (forestry/agricultural, post-production, post-consumer) as well as their applications in energy generation, energy carriers, and wood-based materials. It also investigated logistics and technological processes, cooperation networks, requirements, and constraints in biomass management, and adaptations in transport, storage, and unitization practices. The dataset was anonymised, and respondents were assured of confidentiality, informed consent, and voluntary participation.

4.0.2 Results. The dataset underwent a series of preprocessing steps before being analyzed. The answers to the questionnaire were transformed into categorical and numerical data. The Interquartile Range (IQR) performed to detect the outlier data and resulted in reducing the number of questions to only include relevant ones. The clean data was used to model a graph with the nodes representing the firms and the edge weight between any two companies using position-aware Jaccard similarity [7]: essentially measuring the fraction of survey questions on which the two firms gave identical answers. There were several versions of graph models such as unweighted graph, plain weighted graph, and weighted graph with thresholds (0.1, 0.125, and 0.25)

The graph indicates that most businesses are connected through shared practices. Out of 300 firms, 272 (about 91%) form one large connected group, while the other 28 are isolated or in very small clusters, representing outliers. This core network isn't very dense, but it does have clusters: companies that work with the same partner are commonly connected to each other, creating tiny groupings with shared qualities. The average distance between companies is modest (approximately 3 steps), and the longest distance is tiny (6–7 steps), which gives the network a "small-world" shape. In practice, this means that any two firms, even from different parts of the sector, can be connected through only a few intermediaries.

The proposed algorithm applied to this data, alongside with Louvain, LPA, and LPAm+ to identify the clusters. The graph clustering methods uncovered a non-trivial clustering structure in the data, partitioning the firms into clusters without any apriori categorization. However, the results demonstrate that explicitly optimizing for modularity yields a superior segregation of the network.

Although no explicit categories (such as firm type or size) were given to the clustering algorithms, the communities they detected appear to reflect meaningful behavioral and operational patterns among the companies. In other words, firms ended up clustered together because they answered many survey questions in similar ways – a purely data-driven outcome that likely corresponds to real-world commonalities. For example, one cluster derived from the graph predominantly consists of sawmills and wood processors that generate large volumes of residues and use them for bioenergy, while another cluster groups manufacturers (e.g. furniture or flooring producers) that have different residue uses and logistics practices. Indeed, the analysis suggests that companies naturally form a few distinct sub-communities: even without predefining any segments, those with analogous supply chain roles, residue utilization strategies, or challenges tend to congregate in the same community. This insight is valuable for policymakers and industry stakeholders – it implies the sector can be segmented into groups with shared characteristics, which may each benefit from tailored strategies (for instance, a cluster of firms focused on energy production might face similar regulatory and technological issues).

In constructing the similarity measure, all survey features were treated with equal weight. This methodological choice (i.e. not assigning higher importance to any particular question or topic) means the clustering was driven by overall similarity across many attributes. A side effect is that very common attributes (such as broadly adopted practices) contribute to linking many firms, potentially overshadowing rarer but distinctive features. In the present case, however, even unweighted features yielded intelligible clusters, indicating that the dominant patterns in the data were strong enough to shape the communities. Future work could experiment with feature weighting (for instance, giving more emphasis to specific key questions or rare responses) to see if even clearer or more nuanced groupings emerge. Nonetheless, the current graph clustering results already highlight consistent patterns: companies with similar operational profiles gravitated into the same clusters. In summary, the modularity-aware graph clustering of the BioLog survey data uncovers a modular structure in the wood biomass supply chain, revealing that despite the lack of explicit grouping criteria, firms naturally aggregate into network communities that mirror their shared behaviors and challenges. The performance of the algorithms is presented in Table 4.

5 Conclusion

In conclusion, the results reveal a clear trade-off between internal and external cluster quality; the modularity-aware label propagation consistently attains the highest modularity, which is useful for exploratory tasks. The standard LPA achieved the best agreement with ground truth on NMI, ARI, and F1-score across most real and synthetic datasets, showing it's better for optimizing downstream tasks. Overall, these findings argue against a single best method and support choosing the algorithm according to the evaluation objective and data regime. Moreover, a real-world case study of biomass supply chains demonstrates that our methodology provides results for strategic decision making.

Table 4: Algorithms performance in the case study

	Unweighted Graph			
	LPA	LPAm+	MA-LPA	Louvain
No. of Cluster(s)	1	1	**3**	1
Modularity	0	0	**0.259**	0
	Weighted Graph			
	LPA	LPAm+	MA-LPA	Louvain
No. of Cluster(s)	1	1	**8**	6
Modularity	0	0	0.145	**0.147**
	Weighted Graph (0.1)			
	LPA	LPAm+	MA-LPA	Louvain
No. of Cluster(s)	1	1	**10**	8
Modularity	0	0	**0.155**	0.139
	Weighted Graph (0.125)			
	LPA	LPAm+	MA-LPA	Louvain
No. of Cluster(s)	1	1	**10**	**10**
Modularity	1	1	**0.153**	0.146
	Weighted Graph (0.25)			
	LPA	LPAm+	MA-LPA	Louvain
No. of Cluster(s)	1	1	4	**10**
Modularity	0	0	**0.266**	0.238

Finally, there are several methodological considerations that need to be improved. Future work could explore several improvements, such as (1) multi-objective formulation that balances modularity with information-theoretic alignment; (2) considering variability over multiple runs and using consensus clustering; and (3) examining performance under different mixing parameters and larger datasets.

Acknowledgements

The research was supported by the BioLOG project: the authors are grateful for the support of the National Center of Science (NCN) through grant DEC-2020/39/I/HS4/03533, the Slovenian Research and Innovation Agency (ARIS) through grant N1-0223, and the Austrian Science Fund (FWF) through grant I 5443-N. Balázs Dávid is also grateful for the support of the Slovenian Research and Innovation Agency (ARIS) through grant J1-50000, and gratefully acknowledges the Slovenian Research and Innovation Agency (ARIS) and the Ministry of the Economy, Tourism and Sport (MGTŠ) for the grant V4-2512.

References

[1] Michael J Barber and John W Clark. 2009. Detecting network communities by propagating labels under constraints. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, 80, 2, 026129. DOI: 10.1103/PhysRevE.80.026129.

[2] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008, 10, P10008. DOI: 10.1088/1742-5468/2008/10/P10008.

[3] Peter Christen, David J Hand, and Nishadi Kirielle. 2023. A review of the f-measure: its history, properties, criticism, and alternatives. *ACM Computing Surveys*, 56, 3, 1–24.

[4] Leon Danon, Albert Diaz-Guilera, Jordi Duch, and Alex Arenas. 2005. Comparing community structure identification. *Journal of statistical mechanics: Theory and experiment*, 2005, 09, P09008. DOI: 10.1088/1742-5468/2005/09/P09008.

[5] Linton C Freeman. 1977. A set of measures of centrality based on betweenness. *Sociometry*, 35–41. DOI: 10.2307/3033543.

[6] Lawrence Hubert and Phipps Arabie. 1985. Comparing partitions. *Journal of classification*, 2, 1, 193–218. DOI: 10.1007/BF01908075.

[7] Paul Jaccard. 1901. Étude comparative de la distribution florale dans une portion des alpes et des jura. *Bull Soc Vaudoise Sci Nat*, 37, 547–579.

[8] Valdis Krebs. 2004. Books about us politics. *unpublished, http://www. orgnet. com*.

[9] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. 2008. Benchmark graphs for testing community detection algorithms. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, 78, 4, 046110. DOI: 10.1103/PhysRevE.78.046110.

[10] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. 2007. Graph evolution: densification and shrinking diameters. *ACM transactions on Knowledge Discovery from Data (TKDD)*, 1, 1, 2–es. DOI: 10.1145/1217299.1217301.

[11] Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data. (2014).

[12] Huan Li, Ruisheng Zhang, Zhili Zhao, and Xin Liu. 2021. Lpa-mni: an improved label propagation algorithm based on modularity and node importance for community detection. *Entropy*, 23, 5, 497. DOI: 10.3390/e23050497.

[13] Xin Liu and Tsuyoshi Murata. 2010. Advanced modularity-specialized label propagation algorithm for detecting communities in networks. *Physica A: Statistical Mechanics and its Applications*, 389, 7, 1493–1500. DOI: 10.1016/j.physa.2009.12.019.

[14] Christopher D Manning. 2008. *Introduction to information retrieval*. Syngress Publishing.

[15] Mark EJ Newman and Michelle Girvan. 2004. Finding and evaluating community structure in networks. *Physical review E*, 69, 2, 026113. DOI: 10.1103/PhysRevE.69.026113.

[16] Usha Nandini Raghavan, Réka Albert, and Soundar Kumara. 2007. Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, 76, 3, 036106. DOI: 10.1103/PhysRevE.76.036106.

[17] Seema Rani and Monica Mehrotra. 2017. Hybrid influential centrality based label propagation algorithm for community detection. In *2017 International Conference on Computing, Communication and Automation (ICCCA)*. IEEE, 11–16. DOI: 10.1109/CCAA.2017.8229801.

[18] Vincent A Traag and Lovro Šubelj. 2023. Large network community detection by fast label propagation. *Scientific Reports*, 13, 1, 2701. DOI: 10.1038/s41598-023-29610-z.

[19] Hao Yin, Austin R Benson, Jure Leskovec, and David F Gleich. 2017. Local higher-order graph clustering. In *Proceedings of the 23rd ACM SIGKDD international conference on knowledge discovery and data mining*, 555–564. DOI: 10.1145/3097983.3098069.

[20] Aiping Zhang, Guang Ren, Yejin Lin, Baozhu Jia, Hui Cao, Jundong Zhang, and Shubin Zhang. 2014. Detecting community structures in networks by label propagation with prediction of percolation transition. *The Scientific World Journal*, 2014, 1, 148686. DOI: 10.1155/2014/148686.

A Synthetic Multi-View Tracking and 3D Pose Dataset for Automated Airport Visual Surveillance

Ahmed Mansour
Csaba Beleznai
Fabio F. Oberweger
Verena Widhalm
firstname.lastname@ait.ac.at
Assistive & Autonomous Systems, AIT Austrian Institute of Technology GmbH
Vienna, Austria

Nadezda Kirillova
Horst Possegger
Institute of Visual Computing
Graz University of Technology
Graz, Austria
firstname.lastname@tugraz.at

ABSTRACT

Multiple object tracking (MOT) is a core task of automated visual surveillance systems. Driven by Deep Learning, recent advances have significantly improved tracking accuracy and robustness across varied scenarios. However, wide-area visual surveillance, such as observing airport ground and airborne objects, still remains challenging. In such scenarios the presence of small objects, numerous visually similar targets, frequent occlusions, atmospheric effects, scale diversity, and substantial scene clutter continue to degrade detection and tracking performance. To address these limitations, we present the *AirTrackSim25* synthetic airport surveillance dataset, designed to support and advance research in object detection, tracking, and data association. The dataset includes multiple wide-area perspectives of an airport environment, featuring rich structural details and realistic simulations of ground and air-based object motion. Each aircraft is annotated with comprehensive 2D and 3D bounding box and key-point information, as well as motion trajectories. We demonstrate the applicability of the dataset for MOT tasks by benchmarking a baseline neural learning method. The dataset is publicly available at *https://github.com/cbelez/AirTrackSim25*.

CCS CONCEPTS

• **Computing methodologies** → **Computer vision**; • **Information systems** → **Sensor networks**.

KEYWORDS

synthetic data generation, visual surveillance, multi-target tracking

ACM Reference Format:
Ahmed Mansour, Csaba Beleznai, Fabio F. Oberweger, Verena Widhalm, Nadezda Kirillova, and Horst Possegger. 2025. A Synthetic Multi-View Tracking and 3D Pose Dataset for Automated Airport Visual Surveillance. In *Proceedings of Middle-European Conference on Applied Theoretical Computer Science (MATCOS-25).* ACM, New York, NY, USA, 4 pages. https://doi.org/XXXXXXX.XXXXXXX

MATCOS-25, Oct. 09–10, 2025, Koper, Slovenia

ACM ISBN 978-1-4503-XXXX-X/2018/06
https://doi.org/XXXXXXX.XXXXXXX

Figure 1: Visual summary of the main dataset characteristics, with camera views indicated by red triangles.

1 INTRODUCTION

Automated wide-area visual surveillance systems have become increasingly important for public and private infrastructures, as they enable continuous monitoring of large geographic regions through neural learning and computer vision methods. By integrating real-time target detection and tracking with scene interpretation, these systems facilitate rapid incident response, thus enhancing safety, security, and operational efficiency. Furthermore, their scalability with respect to the number of camera views makes them particularly well suited for managing complex environments, such as airports, shopping centers, and transportation hubs.

In this work, we address the automated analysis of airport scenes, focusing on the multi-object tracking (MOT) task. This problem setting is confronted with data scarcity and presents several unique research challenges. Although extensive and diverse datasets are available for common object categories such as *pedestrians* and *cars* [1],[3], publicly accessible airport-specific datasets with trajectory annotations remain limited to the AGVS-T22 dataset [6] and

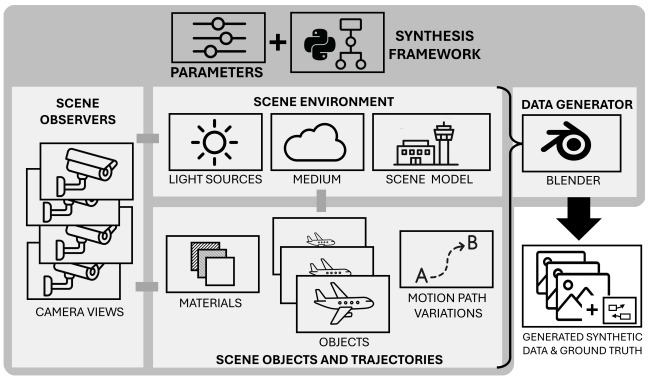


Figure 2: Systemic diagram of the Blender-based synthetic airport data generator.

Figure 3: Top-view of the scene showing specifically positioned path points (orange) and parking points (blue). These points are used to establish randomized directional connections between neighboring nodes, to create realistic airplane movements. A sample generated trajectory is shown in green.

the TartanAviation dataset [7]. Furthermore, recent neural learning frameworks, such as FairMOT [11] and CenterTrack [10, 12], formulate detection and tracking within a multitask learning paradigm, which requires annotation-intensive track-labeled datasets. Lastly, constructing well-balanced training corpora that capture diverse and challenging scenarios — such as interacting aircrafts at diverse spatial scales, motion dynamics, and degraded visibility conditions — is difficult to accomplish manually. In light of these challenges, related work has focused on common object categories, such as in the series of MOTChallenge [1] and the KITTI benchmark [3]. Small object detection and tracking also received attention recently, such as in the SMOT challenge [5], where the limited amount of appearance cues due to the small target size renders the task difficult.

In this work, we present the *AirTrackSim25* synthetic airport dataset (see Figure 1), which comprises 14 distinct camera viewpoints and a total of 23,278 image frames with 131,866 annotated aircraft instances, capturing realistic airport operational scenarios. The dataset is designed to advance research in small object detection and tracking by providing spatially precise 2D bounding box (BB) and part-level annotations, occlusion status, 3D BB information, and consistent target identities with associated trajectories. To assess the utility of the dataset, we present baseline synthetic neural learning experiments for a 2D MOT task, followed by validation on the real-world AGVS-T22 dataset [6].

The paper is structured as follows: Section 2 provides a brief description of the synthetic generation scheme. Section 3 describes the baseline experiments and their evaluation. Finally, Section 4 concludes the paper.

2 DATA GENERATION SCHEME

To construct our synthetic airport dataset, we employ Blender [2] as the primary modeling and rendering environment. Within Blender, we integrate a commercially available, metrically accurate, and highly detailed 3D model of the London Heathrow Airport [4], which serves as the foundation for creating a populated and dynamic scene. The entire simulation is based on Blender's python backend. Our developed framework enables the controlled simulation of realistic aircraft operations, including variations in spatial scale, motion patterns, and interaction scenarios. By leveraging the flexibility of the synthetic setup, we are able to generate diverse viewpoints, temporal sequences, and annotation-rich data that would be prohibitively costly and logistically challenging to obtain through real-world data collection. In the following, we describe the individual aspects of the data generation process, as shown in Figure 2:

Parameters and Domain Randomization: Most scene parameters—including lighting conditions, target kinematics, and camera geometries—are randomized within predefined minimum and maximum bounds using uniform distributions. This randomization strategy ensures broad variability in the generated content while preserving physical plausibility and photorealism.

Scene Environment: The first critical aspect of the environment is the lighting configuration. Sky- and sun-based illumination is controlled through Blender's *SkyTexture* node, allowing the simulation of different times of day and varying atmospheric conditions (e.g., air, dust, and ozone densities). Furthermore, a volumetric fog effect with randomized density is incorporated to emulate realistic atmospheric disturbances, thereby reducing target visibility.

Scene Objects and Trajectories: Aircrafts serve as the primary objects of interest in the scene, as they constitute the targets to be detected and tracked. Four 3D airplane models from the Blender *traffiq* add-on [8] are employed, with randomized colors and both uniform and slightly non-uniform scaling applied to enhance variability. A random number $N \in [10, 20]$ of aircraft instances are introduced into each scene and animated according to the following principles:

The airport environment is structured into three main functional areas: parking aprons, taxiways, and runways. To generate realistic motion paths, points assigned to one of the matching categories $\{parking, path, air\}$, are manually placed across the functional areas of the airport (see Figure 3), defining control points for plausible trajectories from parking positions to take-off locations,or vice versa. Together, the control points form a structured graph that encodes the airport's functional topology. For each aircraft, a random sub-graph connecting a chosen parking position with an *air*-labeled node is selected, to construct a smooth spline-representation as a trajectory along which the airplane is animated. The *air*-labeled

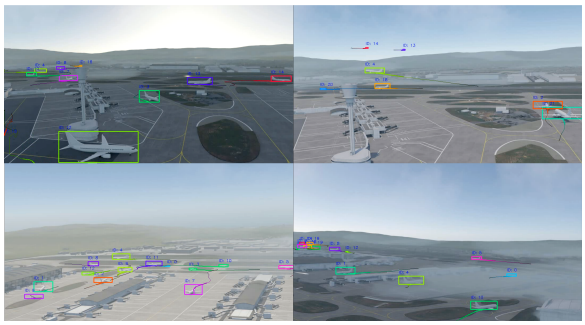


Figure 4: Example synthetic frames with overlaid ground-truth objects and trajectory annotations.

nodes are positioned above ground level to simulate take-off and landing, ensuring that generated trajectories include realistic lift-off and descent dynamics. Trajectory generation follows a procedural scheme: a random *parking*-labeled node is first selected, and its k (k=5) nearest *path*-labeled neighbors are evaluated to construct an initial trajectory segment. Since nearest-neighbor selection alone does not enforce the airport's functional constraints, certain edges with invalid spatial or directional dependencies are explicitly disallowed. This prevents unrealistic movements, such as an aircraft leaving an apron or taxiway at an implausible location or direction. At each step, a permissible next node is chosen from the k candidates, and the process is repeated until an *air*-labeled node is reached, marking the completion of a take-off trajectory. Conversely, landing trajectories are generated in reverse: starting from an *air*-labeled node at a random time, the sub-graph is incrementally extended toward a *parking*-labeled node, resulting in a plausible landing sequence.

Object kinematics play a crucial role in the simulation: since the scene is based on a metrically scaled environment, both metric velocities and velocity limits are utilized to generate diverse dynamic behaviors in the aircraft's motion. Consequently, the edges of the defined sub-graph also carry manually-assigned speed information, allowing for realistic motion patterns, such as slow movement along the taxiway followed by a rapid take-off.

Scene Observers: Fourteen virtual cameras were strategically positioned at various airport locations to simulate realistic surveillance setups, capturing aircraft with typical size distributions. To ensure significant view diversity and prevent the neural learning model from overfitting to specific camera angles, the view for each generated sequence was varied by randomizing both camera intrinsics (focal length) and extrinsics (orientation) within predefined bounds. Figure 4 illustrates representative synthetic views together with their corresponding ground-truth overlays.

3 BASELINE EXPERIMENT AND RESULTS

The attributes of the data produced by the synthetic generation pipeline are shown in Table 1. To assess the representational quality and the minimal *sim-to-real* gap of the generated image data and its annotations, we conducted a neural learning experiment using the CenterTrack [12] unified detection and tracking framework.

Model training: We trained a CenterTrack model [12] on 20,850

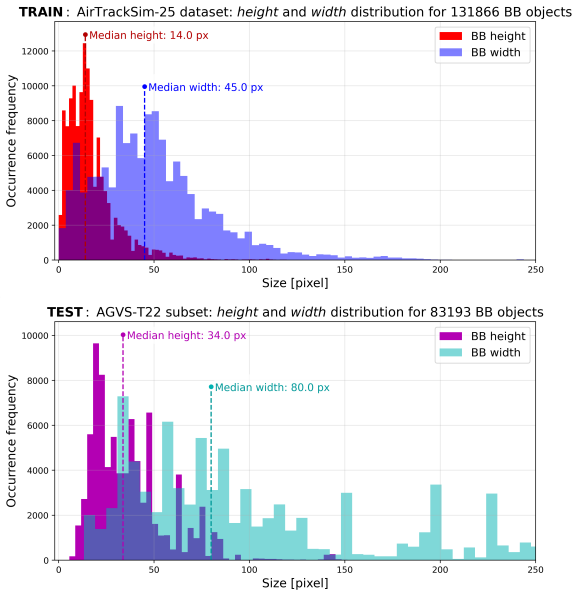


Figure 5: Bounding box height and width distributions within the training (top) and test (bottom) datasets. Each plot shows two semi-transparent histograms, to reveal all distribution details.

# views	# frames	img. resolution	# objects	2D BB	3D BB	2D key-points	occlusion stat.	track-ID
14	23,278	1088 × 608 px	131,866	✓	✓	✓	✓	✓

Table 1: The main *AirTrackSim25* dataset attributes.

image frames from the *AirTrackSim25* dataset, holding out an independent validation set of 2,428 images. We used an input resolution of 1088×608 pixel, the *dla34* backbone, initial learning rate of 10^{-4} and a batch size of 14. We refer to this model as *V50*. For comparison, an otherwise identical CenterTrack model was trained with the same settings on the real-world *MAV* dataset [9] to establish a real-domain detector baseline; we denote this model as *CT*.

Test dataset: For evaluation, we used eight scenarios from the AGVS-T22 dataset [6], comprising a total of 10,642 manually annotated image frames with bounding boxes and track IDs, resized to the image resolution of 1088×608 pixel.

Comparison of training and test data: To assess the representational quality of airplane objects in the proposed synthetic dataset, we computed distributional statistics of bounding box dimensions across all frames. Using the same input resolution, equivalent statistics were also obtained for the test dataset. The resulting distributions for training and test data are shown in Figure 5. As it can be seen, the proposed *AirTrackSim25* dataset predominantly contains small objects, with a median bounding box height of 14 pixels, whereas the test dataset features objects approximately three times larger (median height: 34 pixels). This highlights the suitability of the *AirTrackSim25* dataset for advancing research on small object detection and tracking. Furthermore, these findings motivate future extensions of the dataset to incorporate larger aircraft instances.

MOT results: Figure 6 and Table 2 present a qualitative and quantitative comparison between the *CT* and *V50* models, trained on real and synthetic data, respectively. As illustrated, the synthetically trained *V50* model exhibits substantially higher sensitivity,

SEQ/METHOD	FRAMES	idf1	idp	idr	RECALL	PREC	OBJECTS	mostly_tracked	part_tracked	mostly_lost	N_falsepos	N_misses	N_switches	N_fragm	MOTP
baseline-8/CT	2751	0.65	0.61	0.71	0.88	0.76	31115	9	3	0	8683	3582	13	33	0.35
baseline-8/V50	2751	0.46	0.33	0.78	0.98	0.41	31115	12	0	0	43108	602	33	41	0.41
multi-scale-1/CT	1107	0.52	0.57	0.47	0.65	0.79	20874	12	6	4	3622	7218	16	47	0.39
multi-scale-1/V50	1107	0.44	0.37	0.54	0.77	0.52	20874	14	6	2	14599	4868	68	118	0.45
multi-scale-2/CT	1685	0.58	0.69	0.51	0.58	0.78	6156	3	1	1	1018	2589	7	3	0.19
multi-scale-2/V50	1685	0.30	0.19	0.71	0.82	0.23	6156	3	1	1	17295	1081	37	14	0.34
weather-1/CT	1516	0.34	0.41	0.28	0.55	0.79	12208	2	5	2	1760	5541	11	9	0.30
weather-1/V50	1516	0.28	0.22	0.41	0.97	0.51	12208	9	0	0	11185	347	44	35	0.53
weather-3/CT	1217	0.74	0.92	0.61	0.64	0.97	3430	2	0	1	75	1218	4	0	0.17
weather-3/V50	1217	0.43	0.30	0.77	0.86	0.33	3430	2	1	0	5881	496	5	4	0.50
motion-6/CT	461	0.73	0.99	0.54	0.47	0.98	3175	2	2	4	29	1698	2	232	0.26
motion-6/V50	461	0.68	0.77	0.56	0.45	0.69	3175	2	2	4	649	1743	5	238	0.42
ptz-2/CT	1161	0.29	0.23	0.39	0.49	0.29	5553	2	1	4	6603	2814	1	3	0.28
ptz-2/V50	1161	0.15	0.11	0.25	0.40	0.17	5553	1	2	4	10696	3312	14	43	0.43
lc-5/CT	744	0.07	0.04	0.21	0.32	0.06	682	0	1	0	3334	466	1	2	0.22
lc-5/V50	744	0.01	0.01	0.15	0.29	0.01	682	0	1	0	14920	484	3	29	0.63

Table 2: MOT Metrics Summary at IoU=0.75

resulting in improved recall. However, this comes at the cost of reduced precision, as false alarms increase. In particular, when object size decreases, the model shows reduced specificity for airplanes, occasionally detecting and tracking other small objects (e.g., birds or ground vehicles). Additional false alarms are also triggered by object-like image structures and noise artifacts. Conversely, the *CT* model trained on real data maintains a low false alarm rate but fails to reliably detect distant or partially occluded targets. The MOT evaluation across eight scenarios, summarized in Table 2, confirms these trends. The increased sensitivity of the *V50* model leads to a higher number of *mostly-tracked* targets—including distant and partially occluded ones—while also producing a higher rate of stationary false alarms. Challenging scenarios such as *motion-6*, *ptz-2*, and *lc-5*, which contain motion or imaging artifacts absent in synthetic training data, are particularly problematic for the *V50* model. Future work will therefore concentrate on extending the dataset to achieve a more balanced distribution of aircraft sizes and to incorporate representative real-world artifacts, with the objective of supporting models that attain both high recall and high precision.

Figure 6: Detection/tracking results by two baselines.

4 CONCLUSIONS

We introduce a synthesis framework and the *AirTrackSim25* synthetic dataset, designed to address the scarcity of annotated data for airport surveillance and to support research in object detection, multi-object tracking, and data association. The dataset offers diverse wide-area airport perspectives with realistic aircraft dynamics and comprehensive annotations, including 2D/3D bounding boxes, key-points, and trajectory information. Its applicability to MOT tasks is demonstrated through benchmarking with a baseline neural learning framework, underscoring its relevance for advancing small object detection and tracking.

ACKNOWLEDGMENTS

This work was supported by the SAFER project, funded by the Austrian Research Promotion Agency (FFG) under project number 4452868.

REFERENCES

[1] P. Dendorfer, H. Rezatofighi, A. Milan, J. Shi, D. Cremers, I. Reid, S. Roth, K. Schindler, and L. Leal-Taixé. 2020. MOT20: A benchmark for multi object tracking in crowded scenes. *arXiv:2003.09003[cs]* (March 2020). http://arxiv.org/abs/1906.04567

[2] Stichting Blender Foundation. 2025. Blender - a 3D modelling and rendering package. https://www.blender.org. Accessed: 2025-08-21.

[3] Andreas Geiger, Philip Lenz, and Raquel Urtasun. 2012. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

[4] META Group. 2020. London Heathrow Airport - LHR 3D model. https://www.turbosquid.com/3d-models/london-heathrow-airport-lhr-1543630. Accessed: 2025-08-21.

[5] Riku Kanayama, Yuki Yoshida, and Yuki Kondo. 2025. Baseline code for SMOT4SB by IIM-TTIJ. https://www.mva-org.jp/mva2025/challenge

[6] Tingyu Li, Xiang Zhang, Zihao Tang, and Yudie Liu. 2023. AGVS-T22: A New Multiple Object Tracking Dataset for Airport Ground Video Surveillance. In *2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)*. 4380–4387. https://doi.org/10.1109/ITSC57777.2023.10421871 ISSN: 2153-0017.

[7] Jay Patrikar, Joao Dantas, Brady Moon, Milad Hamidi, Sourish Ghosh, Nikhil Keetha, Ian Higgins, Atharva Chandak, Takashi Yoneyama, and Sebastian Scherer. 2025. Image, speech, and ADS-B trajectory datasets for terminal airspace operations. *Scientific Data* 12, 1 (2025), 468.

[8] polygoniq xyz s.r.o. 2025. traffiq: vehicle library for blender. https://polygoniq.com/software/traffiq. Accessed: 2025-08-21.

[9] Daniel Steininger, Verena Widhalm, Julia Simon, Andreas Kriegler, and Christoph Sulzbachner. 2021. The Aircraft Context Dataset: Understanding and Optimizing Data Variability in Aerial Domains. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*. 3823–3832.

[10] Yihong Xu, Yutong Ban, Guillaume Delorme, Chuang Gan, Daniela Rus, and Xavier Alameda-Pineda. 2022. TransCenter: Transformers with Dense Representations for Multiple-Object Tracking. https://doi.org/10.48550/arXiv.2103.15145 arXiv:2103.15145 [cs].

[11] Yifu Zhang, Chunyu Wang, Xinggang Wang, Wenjun Zeng, and Wenyu Liu. 2021. FairMOT: On the Fairness of Detection and Re-Identification in Multiple Object Tracking. *International Journal of Computer Vision* 129, 11 (Nov. 2021), 3069–3087. https://doi.org/10.1007/s11263-021-01513-4 arXiv:2004.01888 [cs].

[12] Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. 2020. Tracking Objects as Points. https://doi.org/10.48550/arXiv.2004.01177 arXiv:2004.01177 [cs].

Sphere Target-Based Point Cloud Registration in a Railway Safety Application

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

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

Sašo Pečnik
saso.pecnik@um.si
University of Maribor
Faculty of Electrical Engineering and Computer Science
Maribor, Slovenia

Blaž Repnik
blaz.repnik@um.si
University of Maribor
Faculty of Electrical Engineering and Computer Science
Maribor, Slovenia

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

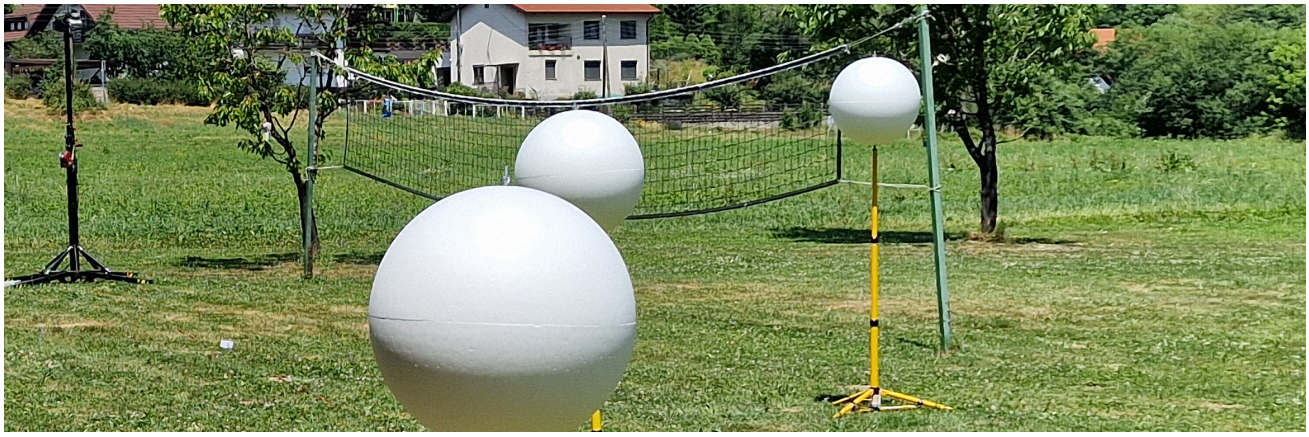


Figure 1: A part of the test environment with three (of six) sphere targets and a LiDAR scanner (of two) left behind.

Abstract

The paper introduces a variation of registration of two LiDAR point clouds based on sphere targets, which uses our own geometric construction method to determine the centres and radii of the spheres. Tests gave encouraging results for use in a railway safety application, as the registration error is below 30 % of the voxel size used there. Unlike the traditional landmark identification approach based on solving the system of linear equations, the proposed approach offers good geometric interpretability and error explainability, which has potential for the development of heuristics that would prune the solution space and, eventually, enable us to use the time saved to conduct a more detailed investigation in the vicinity of the current optima.

Keywords

point cloud registration, sphere target, LiDAR, sphere centre determination, radius determination, level crossing

Information Society 2025, 6–10 October 2025, Ljubljana, Slovenia

1 Introduction

Similar to the human visual system, spatial data capture devices such as cameras and scanners acquire information within a limited field of view (FoV). Furthermore, the density, and hence the quality and usefulness, of the acquired data decreases with distance from the capturing sensor. Therefore, combining individual shots of the same area in order to improve the processing and analysis of spatial data has been common practice since the pre-computer era. Examples include image stitching, e.g., in creating panoramic images, the use of overlapping transparency layers in cartography and film effects, as well as stereographic and photogrammetric techniques. With the development of 3D geometric modelling and the advent of 3D data capture devices, a convenient capability has emerged to capture a 3D scene from multiple viewpoints and merge the captured data. This is achieved by spatially aligning the local coordinate systems of two or more geometric models with respect to a reference coordinate system and thereby aligning the models themselves, which is called geometric data registration or simply registration [11].

In this work, we focus on geometric models in the form of point clouds. Furthermore, we focus on rigid transformations, which are common when dealing with time-aligned point clouds [2]. For simplicity, we consider the registration of two point clouds, as additional ones can be handled incrementally. Specifically, we want to align (register) two point clouds captured by

a pair of LiDAR (Light Detection and Ranging) scanners that monitor events at a level crossing between a railway and a road. The problem here is that the central area is quite far from both scanners, which means that the points captured there are quite far apart, causing problems in detecting unwanted obstacles in each single scan. By registering and merging two point clouds, we obtain a denser merged point cloud that improves detection.

Comprehensive reviews of point cloud registration methods can be found, e.g., in [9, 5, 3, 4]. They are usually divided into target-based and target-free methods [8]. Targets are (easily) identifiable markers artificially inserted into the scene to simplify alignment. The target-based methods align pairs of models of the same target from both point clouds, indirectly aligning the entire scene. The target-free methods are based on features that are referred to as natural targets by some authors. These include, e.g., edges, prominent vertices, and contrasting areas, which are more difficult to detect because their geometry, size, and location are less known. The target-free methods are typically more general, also useful in dynamic environments and, recently, increasingly based on deep learning [10]. In this paper, we focus on target-based methods. Using the points detected by both devices on the target surface, they can calculate additional points (landmarks) where there is a perfect match, and then align the scene accurately by aligning these landmarks.

In Section 2, we present the registration method we developed and used, which consists of preprocessing, sphere target extraction, landmark identification, and registration itself. In Section 3, we describe the test environment, present the registration results, and analyze their accuracy. In Section 4, we summarize the work and present challenges for further improvements and research. The main scientific contributions of the article are the original geometric construction procedure for calculating the centre and radius of a sphere and the analysis of accuracy.

2 Methodology

We use two stationary LiDARs, which means we can perform registration before launching the application itself. For accuracy, we use artificial targets, as we have enough time to safely set them up, identify them accurately, and then remove them. Based on the literature [1, 8], in which various target types and layouts were assessed, and our own experiments, in which we focused on the performance of both LiDARs used and the possibility of separating target and stand points in a simulated railway level crossing layout, we chose sphere targets. The procedure is carried out in four phases described in Subsections 2.1–2.4.

2.1 Preprocessing

This step depends on the initial state of the data. It may include mapping the data from perspective to orthographic projection in order to establish Cartesian coordinate systems CS_i for each device ($i \geq 2$), scaling to unify the unit of measurement in all CS_i, cropping, and various filtering, e.g., denoising. In our case, each CS_i is centred at the corresponding LiDAR lens and is left-handed. The Z-axis points towards the centre of the FoV (diagonally downward), the X-axis horizontally to the left, and the Y-axis in the direction of $Z \times X$, i.e., diagonally upward. The unit of measurement is metre.

2.2 Target Extraction

The next step is to isolate individual target geometric models from the scene. In our case, we determine which points in a point cloud belong to individual target spheres. Our focus is primarily on accuracy rather than speed, automation, and generality of the process. Thereby, we used specific target geometry, such as spheres' instalation heights and their radii. Unwanted points of the stand are above or below the sphere and in the area around its central vertical axis. As a last resort, there is also the option of interactively deleting some unwanted points. More advanced approaches, based on similar principles as registration with natural targets can be found in, e.g., [7].

2.3 Landmark Identification

Although two LiDARs capture different points on the surface of a single sphere target, the centre of the sphere is unique. In this Subsection, we present a procedure for determining the sphere centre from four non-coplanar points on its surface. Three non-collinear sphere centres, given in the local coordinate systems of both LiDARs, can be used as landmarks to establish an intermediate coordinate system. Based on this, we will then introduce the registration transformation matrix M in Subsection 2.4.

Let $P_1(x_1, y_1, z_1)$, $P_2(x_2, y_2, z_2)$, $P_3(x_3, y_3, z_3)$, and $P_4(x_4, y_4, z_4)$ represent four non-coplanar points in $\mathbb{R}^3$. We want to describe the sphere $S(r, C)$ through them, which means we want to determine its centre $C(x_C, y_C, z_C)$ and radius r. Applying a well-known equation of a sphere to such quartet of points gives the system of four quadratic equations (1). Subtracting the other three equations separately from the first one gives a system of linear equations (2) in three unknowns x_C, y_C, and z_C.

$$(x_i - x_C)^2 + (y_i - y_C)^2 + (z_i - z_C)^2 = r^2, i = 1, ..., 4 \qquad (1)$$

$$\begin{bmatrix} 2(x_1 - x_2), & 2(y_1 - y_2), & 2(z_1 - z_2) \\ 2(x_1 - x_3), & 2(y_1 - y_3), & 2(z_1 - z_3) \\ 2(x_1 - x_4), & 2(y_1 - y_4), & 2(z_1 - z_4) \end{bmatrix} \begin{bmatrix} x_C \\ y_C \\ z_C \end{bmatrix} = \begin{bmatrix} (x_1^2 + y_1^2 + z_1^2) - (x_2^2 + y_2^2 + z_2^2) \\ (x_1^2 + y_1^2 + z_1^2) - (x_3^2 + y_3^2 + z_3^2) \\ (x_1^2 + y_1^2 + z_1^2) - (x_4^2 + y_4^2 + z_4^2) \end{bmatrix} \qquad (2)$$

The system (2) can then be solved using traditional methods, such as Gaussian elimination, inverse matrix calculation, etc. Once we have the centre C of the sphere, we calculate the radius r by inserting the coordinates of any point P_1, P_2, P_3, or P_4 into the corresponding line of (1).

This procedure is easy to understand and not difficult to implement, but it also has its drawbacks. The results in Subsection 3.2 show that selecting different four-point sets on the sphere produces significantly different results. Thus, the registration is an optimization problem. Below, we present our own construction approach, which has a clear geometric interpretation. Errors are easier to explain, as individual points of a quartet have different, precisely defined roles.

Athough our alternative method is completely intuitive, we have not yet met it in literature. It first considers three non-collinear points P_1, P_2, P_3 from a quartet on the sphere target $S(r, C)$. They define a plane Σ_{123}, which divides S into two parts: a larger one forming a spherical zone of one base, within which the sphere centre C is located, and a smaller one forming a spherical cap. The intersection $S \cap \Sigma_{123}$ is the circumreference $c_{123} = \texttt{circ}(P_1, P_2, P_3) = \texttt{circ}(r_{123}, C_{123})$ of the triangle $\Delta P_1 P_2 P_3$. C_{123} and r_{123} represent the centre and radius of c_{123}, respectively. Figure 2a shows (initially) known geometric elements and attributes in black, those calculated so far in blue, and those not

yet determined in red. These three colours are used consistently in Figures 2b–c. An important observation at this stage is that the centre C of the sphere must represent the vertex of a right circular cone above c_{123}, i.e. it lies on l_{123}. In the second step, we determine the orthogonal distance f from the fourth point P_4 of the quartet to Σ_{123}, the distance $e = |P_4C_{123}|$, and angle β (Figure 2b). Finally, we use e, β, and the cosine theorem in triangles $\Delta P_4C_{123}C$ and ΔP_1CC_{123} to determine the distance $d = |CC_{123}|$ and the sphere target radius r. Through d, we then also determine the sphere centre C (Figure 2c).

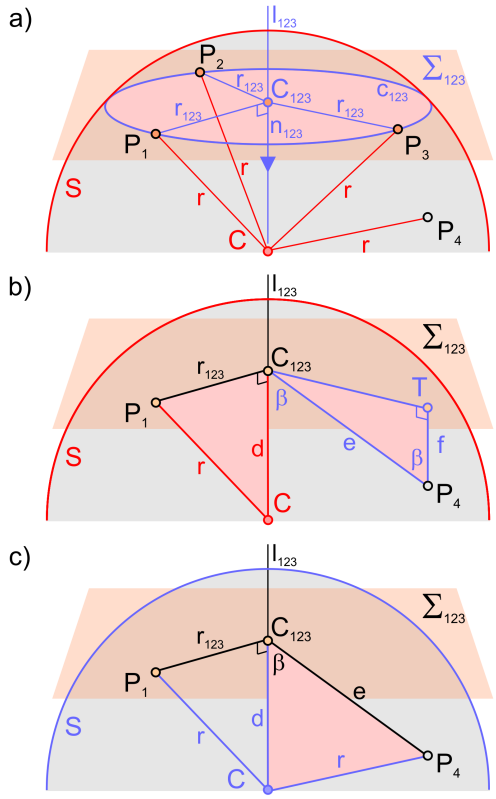


Figure 2: Determination of the sphere centre C and radius r from points P_1, P_2, P_3, and P_4 on the sphere, when P_4 and C are on the same side of Σ_{123}: a) determining Σ_{123}, b) describing the relation between P_4 and Σ_{123}, c) final solution.

Algorithm 1 explains this procedure in a compact form. Note that Figure 2 only shows the case where P_4 and C are on the same side of Σ_{123}. The situation with P_4 in the spherical cap is handled by assigning the absolute value to d, while the exception with both P_4 and C in Σ_{123} is caught with the first if-clause.

2.4 Registration

The determination of the registration matrix M is adopted from [6]. There, it was used to map the point cloud from the CS_S coordinate system of the LiDAR scanner to the robot's CS_R via the intermediate CS_I. Here, S (source) denotes the first LiDAR, R (reference, registered) the target coordinate system of the second LiDAR, and I the intermediate coordinate system. M is determined by three translations and three rotations along/around the coordinate axes. The role of registration is thus to determine the parameters of these six elementary transformations with the best possible accuracy. A triplet of landmark points A, B, and C

Algorithm 1 Centre and radius of the sphere from 4 points

function SphereThroughFourPoints(P_1, P_2, P_3, P_4)
  $C_{123} \leftarrow$ centre of circumreference c_{123} of $\Delta P_1P_2P_3$
  $r_{123} \leftarrow |C_{123}P_1|$ ▷ Radius of c_{123}
  $\mathsf{n}_{123} \leftarrow \dfrac{(P_1 - C_{123}) \times (P_2 - C_{123})}{|(P_1 - C_{123}) \times (P_2 - C_{123})|}$ ▷ Norm. vect. $\Delta P_1P_2P_3$
  if $|C_{123}P_4| = r_{123}$ **then**
    return (C_{123}, r_{123})
  end if
  $T_4 \leftarrow$ orthogonal projection of P_4 on the plane of $\Delta P_1P_2P_3$
  $e \leftarrow |C_{123}P_4|$
  $d \leftarrow \left| \dfrac{e^2 - {r_{123}}^2}{2\,|P_4T_4|} \right|$
  $r \leftarrow \sqrt{d^2 + {r_{123}}^2}$
  if $(\mathsf{n}_{123} \cdot (P_4 - C_{123}) > 0) \oplus (e > r_{123})$ **then**
    $C \leftarrow C_{123} + d\,\mathsf{n}_{123}$
  else $C \leftarrow C_{123} - d\,\mathsf{n}_{123}$
  end if
  return (C, r)
end function

(see Equation 3 and Figure 3) is used to establish CS_I with the origin O and orthogonal unit coordinate vectors U, V and W. M is then computed as a composition of two transformations – M_{S2I} from CS_S to CS_I, and M_{I2R} from the latter to CS_R. Note that the system of formulas (3) and the interpretation from Figure 3 must be employed separately for $\{A_S, B_S, C_S, O_S, U_S, V_S, W_S\}$ expressed in CS_S, and $\{A_R, B_R, C_R, O_R, U_R, V_R, W_R\}$ expressed in CS_R.

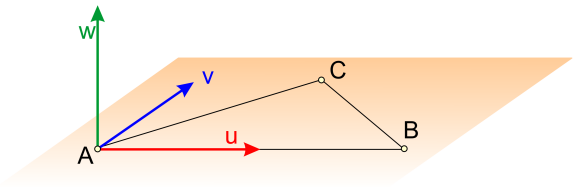


Figure 3: Construction of the intermediate coordinate system from non-collinear A, B and C (adapted from [6]).

$$\begin{aligned} O &= A, \\ U &= \frac{B - O}{|B - O|}, \\ W &= \frac{U \times (C - O)}{|U \times (C - O)|}, \\ V &= W \times U. \end{aligned} \tag{3}$$

The transformation M_{I2R} from CS_I to CS_R is given in homogeneous coordinates as the composition of a 3D rotation Rot_R and translation $Tran_R(O_R)$, as shown in (4). The matrix M_{I2S} from CS_I to CS_S can be generated in the same manner, but the inverse M_{S2I}, as shown in (5), is actually needed. M is then obtained as the composition $M = M_{I2R}M_{S2I}$.

$$M_{I2R} = Tran_R(O_R) \cdot Rot_R = \begin{bmatrix} U_R.x & V_R.x & W_R.x & O_R.x \\ U_R.y & V_R.y & W_R.y & O_R.y \\ U_R.z & V_R.z & W_R.z & O_R.z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4}$$

$$M_{S2I} = M_{I2S}^{-1} = Rot_S^{-1} \cdot Tran_S^{-1}(O_S) = Rot_S^T \cdot Tran_S(-O_S)$$

$$= \begin{bmatrix} U_S.x & U_S.y & U_S.z & 0 \\ V_S.x & V_S.y & V_S.z & 0 \\ W_S.x & W_S.y & W_S.z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 & -O_S.x \\ 0 & 1 & 0 & -O_S.y \\ 0 & 0 & 1 & -O_S.z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5)$$

3 Results

3.1 Test Setup

In tests conducted on the lawn of the primary school in Kamnica (Figure 1), we simulated conditions at an actual railway level crossing, which would be monitored by two LiDARs at the diagonal ends of the crossing. The optical properties, installation height, and viewing angle of LiDARs were also adapted to the expected conditions. We used two Ouster OS1 LiDARs with a vertical resolution of 128 channels, a horizontal resolution of 2048 samples, and a field of view of 42.4° vertically and 360° horizontally. They were 40 m apart, at a height of 3.2 m, and inclined to cover the ground from 3 m onwards. Approximately halfway between them, we placed six styrofoam sphere targets with radii of 25 cm and at different heights, making sure that no three sphere centres were collinear. We manually measured their circumferences in several directions and concluded that the deviations of radii were below 2 mm. Given the optical properties of LiDARs, the nearest neighbour points captured at a distance of 20 m are approximately 6 cm apart in the horizontal direction and 11 cm in the vertical direction. In our railway safety application, we use voxels with sides of 12 cm, which are sufficient for reliable detection of standardized minimum obstacles of $100 \times 50 \times 50$ cm. The spheres were mounted on slender stands, which we "erased" using the procedure described in Subsection 2.2.

3.2 Accuracy Analysis

Each LiDAR acquired between 10 and 50 points on each sphere. We considered 10 frames for each LiDAR, i.e., 10 pairs of point clouds. For each pair, we tested all possible quartets of points on each sphere and selected the best result. After the Landmark identification step, we first checked whether both calculated radii were in the range [24.8 cm, 25.2 cm], and then selected the pair with the smallest centre offset. The centre offset is the deviation between the calculated centres of the same sphere, one originally from CS_R and the other one transformed (registered) from CS_S. The Excel spreadsheets of results contain a total of almost 200 MB of data. Table 1 shows the results of the best alignment. Of the two calculated radii, we write down the worse one, i.e. the one that deviates more from the expected 25 cm. Both metrics gave encouraging results with the radius error below 0.3 mm and the centre offset below 3.5 cm, which is half better than LiDAR resolution applied 20 m from the lens, and below 30 % of the used voxel size. Note that the worst among best offsets in individual frames was 6.1 cm (probably due noise). Besides, there were always quartets of points found that gave completely unusable results, as the calculated radii ranged from 5 cm to 2 m.

Table 1: Sphere identifier, calculated radius [cm], and the distance between both calculated centres [cm]

Sphere ID	Calculated radius	Centre offset
1	25.02	1.15
2	25.02	1.47
3	25.03	0.34
4	25.04	3.13
5	24.99	1.73
6	25.02	2.30

4 Conclusion

We presented a variation of the registration of two LiDAR point clouds based on sphere targets, which uses our own geometric construction method to determine the centres and radii of the spheres. Tests have shown that the method achieves encouraging results for use in a railway safety application for level crossing monitoring. The registration error is below 30 % of the voxel side length used there. Unlike the traditional landmark identification approach based on solving the system of linear equations, the proposed approach offers good geometric interpretability and error explainability, which has potential for the development of heuristics that would prune the solution space and, eventually, enable us to use the time saved to conduct a more detailed investigation in the vicinity of the current optima. Current experience suggests that the best quartets are those where all normals are directed as closely as possible toward the LiDAR, while at the same time the points are not too close together and are as non-coplanar as possible (defining a tetrahedron with a larger volume).

Acknowledgements

The research was funded by the Slovene Research and Innovation Agency under Research Project J2-4458 and Research Programme P2-0041. The authors are grateful to Fokus Tech d.o.o. from Celje and OŠ Kamnica for providing equipment and testing facilities.

References

[1] Burcin Becerik-Gerber, Farrokh Jazizadeh, Geoffrey Kavulya, and Gulben Calis. 2011. Assessment of target types and layouts in 3d laser scanning for registration accuracy. *Autom. Constr.*, 20, 5, 649–658. doi:10.1016/j.autcon.2010.12.008.

[2] Ben Bellekens, Vincent Spruyt, Rafael Berkvens, Rudi Penne, and Maarten Weyn. 2015. A benchmark survey of rigid 3d point cloud registration algorithms. *Int. J. Adv. Intell. Syst*, 8, 5, 118–127. http://72.52.166.99/articles/intsys_v8_n12_2015_10.pdf.

[3] Menthy Denayer, Joris De Winter, Evandro Bernardes, Bram Vanderborght, and Tom Verstraten. 2024. Comparison of point cloud registration techniques on scanned physical objects. *Sensors*, 24, 7, 2142. doi:10.3390/s24072142.

[4] Mengjin Lyu, Jie Yang, Zhiquan Qi, Ruijie Xu, and Jiabin Liu. 2024. Rigid pairwise 3d point cloud registration: a survey. *Pattern Recognit.*, 110408. doi:10.1016/j.patcog.2024.110408.

[5] Jaroslav Marek and Pavel Chmelař. 2023. Survey of point cloud registration methods and new statistical approach. *Mathematics*, 11, 16, 3564. doi:10.3390/math11163564.

[6] David Podgorelec, Suzana Uran, Andrej Nerat, Božidar Bratina, Sašo Pečnik, Marjan Dimec, Franc Žaberl, Borut Žalik, and Riko Šafarič. 2023. Lidar-based maintenance of a safe distance between a human and a robot arm. *Sensors*, 23, 9, 4305. doi:10.3390/s23094305.

[7] Trung-Thien Tran, Van-Toan Cao, and Denis Laurendeau. 2016. Esphere: extracting spheres from unorganized point clouds: how to extract multiple spheres accurately and simultaneously. *Vis. Comput.*, 32, 1205–1222. doi:10.1007/s00371-015-1157-0.

[8] Tilen Urbančič, Žiga Roškar, Mojca Kosmatin Fras, and Dejan Grigillo. 2019. New target for accurate terrestrial laser scanning and unmanned aerial vehicle point cloud registration. *Sensors*, 19, 14, 3179. doi:10.3390/s19143179.

[9] Dongfang Xie, Wei Zhu, Fengxiang Rong, Xu Xia, and Huiliang Shang. 2021. Registration of point clouds: a survey. In *2021 Int. Conf. on Networking Systems of AI (INSAI)*. IEEE, 136–142. doi:10.1109/INSAI54028.2021.00034.

[10] Ningli Xu, Rongjun Qin, and Shuang Song. 2023. Point cloud registration for lidar and photogrammetric data: a critical synthesis and performance analysis on classic and deep learning algorithms. *ISPRS J. Photogramm. Remote Sens.*, 8, 100032. doi:10.1016/j.ophoto.2023.100032.

[11] Dongho Yun, Sunghan Kim, Heeyoung Heo, and Kwang Hee Ko. 2015. Automated registration of multi-view point clouds using sphere targets. *Adv. Eng. Inform.*, 29, 4, 930–939. doi:0.1016/j.aei.2015.09.008.

Indeks avtorjev / Author index

Srednjeevropska konferenca o uporabnem teoretičnem računalništvu in informatiki (MATCOS)

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

Uredniki | Editors:
Andrej Brodnik
Gábor Galambos
Rok Požar