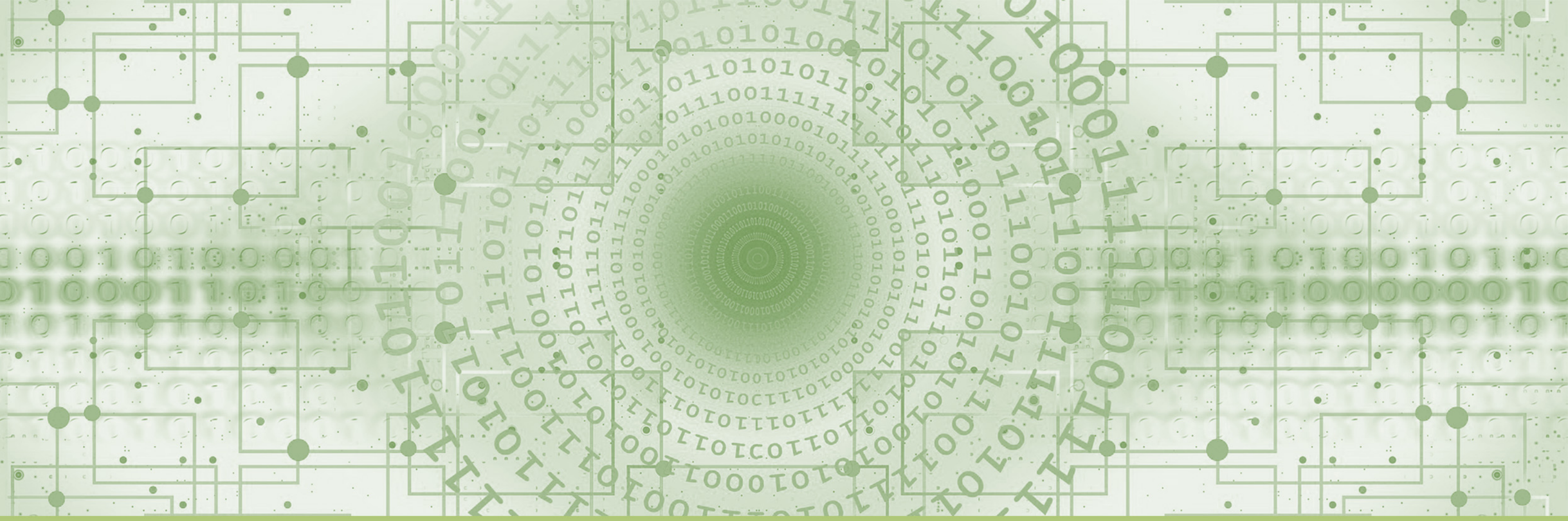
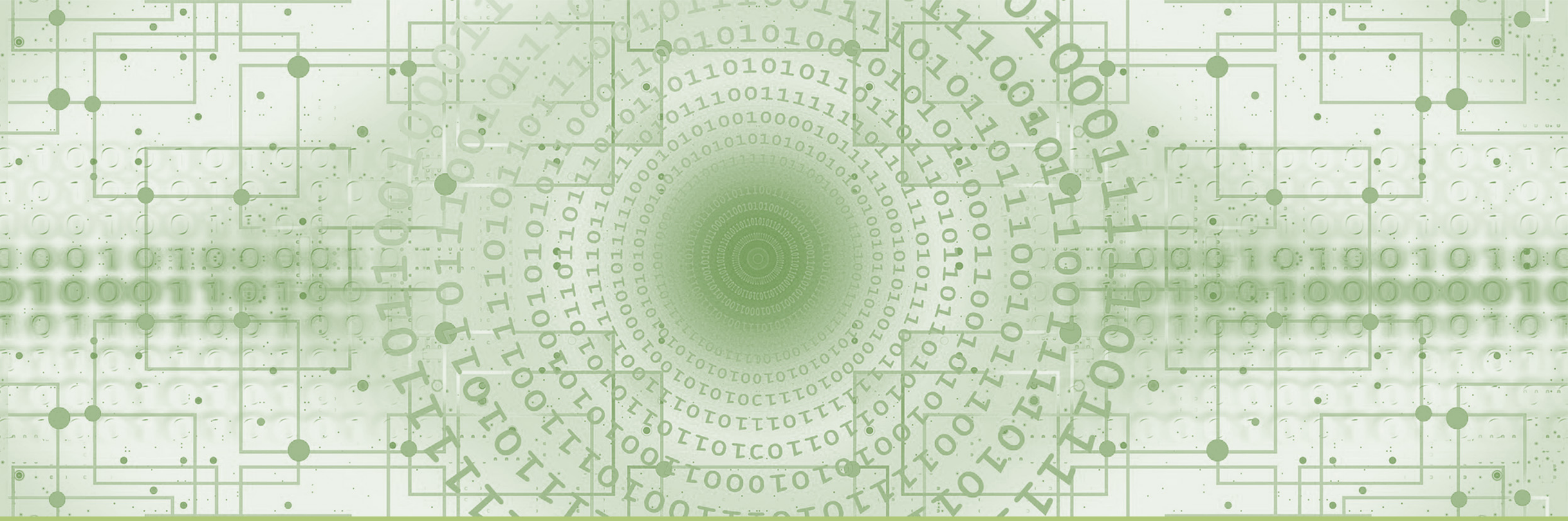


Proceedings of the 2021 7th Student Computer Science Research Conference (StuCoSReC)

Iztok FISTER
Andrej BRODNIK
Janez BREST
Iztok FISTER ml.
Matjaž KRNC
Niko ZIMIC
EDITORS

University of Maribor Press



University of Maribor

Faculty of Electrical Engineering
and Computer Science

Proceedings of the 2021 7th Student Computer Science Research Conference (StuCoSReC)

Editors
Iztok Fister
Andrej Brodnik
Janez Brest
Iztok Fister Jr.
Matjaž Krnc
Niko Zimic

Maribor, September 2021

Title	**Proceedings of the 2021 7th Student Computer Science Research Conference (StuCoSReC)**
Editors	Iztok Fister (University of Maribor, Faculty of Electrical Engineering and Computer Science) Andrej Brodnik (University of Ljubljana, Faculty of Computer and Information Science) Janez Brest (University of Maribor, Faculty of Electrical Engineering and Computer Science) Iztok Fister Jr. (University of Maribor, Faculty of Electrical Engineering and Computer Science) Matjaž Krnc (University of Primorska, Faculty of Mathematics, Natural Sciences and Information Technology) Niko Zimic (University of Ljubljana, Faculty of Computer and Information Science)
Language editing	Shelagh Hedges
Technical editors	Iztok Fister (University of Maribor, Faculty of Electrical Engineering and Computer Science) Jan Perša (University of Maribor, University Press)
Cover graphics	Network, author: geralt (Pixabay.com CC0)
Graphics material	Authors, Editors
Conference	**7th Student Computer Science Research Conference (StuCoSReC)**
Location & date	14th September 2021, Maribor, Slovenia
Program committee	Klemen Berkovič (University of Maribor, Slovenia), Zoran Bosnić (University of Ljubljana, Slovenia), Borko Bošković (University of Maribor, Slovenia), Janez Brest (University of Maribor, Slovenia), Lucija Brezočnik (University of Maribor, Slovenia), Andrej Brodnik (University of Primorska and University of Ljubljana, Slovenia), Patricio Bulić (University of Ljubljana, Slovenia), Mojca Ciglarič (University of Ljubljana, Slovenia), Jani Dugonik (University of Maribor, Slovenia), Iztok Fister (University of Maribor, Slovenia), Iztok Fister Jr. (University of Maribor, Slovenia), Matjaž Gams (Jozef Stefan Institute, Slovenia), Mario Gorenjak (University of Maribor, Slovenia), Aleš Holobar (University of Maribor, Slovenia), Andres Iglesias (Universidad de Cantabria, Spain), Sašo Karakatič (University of Maribor, Slovenia), Branko Kavšek (University of Primorska, Slovenia), Domen Kavran (University of Maribor, Slovenia), Štefan Kohek (University of Maribor, Slovenia), Matjaž Krnc (University of Primorska, Slovenia), Miklos Kresz (University of Szeged, Hungary), Niko Lukač (University of Maribor, Slovenia), Marjan Mernik (University of Maribor, Slovenia), Uroš Mlakar (University of Maribor, Slovenia), Eneko Osaba (University of Deusto, Spain), Vili Podgorelec (University of Maribor, Slovenia), Jan Popič (University of Maribor, Slovenia), Miha Ravber (University of Maribor, Slovenia), Peter Rogelj (University of Primorska, Slovenia), Martin Šavc (University of Maribor, Slovenia), Alexandros Tzanetos (University of Aegean), Damjan Vavpotič (University of Ljubljana, Slovenia), Grega Vrbančič (University of Maribor, Slovenia), Nikolaj

Zimic (University of Ljubljana, Slovenia), Gilbert Lim Yong San (Singapore Eye Research Institute) & Borut Žalik (University of Maribor, Slovenia).

Local organizing committee: Janez Brest (University of Maribor, Slovenia), Iztok Fister (University of Maribor, Slovenia) & Aleš Holobar (University of Maribor, Slovenia).

Published by *Založnik*
University of Maribor
University Press
Slomškov trg 15, 2000 Maribor, Slovenia
https://press.um.si, zalozba@um.si

Co-published by *Izdajatelj*
University of Maribor
Faculty of Electrical Engineering and Computer Science
Koroška cesta 46, 2000 Maribor, Slovenia
http://www.feri.um.si, feri@um.si

Co-published by *Soizdajatelj*
University of Ljubljana
Faculty of Computer and Information Science
Večna pot 113, 1000 Ljubljana, Slovenia
https://fri.uni-lj.si/en, dekanat@fri.uni-lj.si

Co-published by *Soizdajatelj*
University of Primorska
Faculty of Mathematics, Natural Sciences and Information Technologies
Glagoljaška 8, 6000 Koper, Slovenia
https://www.famnit.upr.si/en; info@upr.si

Edition 1st

Publication type E-book

Published Maribor, September 2021

Availabe at https://press.um.si/index.php/ump/catalog/book/611

```
CIP - Kataložni zapis o publikaciji
Univerzitetna knjižnica Maribor

004(082)(0.034.2)

STUDENT Computer Science Research Conference (7 ; 2021 ; Maribor)
    Proceedings of the 2021 7th Student Computer Science Research Conference
(StuCoSReC) [Elektronski vir] / [editors Iztok Fister ... et al.]. - 1st ed. -
E-zbornik. - Maribor : University of Maribor, University Press : Faculty of
Electrical Engineering and Computer Science ; Ljubljana : Faculty of Computer
and Information Science ; Koper : Faculty of Mathematics, Natural Sciences and
Information Technologies, 2021

Način dostopa (URL): https://press.um.si/index.php/ump/catalog/book/611
ISBN 978-961-286-516-0 (pdf)
doi: 10.18690/978-961-286-516-0
COBISS.SI-ID 76158723
```

ISBN	978-961-286-516-0 (pdf)
DOI	https://doi.org/10.18690/978-961-286-516-0
Price	Free copy
For publisher	prof. dr. Zdravko Kačič, rector of University of Maribor
Citiranje *Attribution*	Fister, I., Brodnik, A., Brest, J., Fister, I. Jr., Krnc, M. & Zimic, N. (eds.) (2021). *Proceedings of the 2021 7th Student Computer Science Research Conference (StuCoSReC).* Maribor: University Press. doi: 10.18690/978-961-286-516-0

Table of Contents

University of Maribor Press

Preface

JANEZ BREST

Computer science is one of the fastest growing fields. We live in a digital age where computers, smart phones and many other devices are connected worldwide. To meet all the requirements of the growing digital world; knowledge, skills and abilities are needed, which can be found in young people, especially in brilliant students.

The Student Computer Science Research Conference (StuCoSRec) is organized by the collaboration of the departments and institutes of three public universities in Slovenia with computer science study programmes. We are very proud that over the past years, these StuCoSRec student conferences have been held every year from 2014 at different institutions in Slovenia. Unfortunately, in 2020, which had been marked by the Covid-19 pandemic, the organization of the StuCosRec conference was not possible. This year we want to hold the conference despite the stormy conditions of the Covid-19. The StuCoSReC conference is intended for computer science students to present their research work and achievements, exchange the ideas, socialize and connect among themselves.

This proceeding contains the papers of the seventh Student Computer Science Research Conference 2021 (StuCoSRec'21) that is planned to be held in Maribor. The University of Maribor – Faculty of Electrical Engineering and Computer Science is proud to host the conference this year. We received seventeen submissions, covering several topics of the computer science. Three submissions are in Slovene and the others are in English. The submissions were reviewed by two reviewers. One submission was rejected. The talks are scheduled to be held during the conference. The organizing committee would like to thank the reviewers for the well performed job.

The conference is dedicated to graduate and undergraduate students of computer science, and therefore it is free of charge. We gratefully acknowledge the support of the Faculty of Electrical Engineering and Computer Science (University of Maribor), especially the Institute of Computer Science.

On Artefact Elimination in High Density Electromyograms by Independent Component Analysis

Aljaž Frančič
University of Maribor,
Faculty of Electrical
Engineering and Computer Science,
Koroška cesta 46, 2000 Maribor, Slovenia
aljaz.francic@um.si

Aleš Holobar
University of Maribor,
Faculty of Electrical
Engineering and Computer Science,
Koroška cesta 46, 2000 Maribor, Slovenia
ales.holobar@um.si

Milan Zorman
University of Maribor,
Faculty of Electrical
Engineering and Computer Science,
Koroška cesta 46, 2000 Maribor, Slovenia
milan.zorman@um.si

Abstract

We propose a novel approach to artefact detection and elimination in high density electromyograms by using previously introduced Activity Index and Independent Component Analysis (ICA). 28 electromyographic recordings of the biceps brachii muscle were analysed for the presence of artefacts. Using the technique presented in this study, we eliminated an average of 1.07 ± 1.18 artefacts per HDEMG recording. The mean number of eliminated artefacs per recording with at least one detected artefact was 1.88 ± 0.96. In the HDEMGs used in our study, each artefact was found in a separate ICA component.

Keywords biomedical signal processing, electromyography, blind source separation, independent component analysis, activity index, artefact

1 Introduction

In humans, movement and locomotion is regulated by muscles. An electrical signal travels from the central nervous system towards muscles, where it is electrically amplified. By using non-invasive surface electromyography, it is possible to detect the subtle changes in the voltage on the surface of the skin that originate from the muscles, even through the skin and the subcutaneous fat layers. Such recordings are called surface electromyograms (SEMG). When an array of tens of electrodes is used, we call the resulting recordings high density electromyograms (HDEMGs). Due to many factors involved in recording of HDEMGs that are often impossible to control for, the resulting recordings are prone to contain artefacts from various sources, such as power line interference, inadequate electrode-skin contact, electrode drift, subpar quality of the equipment, movement artefacts, etc.

A motor unit (MU) is made up of a motor neuron and the skeletal muscle fibers innervated by that motor neuron's axonal terminals. Groups of motor units often work together to coordinate the contractions of a single muscle. We can think of the signal that travels from the motor neuron to the skeletal muscle fiber as a time series of zeroes, which represent no firing, and ones, which represent the firing of a muscle fiber and, thus, the MU. Because of the refractory period, the spikes are few and far between, making the resulting signal sparse in time. This signal is called the MU spike train.

MUs fire asynchronously and their contributions are superimposed into HDEMG, forming a highly complex signals that are difficult to interpret. By using computer-aided methods, such as Convolution Kernel Compensation (CKC) [2], it is possible to decompose the HDEMGs into contributions of individual MUs and, therefore, identify the firing times of MUs. This gives us insight into the status of the motor system in humans.

However, due to the reasons mentioned in the initial paragraph of this section, HDEMGs often contain various artifacts. These artefacts hinder the ability of CKC to decompose the HDEMG into contributions of individual motor units. Also worth noting: a single artefact might sometimes be present in multiple EMG channels at the same time, so eliminating a single HDEMG channel (for a period of time) is not always effective.

In this study, we exemplified our technique for artefact detection and elimination in HDEMG by using previously introduced Activity Index [1] and Independent Component Analysis (ICA) techniques [5].

2 Materials and Methods

2.1 HDEMG model and Activity Index

In isometric contractions of skeletal muscles, HDEMG signals can be modeled by the following convolutive model [3]:

$$\boldsymbol{y}(n) = \mathbf{Ht}(n) + \boldsymbol{\omega}(n) \qquad (1)$$

DOI https://doi.org/10.18690/978-961-286-516-0.1
ISBN 978-961-286-516-0

where

$$\boldsymbol{y}(n) = [y_1(n) \ \dots \ y_1(n-F+1) \ \dots \ y_M(n-F+1)]^T \quad (2)$$

stands for time-wise extended vector of HDEMG signals, with extension factor F set between 10 and 20 [3],

$$\boldsymbol{\omega}(n) = [\omega_1(n) \ \dots \ \omega_M(n-F+1)]^T \quad (3)$$

is the time-wise extended noise vector, and

$$\mathbf{t}(n) = [t_1(n) \dots t_1(n-L-F+1) \dots t_J(n-L-F+1)]^T \quad (4)$$

is similarly extended vector of MU spike trains. Here, the spike train of the j-th MU is defined as

$$t_j(n) = \sum_k \delta(n - \tau_j(k)), \quad j = 1, \ \dots \ , J \quad (5)$$

where $\delta()$ denotes the Delta function and the k-th spike of the j-th MU appears at time $\tau_j(k)$.

The mixing matrix $\mathbf{H}$ comprises L samples long MUAPs of J active MUs, as detected by M uptake electrodes [3].

The Convolution Kernel Compensation (CKC) method [3] estimates the MU filter iteratively as

$$\hat{\mathbf{f}}_j = \hat{\mathbf{f}}_j + \alpha E(g(\hat{t}_j(n))\boldsymbol{y}(n))^T \mathbf{C}_{\boldsymbol{y}}^{-1} \quad (6)$$

$$\hat{\mathbf{f}}_j = \frac{\hat{\mathbf{f}}_j}{\left\|\hat{\mathbf{f}}_j\right\|} \quad (7)$$

where α determines the speed of convergence, $E()$ stands for mathematical expectation, $g(t)$ is a nonlinear weighting function, e. g. $g(t) = log(1+t^2)$ and $\mathbf{C}_{\boldsymbol{y}} = E(\boldsymbol{y}(n)\boldsymbol{y}^T(n))$ represents the correlation matrix of extended HDEMG measurements. After each iteration of Eqs. 6 and 7, the estimate of MU spike train gets updated

$$\hat{t}_j(n) = \hat{\mathbf{f}}_j^T \boldsymbol{y}(n) \quad (8)$$

The Activity Index I_A at a given time n is defined as [3]

$$I_A(n) = \boldsymbol{y}^T(n)\mathbf{C}_{\boldsymbol{y}}^{-1}\boldsymbol{y}(n) \quad (9)$$

Calculating the Activity Index is the first step of the CKC decomposition approach [3] and is susceptible to HDEMG artefacts. Thus, it can also be used to detect them before further processing. If we do not eliminate the artefacts in HDEMG, they can cause problems for decomposition, preventing the convergence of MU filter in Eq. 6 or segmentation of MU firing moments from identified MU spike train $\hat{t}_j(n)$.

2.2 Independent Component Analysis

In signal processing, ICA is a method for separating a multivariate signal into its additive subcomponents. This is done by assuming that the subcomponents are non-Gaussian signals and that they are statistically independent from each other [4]. ICA is a special case of blind source separation.

In our case, the fastICA [5] decomposition was applied to HDEMG, yielding the individual independent components. Deflation was chosen as the decorrelation approach in fastICA decomposition, and the nonlinearity $g(u) = u^3$ was selected in the fixed-point algorithm. The stopping criterion ϵ was set to 0.0001 and the maximum number of iterations was set to 1000 (the default values).

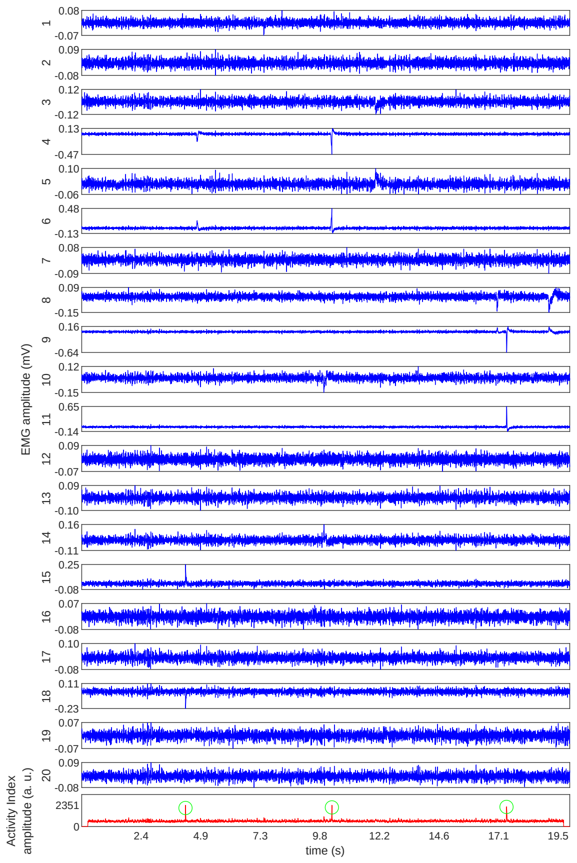


Figure 1: Representative HDEMG channels recorded during the isometric contraction of biceps brachii at 10 % MVC contraction (in blue) and the corresponding Activity Index (in red), with the detected outliers encircled in green. The two most prominent artefacts are noticeable in the 4th and 6th EMG channel at approximately 10th second, in the 9th and 11th EMG channel at approximately 17th second and in the 15th and 18th EMG channel at approximately 4th second. A single artefact is often present in several EMG channels. Also evident, not all the HDEMG artefacts were detrimental for Activity index calculation and, therefore, for MU identification.

2.3 Artefact detection and elimination

An example of a few representative HDEMG channels recorded during isometric contraction of biceps brachii muscle at 10 % of maximum voluntary contraction (MVC) is shown in Fig. 1.

The artefact detection process began by calculating the Activity Index (red line in Fig 1) from HDEMG. In our study, the extension factor F was set to 1. This yielded one time series from several HDEMG channels.

Next, the outliers (denoted by green circles on the red line in Fig 1) were found in the Activity Index. In our case, an outlier was defined as an element that was more than 15 scaled median absolute deviations (MAD) away from the local median. Noteworthy, this could be parameterized and the outliers could be determined in some other fashion. The local median was defined as the median inside the window, where the window size was equal to 2048 samples. To the best of our knowledge there is no established method for artefact elimination using the Activity Index, hence the MAD threshold for determining the outliers in the Activity Index was selected empirically using visual inspection of the results by an expert. As an additional step, we also ignored any outlier that was fewer than 200 samples away from an already detected outlier. In this way, we prevented the multiple identifications of the same artefact.

Afterwards, fastICA decomposition was applied to the HDEMG, yielding the individual independent components. All the identified components were taken into consideration, as we would like to preserve as much of the information as possible after artefact elimination. Each independent component was then excluded and a new Activity Index I_{EX} was calculated without the excluded independent component. Noteworthy, calculating the Activity Index from either HDEMG or all it's ICA components will always yield the same result. Then, for each detected outlier in the Activity Index I_A, we tried to identify the ICA component that contributed the most to the outlier in the Activity Index (to the HDEMG artefact) by observing the difference in the Activity Index at the time of the outlier before and after the exclusion of the individual ICA component. We did this by using the interest metric defined as:

$$IntMet_{EX}(n) = 1 - I_{EX}(n)/I_A(n) \tag{10}$$

at any given time n. In partucular, for each detected outlier at time x in the old Activity Index $I_A(x)$, we looked at all the new Activity Indices $I_{EX}(x)$ (with individual ICA components excluded) and calculated the interest metrics $IntMet_{EX}(x)$ at the time x of the outlier in the Activity Index $I_A(x)$.

The interest metric $IntMet_{EX}$ helped to expose the relationship between the old Activity Index I_A and the new Activity Indices I_{EX}. Higher value of the interest metric corresponded to a greater chance that there was an artefact in the excluded ICA component. Using this metric it was possible to determine which ICA component contains the artefact. It was assumed that only a single ICA component will contain a single artefact, as ICA works under the assumption that the sources are statistically independent from each other. If we were to see artefacts in multiple ICA components at the same time, this would imply that they came from statistically independent sources. This would imply artefact co-occurrence, which is unlikely given our findings about the number of outliers and artefacts found per recording (Section 3). Interest metric threshold, above which we considered the artefacts to be successfully eliminated was set to 0.5.

By using the ICA algorithm it would also possible to reconstruct the original recordings from ICA components. This would allow us to first transform the HDEMGs to ICA space, eliminate the artefacts to the best of our ability and then transform the ICA components back to HDEMG space using simple matrix multiplication. In our current study, we eliminated the whole ICA component, but it would be worth considering "repairing" that component (e.g. locally setting the component elements to 0).

2.4 Dataset and evaluation

To evaluate our method for artefact detection and elimination, we used 20 second long HDEMG recordings from 7 neurologically intact young subjects performing isometric contractions of the biceps brachii muscle, at 5, 10, 15 and 20 % of MVC for a total of 28 HDEMG recordings. 13 × 5 electrode array was used. Visual feedback on force was provided to the participants. All the experiments were conducted in accordance with the Declaration of Helsinki, and were approved by the local Ethical Committee.

In Section 3 we reported the mean ± the Standard Deviation (SD) of the number of outliers found in the Activity Index per HDEMG recording, the mean number of eliminated artefacts per HDEMG recording, the mean number of outliers per HDEMG, where we identified at least one outlier, the mean number of eliminated artefacts per HDEMG where we eliminated at least one artefact as well as the mean interest metric $IntMet_{EX}$ of the eliminated artefacts. We also provided a visual example of an elimination of an artefact.

3 Results

The mean number of outliers found in the Activity Index per HDEMG recording was equal to 1.25 ± 1.29. The mean number of eliminated artefacts per HDEMG recording was equal to 1.07 ± 1.18. The mean number of outliers in Activity Index per HDEMG with at least one outlier was 2.06 ± 1.03 and the mean number of eliminated artefacs per HDEMG with at least one eliminated artefact present was 1.88 ± 0.96. The mean interest metric $IntMet_{EX}$ of the eliminated artefacts was 0.85 ± 0.11. In the HDEMGs used in our study, each artefact was found in a separate ICA component. A representative example of our results is provided in Fig. 2.

4 Discussion

Our results indicated that it was possible to eliminate artefacts in HDEMG using Activity Index and ICA. However, it was difficult to accurately quantify the efficiency of this approach, as we did not know the ground truth about the artefacts' locations in time, nor the HDEMG channels where they were present. We could simulate certain kinds of artefacts at known times and in known HDEMG channels. However this would only account for certain types of artefacts. For example, we could induce an artefact by touching certain electrodes during recording at a predefined time, or during the whole recording by incorrectly applying the contact gel. But we would still be left with other artefacts that we have little control over. Moreover, not all the artefacts have a significant impact on the Activity Index and on MU identification. By observing the Activity Index and comparing it to the HDEMG channels, it is quite clear, that certain artefacts are more detrimental for the Activity Index than others. Therefore, without using the Activity Index or a similar metric, the assessment of artefact impact on EMG decomposition is often difficult.

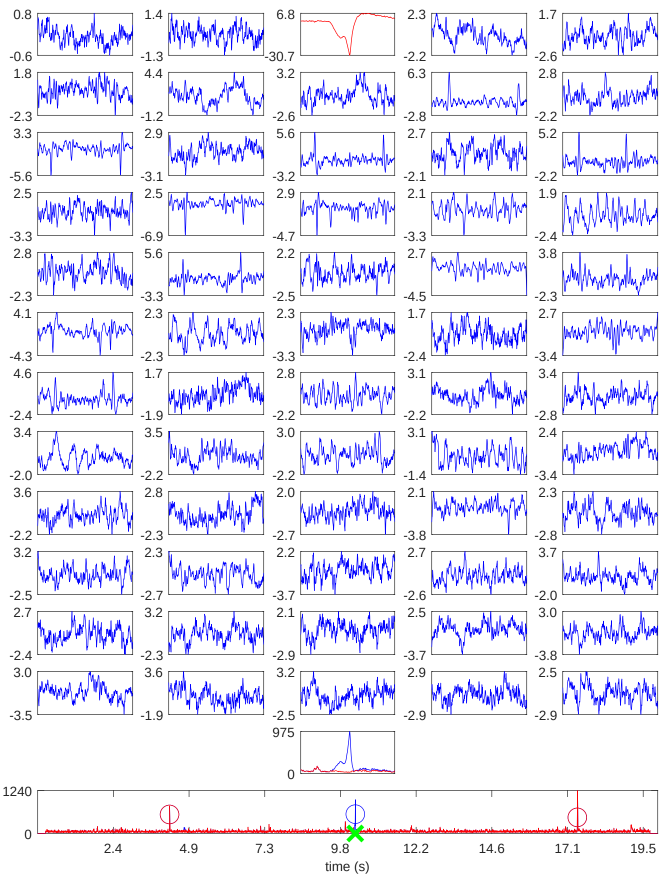


Figure 2: Artefact detection and elimination in ICA components of recorded HDEMG. The bottom panel depicts the I_A of all the ICA components (in blue) and the I_{EX} without the excluded component (in red). The green cross denotes the detected artefact time. The blue and red circles represent the detected outliers in I_A and I_{EX}, respectively. The second row from the bottom shows the 200 samples of Activity Index before and after the detected artefact time (green cross from the bottom panel). The blue line shows the I_A and the red line shows I_{EX}. The first dozen rows show the ICA components 200 samples before and after the detected artefact time (the green cross). The excluded component is depicted in red. It contains an artefact with an interest metric value of 0.96. All y-axis units are arbitrary.

In our method, determining the outliers in the Activity Index depend upon several parameters. The first one was the extension factor F, which was set to 1 for the purposes of this study. Also important was the scaled median absolute deviation (MAD) threshold, above which we considered an element of the Activity Index to be an outlier. In our case, we set it to 15. Lowering this threshold would yield more outliers in the Activty Index. Another parameter was the sliding window size in the Activity Index outlier detection, which in our case was set to 2048 samples.

The presented outlier location detection could also be refined, as using the currently described technique does not guarantee the identified outlier location to be at the location of the actual outlier peak. Instead, we aimed at identifying the first of the 200 samples that is at least 15 MAD away from the local median. This selection could have significant implications in the case of longer artefacts in HDEMG signals.

In our dataset, we found the mean interest metric $IntMet_{EX}$ value of the eliminated artefacts to be 0.85 $\pm$ 0.11, while the threshold, above which we considered the artefacts to be eliminated was set to 0.5. We also found slightly more outliers than actual artefacts as identified by visual inspection of HDEMG signals. This indicated that the current parameters (especially the extension factor for Activity Index calculation F of 1, the MAD threshold for outlier detection of 15 and the interest metric threshold of 0.5) were suitable to identify artefacts in the HDEMGs. However, further tests are required to confirm these findings in other muscles and contraction levels.

Acknowledgment

This study was supported by the Slovenian Research Agency (Project J2-1731 and Program funding P2-0041).

References

[1] Holobar, A., and Zazula, D. Correlation-based decomposition of surface electromyograms at low contraction forces. *Medical and Biological Engineering and Computing 42*, 4 (2004), 487–495.

[2] Holobar, A., and Zazula, D. Gradient convolution kernel compensation applied to surface electromyograms. In *International Conference on Independent Component Analysis and Signal Separation* (2007), Springer, pp. 617–624.

[3] Holobar, A., and Zazula, D. Multichannel blind source separation using convolution kernel compensation. *IEEE Transactions on Signal Processing 55*, 9 (2007), 4487–4496.

[4] Hyvärinen, A. Independent component analysis: recent advances. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 371*, 1984 (2013), 20110534.

[5] Hyvärinen, A., and Oja, E. Independent component analysis: algorithms and applications. *Neural networks 13*, 4-5 (2000), 411–430.

Zero-Knowledge Authentication

Jakob Povšič
University of Primorska,
Faculty of Mathematics, Natural Sciences
and Information Technologies,
Glagoljaška 8, 6000 Koper, Slovenia
jakob.povsic@gmail.com

Andrej Brodnik
University of Primorska,
Faculty of Mathematics, Natural Sciences
and Information Technologies,
Glagoljaška 8, 6000 Koper, Slovenia
andrej.brodnik@upr.si

Abstract

Zero-Knowledge proofs (ZKPs) enable proving of mathematical statements, revealing nothing but their validity. We design an authentication system with a ZKP as a password verification mechanism within the Extensible Authentication Protocol (EAP) framework. Designing a secure password authentication system requires us to adopt security practices for protecting ourselves against the vulnerabilities of passwords. Integrating said practices is not trivial because of the tight coupling with the password verification method.

Keywords extensible authentication protocol, zero-knowledge proofs, authentication, cryptography, key-stretching, passwords, quadratic residuosity problem

1 Introduction

Today privacy is a necessary sacrifice we have to make in order to take part in the digital world, imperative to our modern life. Every day, more digital systems gain access to our personal information. While this practice is often a necessary evil, many companies seek to exploit this position. Zero-knowledge proofs (ZKPs) are an intriguing cryptographic phenomenon for proving mathematical statements without revealing *why* they are true, and have the potential to change how our data exists in the digital space.

Our focus will be to define a simple use for zero-knowledge proofs. We will design an authentication system using a zero-knowledge proof as a password verification method, as an authentication method in the extensible authentication protocol (EAP). When designing the system, we need to protect ourselves from vulnerabilities of passwords. However, integrating security methods presents a challenge because of the zero-knowledge proof system.

2 Extensible Authentication Protocol

Extensible authentication protocol [10] (EAP) is a general purpose authentication framework designed for network access authentication. EAP defines a set of messages that support negotiation and execution of a variety of authentication protocols. EAP is a two-party protocol between a *peer* and an *authenticator* at each end of a link.

Messages. The peer and the authenticator communicate by exchanging *EAP messages*. The protocol starts with the authenticator sending a message to the peer. They keep exchanging messages until the authenticator can either authenticate the peer or not. Messages are exchanged in a lock-step manner, where an authenticator sends a message and the peer responds to it. The authenticator dictates the order of messages, meaning it can send a message at any point of communication, as opposed to the peer, which can only respond to messages from the authenticator.

Messages are composed of fields, each field length is multiple of an octet of bits (Table 1). We will store our authentication method data within the *Type-Data* field.

Our EAP method is identified by the *Type* 84.

3 Zero-Knowledge Proofs

Zero-Knowledge Proofs [5, 6, 7] (ZKPs) are a concept in cryptography for proving the validity of mathematical statements. What makes them particularly interesting is that ZKPs can prove a statement revealing no information about why a statement is true, hence the term *zero-knowledge*. In mathematics, theorem proofs are logical arguments that establish truth through inference rules of a deductive system based on axioms and other proven theorems. ZKPs are probabilistic, meaning they *convince* the verifier of the validity. We use the term convince, because ZKPs are not absolute truth, but the probability of someone being convinced by a false statement is arbitrarily small.

3.1 ZKP System for the Quadratic Residuosity Problem

Definition 3.1 (Quadratic Residuosity Problem) *Given an integer x, a semiprime modulus $n = pq$, where p and q are unknown different primes, and a Jacobi symbol value $\left(\frac{x}{n}\right) = 1$. Determine if x is a quadratic residue modulo n or not.*

The *law of quadratic reciprocity* enables efficient computation of the Jacobi symbol value $\left(\frac{x}{n}\right)$. However, when $\left(\frac{x}{n}\right) = 1$, it does not tell if x is a quadratic residue modulo n or not. x is only a quadratic residue if it's a quadratic residue of both modulo p and q ($\left(\frac{x}{p}\right) = \left(\frac{x}{q}\right) = 1$). To compute this, we would have to know the factorization of n. However, since n is a product of two primes $pq = n$, this is computationally hard [2]. The only efficient way

DOI https://doi.org/10.18690/978-961-286-516-0.2
ISBN 978-961-286-516-0

Table 1: EAP Message Format

Length (Octets)	1	1	2	1	$n \leq 2^{16}$
Field	Code	Identifier	Length	Type	Type-Data

to prove x is a quadratic residue modulo n, is with the root w. The problem acts as a *trapdoor* function, where it's hard to prove if x is a quadratic residue modulo n solely from x and n, while it is easy to prove when you know its root w.

Authors [7] described a ZKP system for the *quadratic residuosity problem.* To prove x is a quadratic residue modulo n in zero-knowledge we need to prove the existence of the root w, where $w^2 \equiv x \pmod{n}$, without revealing w to the verifier.

n	Semiprime, where Jacobi $\left(\frac{x}{n}\right) = 1$
x	Residue, where $w^2 \equiv x \pmod{n}$
w	Root

	Peer		Authenticator
1	$u \quad {}_R\, \mathbb{Z}_n^*$		
	$y = u^2 \pmod{n}$	$\xrightarrow{y}$	
2		$\xleftarrow{b}$	$b \quad {}_R\, \{0,1\}$
3	$z = uw^b \pmod{n}$	$\xrightarrow{z}$	verify $z^2 \equiv yx^b \pmod{n}$

Table 2: ZKP Authentication with EAP

In Table 2 of the ZKP authentication process, the middle spaces represent the EAP message *Type-Data* field.

The prover begins by picking a random integer u from the field $\mathbb{Z}_n$, computing $y = u^2 \pmod{n}$, and sending y to the verifier. The verifier picks a random bit b and sends it to the prover. The prover computes the value z with b and sends it back. The verifier checks the proof by asserting $z^2 \equiv yx^b \pmod{n}$, this is possible since

$$\begin{aligned} z^2 &\equiv yx^b \pmod{n} \\ (uw^b)^2 &\equiv u^2(w^2)^b \pmod{n} \\ u^2w^{2b} &\equiv u^2w^{2b} \pmod{n}. \end{aligned}$$

For each round a cheating prover has a $\frac{1}{2}$ probability of succeeding by correctly guessing the value of the random bit b. To improve the strength of the proof, we repeat this process m times for a confidence of $1 - 2^{-m}$.

To use this protocol as a password verification method, we can treat the root w as the password $p = w$ known by the peer. The ZKP protocol proves that x is a quadratic residue modulo n, by proving the knowledge of the root w, where $w^2 \equiv x \pmod{n}$. The peer will prove that x is a quadratic residue modulo n, to do this however, the peer needs to prove the knowledge of the password $p = w$. With this, the authenticator can assert that the password is valid.

4 Password Protection

Password cracking [1] is an *offline attack* [8], where an attacker extracts passwords from data used by the authentication system for password verification. Protecting passwords on the data layer is of critical importance. *Key-stretching*, [9, 1] also called *password hashing*, is the industry standard method for improving security of low entropy secrets like passwords.

The quadratic residue x is derived from root $w = p$ and persistently stored with the authenticator. This introduces a vulnerability, as an attacker with access to x could crack the password w in an offline attack. To provide adequate security, we need to use key-stretching in our authentication method. A common application of a key-stretching method is to transform the vulnerable data stored in the authentication system. However, this approach doesn't work in our case. Let us revisit how the authenticator verifies the proof, and why key-stretching the password verification data (x) data is an issue. We'll begin by assuming the system can verify the proof and key-stretching the password verification data (x) data. As we define our process, we will see why it is not possible.

Key-Stretching x. On the last step of the protocol the authenticator verifies that

$$z^2 \equiv yx^b \pmod{n}.$$

If we stretch x with a function H and a salt s

$$H(x, s) = x_H,$$

we can then verify the proof with an inverse function H^{-1}

$$z^2 \equiv yH^{-1}(x_H, s)^b.$$

This is possible assuming a polynomial algorithm H^{-1} exists, however, since key-stretching methods are based on hashing functions (one-way functions), we know that the probability of a polynomial algorithm H^{-1} to successfully compute a *pseudo-inverse* is negligibly small. For all positive integers c [4]

$$\Pr[H(H^{-1}(H(x))) = H(x)] < |x|^{-c}.$$

Even if given unbounded time and resources, the *pseudo-inverse* $x' = H^{-1}(H(x))$ might not be equal to $x' \neq x$. The set $x, x' \in I_x$ are all values that map into $H(x) = H(x')$, and since H is not injective we know that $|I_x| \geq 1$. Meaning that the probability that $x' = x$ is

$$\Pr[H^{-1}(H(x)) = x] = \frac{1}{|I_x|}.$$

Key-stretching x prevents us from verifying the ZKP. However, by increasing the entropy of the root w, we can eliminate the vulnerability and ensure adequate security. Our new approach won't treat the password $p \neq w$ as the root w. However, we will use the password p to derive the root $w = H(p, s)$, using a key-stretching function H and salt s. This way we've ensured the same level of protection against offline attacks as if we stretched the data stored in the system. And because we didn't transform x, we can verify the proof without being affected by issues mentioned in the previous paragraph. A similar approach is used in the PPP EAP SRP-SHA1 protocol [3]. Earlier we argued the ZKP works as a password verification method because $p = w$, this argument isn't true anymore. However, even though $w \neq p$, the peer can only derive w knowing the password p, so when the peer proves the knowledge of w, it can only be so because they know p as well.

5 Secure Authentication

The authentication process now begins with the peer sending his identifier to the authenticator and the authenticator responding with the peer's unique salt s and modulo n. The peer can now derive the root w from the password p and salt s. This part of the process (Steps 1. and 2. in the Table 3) happens only once.

The peer can then authenticate by following the process as described in §3.1. This part (Steps 3., 4. and 5.) of the process is repeated m times for a confidence of $1 - 2^{-m}$.

The middle space in the Table 3 represents the *Type-Data* field of the EAP messages.

	Peer		Authenticator
1		$\xrightarrow{I}$	
2	$w = H(p, s)$	$\xleftarrow{s,n}$	
3	$u \quad {}_R\,\mathbb{Z}_n$		
	$y = u^2 \pmod{n}$	$\xrightarrow{y}$	
4		$\xleftarrow{b}$	$b \quad {}_R\,\{0, 1\}$
5	$z = uw^b \pmod{n}$	$\xrightarrow{z}$	verify $z^2 \equiv yx^b \pmod{n}$

Table 3: Improved ZKP Authentication with EAP

Let us examine the EAP messages (Figure 1) of the authentication process described in Table 3. The mapping between EAP messages and the steps in Table 3 is *not one-to-one*. We merged some steps to reduce the number of message exchanges required for the process to complete.

Identity This message is used to query the identity of the peer. In a system with multiple peers, this is required to identify the peer authenticating, and to find the correct salt s and quadratic residue x. (Table 3, Step 1.)

Setup The peer needs both the salt s and the modulus n to compute the proof, however, he only knows the password p. Once the peer identifies himself, the authenticator needs to send him the salt s and modulus n in the *setup request* message. (Table 3, Step 2. and 3.)

Verification With this message pair the peer and the authenticator exchange data to compute and verify the proof. The authenticator sends the random bit b and the peer responds with the proof z. The peer also sends the y_{i+1} for the next verification round $i + 1$, this is done as an optimisation to improve the speed of the process. (Table 3, Step 4., 5. and 3.)

Success/Failure After each *verification* message, the authenticator verifies the proof, and once it's done successfully for m rounds, the authenticator sends the *success* message. However, if the proof isn't valid, the authenticator must send a *failure* message.

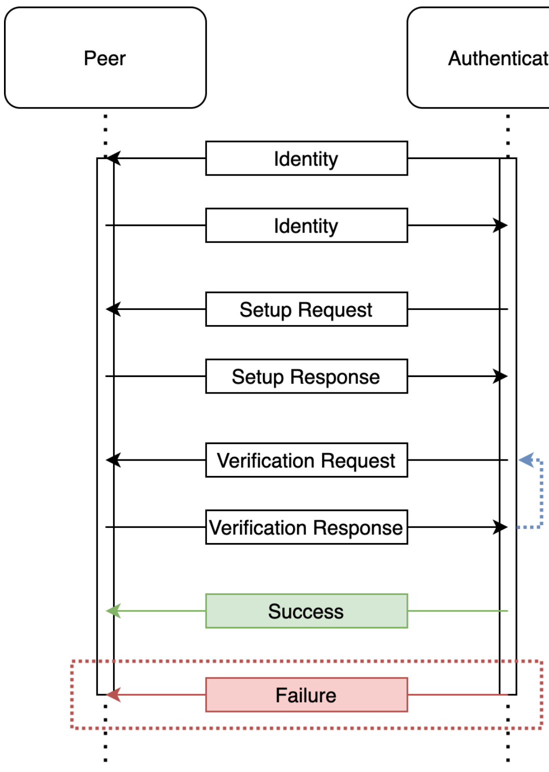


Figure 1: EAP Method Execution

6 Conclusions and Future Work

The aim of this work was to study the utility of zero-knowledge proofs as an EAP authentication method. We've presented an EAP method using a ZKP system for password verification. Additionally, we ensured adequate password protection by using a key-stretching method.

We have been successful in our goal of studying and using the ZKP protocol. While theoretically interesting the system's performance may not appropriate for real-world applications. The iterative nature of the underlying ZKP protocol accumulates communication latencies, slowing down the system.

Future work.

- The EAP method presented in this work can be implemented and tested in a real-world environment.

- The ZKP protocol used in this work is a first generation protocol. Today there are many newer protocols that have solved many shortcomings of the older generation ZKPs. Using a newer generation ZKP protocol can improve the performance of the authentication system.
- The ZKP protocol we've examined is iterative, which can cause worse performance. A parallel ZKP construction is assumed to have a weaker strength of zero-knowledge. However, in a real-world application, the performance improvements might justify the theoretical shortcomings.

References

[1] Blocki, J., Harsha, B., and Zhou, S. On the economics of offline password cracking. In *2018 IEEE Symposium on Security and Privacy (SP)* (2018), IEEE, pp. 853–871.

[2] Buchmann, J. A. *Factoring.* Springer US, New York, NY, 2001, pp. 171–183.

[3] Carlson, J. D., Aboba, D. B. D., and Haverinen, H. PPP EAP SRP-SHA1 Authentication Protocol. Internet-Draft draft-ietf-pppext-eap-srp-03, Internet Engineering Task Force, July 2001. Work in Progress.

[4] Goldreich, O. *Foundations of cryptography: Volume 1, basic tools.* Cambridge university press, 2007.

[5] Goldreich, O., and Krawczyk, H. On the composition of zero-knowledge proof systems. *SIAM Journal on Computing 25*, 1 (1996), 169–192.

[6] Goldreich, O., Micali, S., and Wigderson, A. How to Prove all NP-Statements in Zero-Knowledge, and a Methodology of Cryptographic Protocol Design. vol. 263, pp. 171–185.

[7] Goldwasser, S., Micali, S., and Rackoff, C. The knowledge complexity of interactive proof systems. *SIAM Journal on computing 18*, 1 (1989), 186–208.

[8] Grassi, P. A., Garcia, M. E., and Fenton, J. L. NIST Special Publication 800-63-3 Digital Identity Guidelines. *National Institute of Standards and Technology, Los Altos, CA* (2017).

[9] Hornby, T. Salted password hashing-doing it right, 2016.

[10] Vollbrecht, J., Carlson, J. D., Blunk, L., Aboba, D. B. D., and Levkowetz, H. Extensible Authentication Protocol (EAP). RFC 3748, June 2004.

Graphs where Search Methods are Indistinguishable

Matjaž Krnc and Nevena Pivač[*]
University of Primorska,
Faculty of Mathematics, Natural Sciences and Information Technologies,
Glagoljaška 8, 6000 Koper, Slovenia
`matjaz.krnc@upr.si nevena.pivac@iam.upr.si`

Abstract

Graph searching is one of the simplest and most widely used tools in graph algorithms. Every graph search method is defined using some particular selection rule, and the analysis of the corresponding vertex orderings can aid greatly in devising algorithms, writing proofs of correctness, or recognition of various graph families.

We study graphs where the sets of vertex orderings produced by two different search methods coincide. We characterise such graph families for ten pairs from the best-known set of graph searches: Breadth First Search (BFS), Depth First Search (DFS), Lexicographic Breadth First Search (LexBFS) and Lexicographic Depth First Search (LexDFS), and Maximal Neighborhood Search (MNS).

Keywords graph search methods, breadth first search, depth first search

1 Introduction

Graph search methods (for instance, Depth First Search and Breadth First Search) are among essential concepts classically taught at the undergraduate level of computer science faculties worldwide. Various types of graph searches have been studied since the 19th century, and used to solve diverse problems, from solving mazes, to linear-time recognition of interval graphs, finding minimal path-cover of co-comparability graphs, finding perfect elimination order, or optimal coloring of a chordal graph, and many others [1, 4, 7, 8, 10, 11].

In its most general form, a *graph search* (also *generic search* [5]) is a method of traversing vertices of a given graph such that every prefix of the obtained vertex ordering induces a connected graph. This general definition of a graph search leaves much freedom for a selection rule determining which node is chosen next. By defining some specific rule that restricts this choice, various different graph search methods are defined. Other search methods that we focus on in this paper are Breadth First Search, Depth First Search, Lexicographic Breadth First Search, Lexicographic Depth First Search, and Maximal Neighborhood Search.

We briefly present the studied graph search methods in Section 2, and then state the obtained results in Section 3. Due to lack of space we omit the proofs and provide some directions for further work in Section 4. All proofs are available in the full version of paper on www.arxiv.org.

[*]This work is supported in part by the Slovenian Research Agency (research programs P1-0285 and P1-0383, research projects N1-0102, J1-9110, N1-0160, N1-0209 and Young Researchers Grant).

2 Preliminaries

We now briefly describe the above-mentioned graph search methods, and give the formal definitions. Note that the definitions below are not given in a historically standard form, but rather as so-called *three-point conditions*, due to Corneil and Kruger [5] and also Brändstadt et. al. [3]. Two vertices $u, v \in V(G)$ satisfy the relation $u <_\sigma v$ if u appears before v in the ordering $\sigma : V(G) \to \{1, 2, \dots, n\}$ of vertices in G.

Breadth First Search (BFS), first introduced in 1959 by Moore [9], is a restriction of a generic search which puts unvisited vertices in a queue and visits a first vertex from the queue in the next iteration. After visiting a particular vertex, all its unvisited neighbors are put at the end of the queue, in an arbitrary order.

Definition 2.0.1 *An ordering σ of V is a BFS-ordering if and only if the following holds: if $a <_\sigma b <_\sigma c$ and $ac \in E$ and $ab \notin E$, then there exists a vertex d such that $d <_\sigma a$ and $db \in E$.*

Any BFS ordering of a graph G starting in a vertex v results in a rooted tree (with root v), which contains the shortest paths from v to any other vertex in G (see [6]). We use this property implicitly throughout the paper.

Depth First Search (DFS), in contrast with the BFS, traverses the graph as deeply as possible, visiting a neighbor of the last visited vertex whenever it is possible, and backtracking only when all the neighbors of the last visited vertex are already visited. In DFS, the unvisited vertices are put on top of a stack, so visiting a first vertex in a stack means that we always visit a neighbor of the most recently visited vertex.

Definition 2.0.2 *An ordering σ of V is a DFS-ordering if and only if the following holds: if $a <_\sigma b <_\sigma c$ and $ac \in E$ and $ab \notin E$, then there exists a vertex d such that $a <_\sigma d <_\sigma b$ and $db \in E$.*

Lexicographic Breadth First Search (LexBFS) was introduced in the 1970s by Rose, Tarjan and Lueker [10] as a part of an algorithm for recognizing chordal graphs in linear time. Since then, it has been used in many graph algorithms mainly for the recognition of various graph classes.

DOI https://doi.org/10.18690/978-961-286-516-0.3
ISBN 978-961-286-516-0

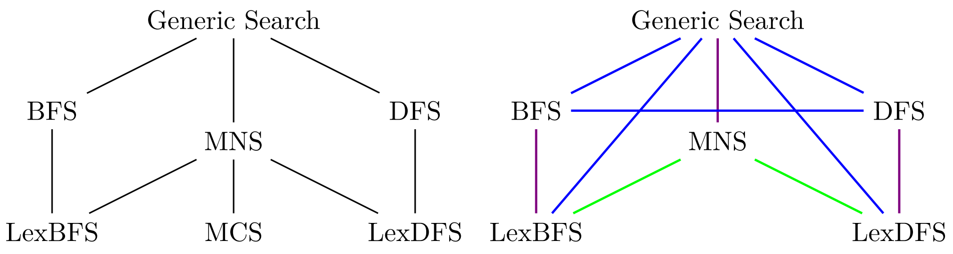


Figure 1: On the left: Hasse diagram showing how graph searches are refinements of one another. On the right is a summary of our results: Green pairs are equivalent on $\{P_4, C_4\}$-free graphs. Violet pairs are equivalent on {pan, diamond}-free graphs. Blue pairs are equivalent on {paw, diamond, P_4, C_4}-free graphs.

Definition 2.0.3 *An ordering σ of V is a LexBFS ordering if and only if the following holds: if $a <_\sigma b <_\sigma c$ and $ac \in E$ and $ab \notin E$, then there exists a vertex d such that $d <_\sigma a$ and $db \in E$ and $dc \notin E$.*

LexBFS is a restricted version of Breadth First Search, where the usual queue of vertices is replaced by a queue of unordered subsets of the vertices which is sometimes refined, but never reordered.

Lexicographic Depth First Search (LexDFS) was introduced in 2008 by Corneil and Krueger [5] and represents a special instance of a Depth First Search.

Definition 2.0.4 *An ordering σ of V is a LexDFS ordering if and only if the following holds: if $a <_\sigma b <_\sigma c$ and $ac \in E$ and $ab \notin E$, then there exists a vertex d such that $a <_\sigma d <_\sigma b$ and $db \in E$ and $dc \notin E$.*

Maximal Neighborhood Search (MNS), introduced in 2008 by Corneil and Krueger [5], is a common generalization of LexBFS, LexDFS, and MCS, and also of Maximal Label Search (see [2] for definition).

Definition 2.0.5 *An ordering σ of V is an MNS ordering if and only if the following statement holds: If $a <_\sigma b <_\sigma c$ and $ac \in E$ and $ab \notin E$, then there exists a vertex d with $d <_\sigma b$ and $db \in E$ and $dc \notin E$.*

The MNS algorithm uses the set of integers as the label, and at every step of iteration chooses the vertex with maximal label under set inclusion.

Corneil [5] exposed an interesting structural aspect of graph searches: the particular search methods can be seen as restrictions, or special instances of some more general search methods. For six well-known graph search methods they present a depiction, similar to the one in Figure 1, showing how those methods are related under the set inclusion. For example, every LexBFS ordering is at the same time an instance of BFS and MNS ordering of the same graph. Similarly, every LexDFS ordering is at the same time also an instance of MNS, and of DFS (see Figure 1). The reverse, however, is not true, and there exist orderings that are BFS and MNS, but not LexBFS, or that are DFS and MNS but not LexDFS.

3 Problem description and results

Since the connections in Figure 1 represent relations of inclusion, it is natural to ask under which conditions on a graph G the particular inclusion holds also in the converse direction. More formally, we say that two search methods are *equivalent on a graph G* if the sets of vertex orderings produced by both of them are the same. We say that two graph search methods are *equivalent on a graph class $\mathcal{G}$* if they are equivalent on every member $G \in \mathcal{G}$. Perhaps surprisingly, three different graph families suffice to describe graph classes equivalent for the ten pairs of graph search methods that we consider. Those are described in Theorems 3.1 to 3.3 below, but first we need a few more definitions.

All the graphs considered in the paper are finite and connected. A *k-pan* is a graph consisting of a k-cycle, with a pendant vertex added to it. We say that a graph is *pan-free* if it does not contain a pan of any size as an induced subgraph. A 3-pan is also known as a *paw graph.*

Theorem 3.1 *Let G be a connected graph. Then the following is equivalent:*

A1. Graph G is $\{P_4, C_4$, paw, diamond$\}$-free.
A2. Every graph search of G is a DFS ordering of G.
A3. Every graph search of G is a BFS ordering of G.
A4. Any vertex-order of G is a BFS, if and only if it is a DFS.

Theorem 3.2 *Let G be a connected graph. Then the following is equivalent:*

B1. Graph G is {pan, diamond}-free.
B2. Every DFS ordering of G is a LexDFS ordering of G.
B3. Every BFS ordering of G is a LexBFS ordering of G.
B4. Every graph search of G is an MNS ordering of G.

Theorem 3.3 *Let G be a connected graph. Then the following is equivalent:*

C1. Graph G is $\{P_4, C_4\}$-free.
C2. Every MNS ordering of G is a LexDFS ordering of G.
C3. Every MNS ordering of G is a LexBFS ordering of G.

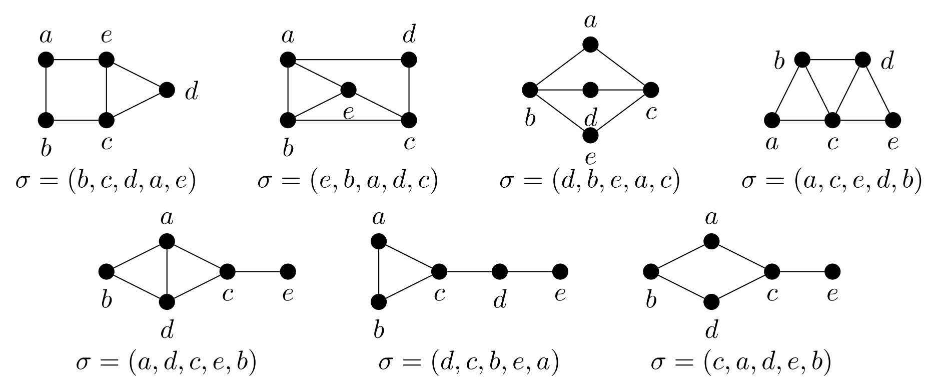

$\sigma = (b, c, d, a, e)$
$\sigma = (e, b, a, d, c)$
$\sigma = (d, b, e, a, c)$
$\sigma = (a, c, e, d, b)$
$\sigma = (a, d, c, e, b)$
$\sigma = (d, c, b, e, a)$
$\sigma = (c, a, d, e, b)$

Figure 2: Graphs and corresponding orderings that are MNS and not MCS orderings.

From Theorems 3.1 and 3.2 we can immediately derive similar results for two additional pairs of graph search methods.

Corollary 3.3.1 *Let G be a connected graph. Then the following is equivalent:*

A1. Graph G is $\{P_4, C_4,$ paw, diamond$\}$-free.
A5. Every graph search of G is a LDFS ordering of G.
A6. Every graph search of G is a LBFS ordering of G.

4 Conclusion and further work

In this paper we consider the major graph search methods and study the graphs in which vertex-orders of one type coincide with vertex-orders of some other type. Interestingly, three different graph families suffice to describe graph classes equivalent for the ten pairs of graph search methods that we consider, which provides an additional aspect of similarities between the studied search methods.

Among the natural graph search methods not yet considered in this setting would be the *Maximum Cardinality Search* (MCS), introduced in 1984 (for definition see Tarjan and Yannakakis [12]). As shown on Figure 1, every MCS is a special case of an MNS vertex-order. While it is easy to verify that $\{P_4, C_4, \text{paw}, \text{diamond}\}$-free graphs do not distinguish between MNS and MCS vertex orders, Figure 2 provides examples of graphs which admit MNS, but not MNS vertex orders. Characterising graphs equivalent for MNS and MCS remains an open question.

Acknowledgements

The authors would like to thank prof. Martin Milanič for the initial suggestion of the problem, and to Ekki Köhler and his reseach group, for introducing the diverse world of graph searches to us.

References

[1] Beisegel, J. Characterising AT-free graphs with BFS. In *Graph-Theoretic Concepts in Computer Science* (2018), A. Brandstädt, E. Köhler, and K. Meer, Eds., pp. 15–26.

[2] Berry, A., Krueger, R., and Simonet, G. Maximal label search algorithms to compute perfect and minimal elimination orderings. *SIAM Journal on Discrete Mathematics 23*, 1 (2009), 428–446.

[3] Brandstädt, A., Dragan, F. F., and Nicolai, F. LexBFS-orderings and powers of chordal graphs. *Discrete Math. 171*, 1-3 (1997), 27–42.

[4] Corneil, D. G., Dusart, J., Habib, M., and Kohler, E. On the power of graph searching for cocomparability graphs. *SIAM Journal on Discrete Mathematics 30*, 1 (2016), 569–591.

[5] Corneil, D. G., and Krueger, R. M. A unified view of graph searching. *SIAM Journal on Discrete Mathematics 22*, 4 (2008), 1259–1276.

[6] Even, S. *Graph algorithms.* Cambridge University Press, 2011.

[7] Golumbic, M. *Algorithmic Graph Theory and Perfect Graphs.* Annals of Discrete Mathematics, Volume 57. Elsevier, 2004, pp. 98–99.

[8] Köhler, E., and Mouatadid, L. Linear time lexdfs on cocomparability graphs. In *Scandinavian Workshop on Algorithm Theory* (2014), Springer, pp. 319–330.

[9] Moore, E. F. The shortest path through a maze. In *Proc. Int. Symp. Switching Theory, 1959* (1959), pp. 285–292.

[10] Rose, D. J., Lueker, G. S., and Tarjan, R. E. Algorithmic aspects of vertex elimination on graphs. *SIAM Journal on Computing 5*, 2 (1976), 266–283.

[11] Tarjan, R. E. Depth-first search and linear graph algorithms. *SIAM journal on computing 1*, 2 (1972), 146–160.

[12] Tarjan, R. E., and Yannakakis, M. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM Journal on computing 13*, 3 (1984), 566–579.

System for Remote Collaborative Embedded Development

Martin Domajnko
University of Maribor,
Faculty of Electrical
Engineering and Computer Science,
Koroška cesta 46, 2000 Maribor, Slovenia
martin.domajnko@student.um.si

Nikola Glavina
University of Maribor,
Faculty of Electrical
Engineering and Computer Science,
Koroška cesta 46, 2000 Maribor, Slovenia
nikola.glavina@student.um.si

Aljaž Žel
University of Maribor,
Faculty of Electrical
Engineering and Computer Science,
Koroška cesta 46, 2000 Maribor, Slovenia
aljaz.zel@student.um.si

Abstract

This paper explores the challenges and devised solutions for embedded development which arose during the COVID-19 pandemic. While software development, nowadays with modern tools and services such as git, virtual machines and communication suits, is relatively unaffected by resource location. That is not the case for firmware and embedded systems, which relies on physical hardware for design, development, and testing. To overcome the limitations of remote work and obstructed access to actual hardware, two ideas were implemented and tested. First, based on integrated circuit emulation using QEMU to emulate an ARM core and custom software to facilitate communication with the embedded system. Second, remote programming and debugging over the internet with a dedicated computer system acting as a middle man between a development environment and physical hardware using OpenOCD debugger.

Keywords embedded development, remote development, OpenOCD, QEMU, ARM, ARM semihosting

1 Introduction

During the design, development, and testing phases of our Passive Floating Probe project [7], we heavily relied on physical access to the hardware. The restrictions that were put in place during the COVID-19 pandemic didn't stop the development process of our project, but they heavily limited it. Lack of working hardware for every member of the team was our major issue, so we tried to think up solutions to overcome that problem. The first idea, which would allow our work on the project to continue, was based on the QEMU [8] machine emulator. We emulated our integrated circuit with an ARM core processor and added a custom software layer, which served as an emulation of the real-world communication pipeline with the system. The solution proved useful, albeit with the restriction that it allowed only local development, prompting us to develop our second idea. Remote programming and debugging of our hardware over the internet was the main part of that solution. This was achieved with a dedicated computer system acting as a middle man between our development environment and the targeted hardware, enabling programming and debugging.

Finally, we present the structure of this paper. In the second section, we are going to explain in detail the implementations of both solutions. Following in section three, where we are going to talk about the limitations of the implemented solutions and the lessons we learned. The paper will conclude in section four with possible improvements.

2 Implementation of remote embedded development

For the work on the project to continue under COVID-19 restrictions, a solution had to be devised to solve the problems that came with remote work on a project dependent on specific hardware. This included hardware development, maintenance, and the ability to develop and test software on the hardware remotely. During the academic year, two solutions were devised and implemented.

2.1 Solution using emulator

The first solution was based on microcontroller emulation, as seen in figure 1, using a specialized version of QEMU emulation software called xPack [14] version 2.8.0-9. This enabled us to emulate a variant of STM32

*Listed in alphabetical order

DOI https://doi.org/10.18690/978-961-286-516-0.4
ISBN 978-961-286-516-0

family of microcontrollers. The variant chosen was STM32F407-Discovery development board, since it was closest to our target hardware, and we had access to a matching development board on which to test the differences. Using the emulator method proved to be a great benefit, since not all members had access to real hardware at that time, but everyone could set up the emulator on their computer. Since we didn't have previous experience working with STM family microcontrollers, the emulator also allowed us to focus on platform-specific software issues without the complexity of hardware issues in novice programmers. However, this also came with a cost. First one, the emulator didn't work out of the box. Second one, no sensors can be connected to the board directly since interfaces in the emulator are virtual.

Before the emulator could be used with the desired family, a minor bug inside the emulator source code, that was causing ROM memory overlap, needed to be patched manually. After initial setup, we established a basic form of IO system using a script that allowed us to run the emulator and automatically redirect the standard output using UNIX pipes to a file where we could monitor the output. The emulator approach enabled us to test out the version of real time operating system FreeRTOS [2] for STM32 microcontrollers. Some modification to compiler flags were required, especially the soft floating-point unit because the emulator was unable to emulate real FPU. While unable to connect sensors to the virtual environment, we were mostly focused on creating task manager, which took care of the correct operation and communication between individual tasks or sensors. Tasks, that were supposed to be related to the sensors, were temporarily replaced with empty functions that returning predefined values for testing.

To enable outside communication with our embedded software, semihosting feature of ARM architecture was used [1]. A custom communication service was implemented in which standard input and output were redirected to netcat running locally, creating a network interface. After successful compilation of the program, the emulator could be run with the compiled elf binary and redirect semihosting IO to stdio. An example in our case: *< /dev/null nc -q -1 -l 5000 | qemu-system-gnuarmeclipse –verbose –verbose –board STM32F4-Discovery –mcu STM32F407VG -d unimp,guest_errors –image STM32F407-Discovery-blinky.elf –semihosting-config enable=on,target=native | nc -l 6000 > /dev/null.* This allowed us to emulate a serial connection to and from the emulated microcontroller over a network connection.

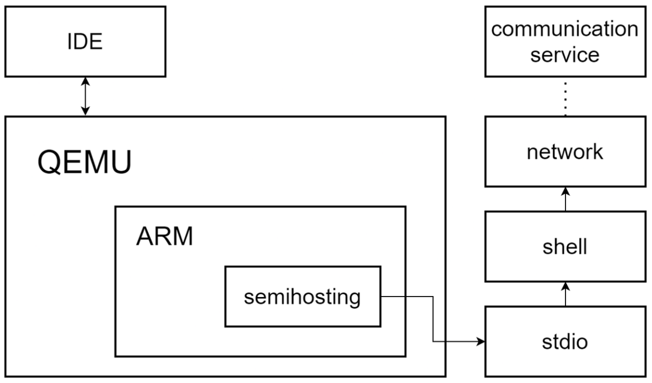


Figure 1: Schematic of the development system using emulation.

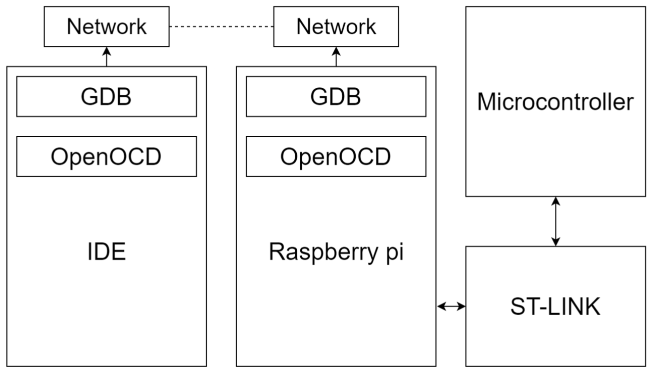


Figure 2: Schematic of the remote development system.

2.2 Solution using remote development

To amend the shortcomings of the development on the emulator, a remote development solution was devised and implemented. The solution, as seen on figure 2, is based on a dedicated computer system leveraging remote connection functionality of GDB or "GNU Project Debugger" [4] for remote programming and real-time debugging.

The system was built using Raspberry Pi 3 Model B [9], running ARM version of Ubuntu version 20.04 LTS [13]. The selected single board computer handled network communications, attached peripherals and services needed to facilitate remote programming and real-time debugging. For embedded development, an ST-LINK in-circuit debugger and programmer [11], in our case STM32F407G-DISC1 development board [10] providing ST-LINK/V2-A, was connected via USB connection to the Raspberry Pi. To connect GDB debugging functionality with the ST-LINK programming functionality with the target integrated circuit, the OpenOCD software [6] version 0.10.0 was used.

To prepare the setup, two configuration files in OpenOCD format needed to be created. First one defining "hla_serial" value, describing the serial number of the connected ST-LINK device. The second one defining "-event gdb-detach" behavior as "resume". Defining this permitted the embedded program to run even after the debug session disconnected, allowing to test the functionality over longer periods of time without constant connection to the host machine.

With the prepared configuration files, a script was created to start the OpenOCD session with the correct parameters for the ST-LINK device, target device, network settings and created configuration files. An example in our case: *openocd -c "bindto $HOSTNAME"*

```
-c "gdb_port 3333" -c "tcl_port disabled" -c "telnet_port disabled" -f /usr/share/openocd/scripts/interface/stlink-v2.cfg -c "adapter_khz 480" -c "transport select hla_swd" -f /usr/share/openocd/scripts/target/stm32l4x.cfg -f ./gdb_resume.cfg -f ./serial.cfg » log.txt
```

The script was started inside a tmux [12] instance. This allowed for the OpenOCD session to run without an active user connection to the shell executing the script, or alternately for multiple users to be connected to the same shell instance to observe debug in print messages.

Once the system was set up, multiple ST-LINK devices could be connected and used simultaneously by adding additional configuration files with serial numbers and starting OpenOCD sessions on different network ports.

3 Usage and lessons learned

The development process was arranged in the form of the required hardware development and maintenance to be in the domain of our mentor and be kept at the university. Team members could make request for hardware modifications and then develop and debug the project software remotely. While concurrent remote software collaboration was achieved through distributed revision control system Gitea [3] as source code repository and management tool.

3.1 Emulator

The emulator provided a good start into getting acquainted with embedded development. This allowed us to learn and develop embedded software without physical hardware. The downside was the inability to work with real sensors, which later made us switch to a remote system. Another problem was that the emulator was not capable of running functions that required precise timing, such as real-time code or interrupt execution.

3.2 Remote

The advantage of remote system made it possible for us to connect to the targeted hardware from anywhere as if it was accessible locally. This included real-time debugging with the ability to see microcontroller processor states and memory values. The problem with this solution was the restriction of a single connection to the OpenOCD instance, which limited the work on the hardware to a single developer at a time. This was resolved with communication and access scheduling.

4 Conclusions

The system was sufficient to allow our work on the project to continue. During development, we were fortunate enough to not have major problems with security. The system, as it was used, would allow anyone to access the system if they identified the used network ports and protocols. This was not a major concern, as it only allowed to program our particular microcontroller with a dedicated firmware. As such, this problem was not addressed during the production. Possible additional security was tested, with the implementation of username and password authentication using NGINX reverse proxy server [5] and httpd access restrictions on the system URL.

Acknowledgment

The authors acknowledge the financial support from the Institute of Computer Science of the Faculty of Electrical Engineering and Computer Science and would like to thank mag. Jernej Kranjec for his guidance and assistance. The authors would also like to acknowledge the remaining members of the project group, namely Tilen Koren, Anna Sidorova and Viktorija Stevanoska for their work on the project.

References

[1] ARM. Arm target input/output facilities. https://developer.arm.com/documentation/dui0471/g/Bgbjjgij, 2021. Accessed: 2021-07-30.

[2] FreeRTOS. Real time operating system for microcontrollers. https://www.freertos.org/, 2021. Accessed: 2021-07-30.

[3] Gitea. Lightweight code hosting solution. https://gitea.io, 2021. Accessed: 2021-07-30.

[4] GNU. The gnu project debugger. https://www.gnu.org/software/gdb/, 2021. Accessed: 2021-07-30.

[5] NGINX. Reverse proxy. https://www.nginx.com/, 2021. Accessed: 2021-07-30.

[6] OpenOCD. Open on-chip debugger. http://openocd.org/, 2021. Accessed: 2021-07-30.

[7] Perrone, M., Knupleš, U., Žalik, M., Keršič, V., and Šinko, T. Pasive floating probe. In *StuCoSReC: Proceedings of the 2019 6th Student Computer Science Research Conference* (2019), pp. 13–17.

[8] QEMU. the fast! processor emulator. https://www.qemu.org/, 2020. Accessed: 2020-04-30.

[9] Raspberry Pi. Raspberry pi 3 model b. https://www.raspberrypi.org/products/raspberry-pi-3-model-b/, 2021. Accessed: 2021-07-30.

[10] STM. Discovery kit with stm32f407vg mcu. https://www.st.com/en/evaluation-tools/stm32f4discovery.html, 2021. Accessed: 2021-07-30.

[11] STM. Stm st-link/v2 in-circuit debugger/programmer. https://www.st.com/en/development-tools/st-link-v2.html, 2021. Accessed: 2021-07-30.

[12] tmux. Terminal multiplexer. https://github.com/tmux/tmux/wiki, 2021. Accessed: 2021-07-30.

[13] Ubuntu. Ubuntu os. https://ubuntu.com/download/raspberry-pi, 2021. Accessed: 2021-07-30.

[14] xPack. The xpack qemu arm. https://xpack.github.io/qemu-arm/, 2020. Accessed: 2020-04-30.

Leaf Segmentation of Rosette Plants using Rough K-Means in CIELab Color Space

Arunita Das [*]
Dept. of Computer Science and Application
Midnapore College (Autonomous)
Paschim Medinipur, West Bengal, India
arunita17@gmail.com

Daipayan Ghosal
Dept. of Computer Science and Application
Midnapore College (Autonomous)
Paschim Medinipur, West Bengal, India
daipayan.ghosal@gmail.com

Krishna Gopal Dhal [†]
Dept. of Computer Science and Application
Midnapore College (Autonomous)
Paschim Medinipur, West Bengal, India
krishnagopal.dhal@midnaporecollege.ac.in

Abstract

Segmentation of Plant Images plays an important role in modern agriculture where it can provide accurate analysis of a plant's growth and possible anomalies. In this paper, rough set based partitional clustering technique called Rough K-Means has been utilized in CIELab color space for the proper leaf segmentation of rosette plants. The efficacy of the proposed technique have been analysed by comparing it with the results of traditional K-Means and Fuzzy C-Means clustering algorithms. The visual and numerical results reveal that the RKM in CIELab provides the nearest result to the ideal ground truth, hence the most efficient one.

Keywords image segmentation, color space, rough set, partitional clustering

1 Introduction

Leaf check and leaf region (i.e., plant area) are the key plant phenotyping qualities used to examine the plant development and advancement [19] [25] [28], blossoming time [13], and yield potential. The leaf include can be tended to in different manners from the AI viewpoint [1]. One such route is to check the number of leaves from fragmented plant area. Several image segmentation techniques are reported in literature for leaf segmentation. For example, Maximal Similarity-based Region Merging (MSRM) [18] is an intuitive segmentation approach which utilizes a region merging framework to meld super-pixel division. gPb-owt-ucm [2] is another segmentation method which is dependent on spectral clustering and contour detection. The IPK technique [20] utilized 3D histogram of L*a*b* tone space of the plant images for regulated segmentation of closer view/foundation.

Vukadinovic and Polder employed neural networks combined with watershed for leaves segmentation [24]. A collation study among several leaf segmentation algorithms had been presented in [27]. Well-known clustering techniques like KM, FCM, Self-organizing Map (SOM), and Particle Swarm Optimization (PSO) are also applied for leaf segmentation in [11] and SOM outperforms other tested methods visually and numerically. Other than the above techniques, deep learning is also utilized for the leaf segmentation [15]. However, deep learning needs large dataset in order to produce good results.

Therefore, it can be seen from the above discussion that different clustering strategies provide promising segmentation results. Although classical KM, FCM, SOM , and PSO are utilized for leaf segmentation, but rough set based K-Means (RKM) clustering did not used in this area according to the knowledge of the authors. Rough k-means is developed by Lingras et al. [16] and a refined version is proposed by Peters [21] and it shows the performance in image clustering domain as a similarity based clustering model like KM and FCM [14] [10] [12] [22]. RKM has been efficiently applied for the proper segmentation of tumor region from brain MRI images [14] [10], White blood cell segmentation [12], and satellite image segmentation [22]. As a consequence, the main contribution of the paper is the utilization of RKM in CIELab and its application for leaf segmentation. The proposed methodology which is represented in Figure 1 has been compared with classical KM and FCM. Experimental results show the supremacy of the proposed approach over other tested techniques.

The rest of the paper is organized as follows: Section 2 discusses the proposed methodology. Section 3 describes the experimental results and the paper is concluded in section 4.

2 Proposed Methodology

Clustering is a procedure of consortium a bunch of data into clusters that have superior intra-cluster and inferior inter-cluster resemblance among clusters. Two rudimentary types of image clustering practices are hard clustering and soft clustering. In hard clustering, one pixel can be the adherent of only one cluster and the proper exam-

[*]First Author
[†]Corresponding Author

DOI https://doi.org/10.18690/978-961-286-516-0.5
ISBN 978-961-286-516-0

ple of this is K-means [5] [9] [7]. On the contrary of the previous one, soft clustering uses a miniscule membership unlike hard clustering which makes it more practicable for real world usages. One pixel can be the fragment of several clusters with some degree of belongingness which is described by the fractional membership. Fuzzy C-means is a specimen of this mechanism which had been projected by Bezdek [3] [6]. FCM is better than hard clustering technique like K-means because it has the more ability to handle the ambiguity of gray levels. In some cases, the fuzzy degree of membership may be too descriptive for interpreting clustering results. Therefore, researchers have applied rough set theory into k-means and developed rough k-means [16] [21]clustering algorithm which manages these equidistant data objects or overlapping clusters using upper and lower approximations of each cluster. Rough set-based clustering provides a solution that is less restrictive than conventional clustering and less descriptive (specific) than fuzzy clustering. In this study, Rough K-Means has been utilized to segment the leaf images. Due to non-uniform illumination of regions, the segmentation algorithm's performance is influenced by the color spaces used. According to the literature, perceptually uniform color spaces such as L*a*b* or L*u*v* achieve much better segmentation results than non-uniform color spaces such as RGB [23], which was developed for better color representation. As a first stage in our approach, we used MATLAB to transform all RGB images to CIE L*a*b* color space, which yielded three components: L*, a*, and b*. "L*" denotes lightness, while "a*" and "b*" denote colors, with "a*" denoting red-green and "b*" denoting blue-yellow, respectively. The flowchart of the proposed work is presented in Figure 1 The brief mathematical implementation of RKM is described in section 2.1.

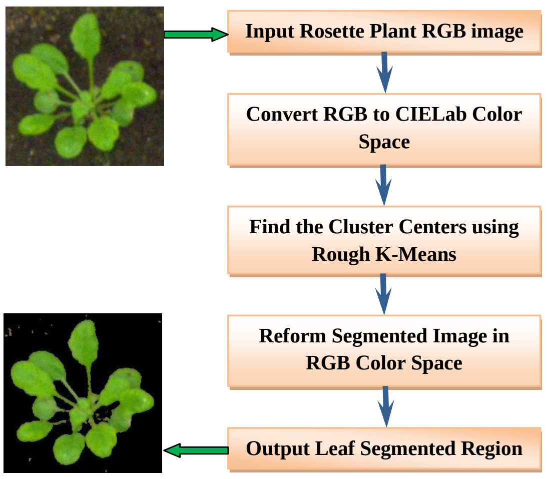

Input Rosette Plant RGB image
Convert RGB to CIELab Color Space
Find the Cluster Centers using Rough K-Means
Reform Segmented Image in RGB Color Space
Output Leaf Segmented Region

Figure 1: Flowchart of the Proposed Methodology

2.1 Rough K-Means (RKM)

Suppose, a hypothetical clustering scheme is defined as Eq. (1) which partitions U depending on the equivalence relation P. Again, assume that it may not possible to accurately describe the sets $C_i, 1 \leq i \leq k$ due to inadequate knowledge in the partition. But it is possible to define each set $C_i \in U/P$ using its lower approximation $\underline{A}(C)$and upper approximation $\bar{A}(C)$.

$$U/P = \{C_1, C_2, \ldots, C_k\} \tag{1}$$

Let, v and c_i are the vector representation of the data object and cluster C_i respectively. Upper and lower approximations of only a few subsets of U have been considered. Hence, it is not possible to verify all the properties of the rough sets [16] [21]. However, the upper and lower estimates of $C_i \in U/P$ are obligatory to follow some of the basic rough set properties which are as follows:

P1: A data entity v can be a participant of at most one lower approximation $\underline{A}(c_i)$.

P2: If a data object v is the portion of the lower approximation $\underline{A}(c_i)$, then it is also portion of the upper approximation $\bar{A}(C)$ i.e.,$v \in \underline{A}(c_i) \implies v \in \bar{A}(c_i)$.

P3: If a data article v does not belong to any lower approximation$\underline{A}(c_i)$ then it belongs to two or more upper approximations $\bar{A}(c_i)$.

In rough k-means, the lower and upper approximations of the clusters have been computed by the following rules: Let v be a data object and $d(v, z_i)$be the distance between v and z_i which is the centroid of cluster c_i.

Let $d(v, z_i) = \min_{(1 \leq j \leq k)} d(v, z_j)$

$$T = \left\{ j : \frac{d(v, z_i)}{d(v, z_i)} \leq th \, and \, i \neq j \right\} \tag{2}$$

Where, th is the threshold value specified by the user. In order to classify a data object to the correct approximation(s), the following classification criteria are being used:

R1: If the set, T is not an empty set, then the data object is classified as upper approximation of both cluster i and j. So, if $T \neq \phi$ then $[v \in \bar{A}(c_i)$ and $v \in \bar{A}(c_j), \forall j]$

R2: If T is a vacant set, the data object is being categorized as lower approximation for cluster i. Then the pixel is categorized as upper approximation for clusters i as per the hypothesis P2. So, if $T = \phi$ then $[v \in \bar{A}(c_i) \, and \, v \in \underline{A}(c_i)]$

Depending on the above deliberations the algorithm steps for rough k-means are represented as Algorithm 1.

3 Experimental Results

The experiment has been performed over 30 plant images using MatlabR2018b on Windows-10 OS, x64-based PC, Intel core i5 CPU with 8 GB RAM. The plant images are collected from [17]. The parameter settings of the utilized clustering techniques are as follows. Number of cluster prototype value depends on the user which is taken as 2 for all clustering techniques. For FCM, fuzzification parameter is taken as 2 and if maximum difference between two successive partition matrices U is less than minimal error threshold η then stop the corresponding algorithm. Mathematically, if $[Max \, U^t - U^{(t+1)}] < \eta$ then stop, where, minimal error threshold $\eta = 10^{(-5)}$. For KM and RKM, if the change in centroid values are smaller

```
1  For each cluster and data object, find the
     distance d and threshold $T$
2  Classify the data object to lower and upper
     estimates utilizing the classification criteria i.e.,
     $R1$ and $R2$.
3  Calculate the new cluster center ($mean\, z_i$) as per
     following expressions:
4  If $[\underline{A}(c_i) \neq \phi \, and \, \bar{A}(c_i) - \underline{A}(c_i) = \phi]$
5  then $z_i = \frac{\sum_{v \in \underline{A}(c_i)} v}{|\underline{A}(c_i)|}$
6  else if $\underline{A}(c_i) \neq \phi \, and \, \bar{A}(c_i) - \underline{A}(c_i) \neq \phi]$
7  then $z_i = \frac{\sum_{v \in A(c_i) - \underline{A}(c_i)} v}{|\underline{A}(c_i) - \bar{A}(c_i)|}$
8  else $z_i = w_l \times \frac{\sum_{v \in \underline{A}(c_i)} v}{|\underline{A}(c_i)|} + w_u \times \frac{\sum_{v \in A(c_i) - \underline{A}(c_i)} v}{|\underline{A}(c_i) - \bar{A}(c_i)|}$
9  $w_l + w_u = 1$ and usually, $w_l > w_u$ The
     parameters $w_l$ and $w_u$ correspond to the
     relative importance of lower and upper
     approximations respectively.
10 If the algorithm converges, then stop. Otherwise,
     repeat steps 2 to 4.
```

Algorithm 1: Procedure of Rough K-Means (RKM)

than η the stop the procedure.The rough set parameters for classical RKM are $th = 0.7, w_l = 0.6$, and $w_u = 0.4$. Threshold (th) selection in RKM is tough for different image. We have done the experiment within the range $0 < th <= 10$ and optimally set to 0.7. The performance of the utilized clustering techniques has been evaluated by calculating four ground truth based performance evaluation parameters namely accuracy, dice, Jaccard, and Matthews correlation coefficient (MCC) which are summarized in Table 1 [6] [26]. Here, TP - true positive, FP - false positive, TN - true negative, FN - false negative.

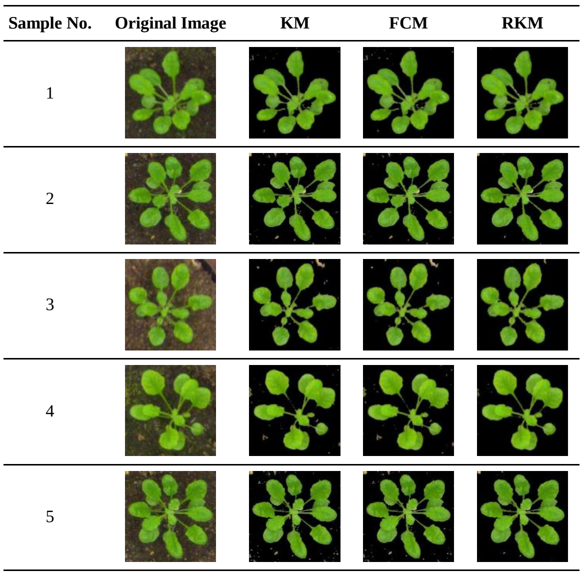


Figure 2: Color segmentation results of clustering techniques over five sample images

Table 1: Performance parameters considered for evaluation of the clustering methods.

Sl.	Parameters	Formulation and Remarks
1	Accuracy (AC)	$AC = \frac{(TP+TN)}{(FN+FP+TP+TN)}$; Accuracy is one metric for evaluating classification models. We calculate the accuracy to know how good our model predicts.
2	Dice Index (DI)	$DI = \frac{2\times}{(2\times TP+FP+FN)}$; It combines the precision and recall concepts from information retrieval. It is the harmonic mean of the precision and recall. The DI values are within the interval [0, 1] and larger the value indicates higher clustering quality.
3	Jaccard Index(JI)	$JI = DI/(2-DI)$; Jaccard similarity index measures the overlap between two sets. It is defined as the size of the intersection of two sets divided by the size of their union. The higher value indicates more similarities between two objects.
4	Matthews correlation coefficient (MCC)	$MCC = \frac{(TP\times TN - FP\times FN)}{\sqrt{(TP+FP)\times(TP+FN)\times(TN+FP)\times(TN+FN)}}$; (MCC) is a more reliable statistical rate which produces a highscore only if the prediction obtained good results in all TP, TN, FP, FN categories and proportionally both to the size of positive elements and the size of negative elements in the dataset. Higher value indicates the better results.

Table 2: Numerical values of segmentation quality parameters over five sample images

Sample No.	Method	Accuracy	Dice	Jaccard	MCC
1	KM	0.9756	0.9651	0.9325	0.9463
	FCM	0.9743	0.9633	0.9292	0.9435
	RKM	**0.9851**	**0.9791**	**0.9590**	**0.9677**
2	KM	0.9786	0.9720	0.9455	0.9548
	FCM	0.9819	0.9762	0.9536	0.9617
	RKM	**0.9845**	**0.9798**	**0.9603**	**0.9675**
3	KM	0.9730	0.9555	0.9147	0.9364
	FCM	0.9761	0.9607	0.9244	0.9440
	RKM	**0.9773**	**0.9630**	**0.9287**	**0.9476**
4	KM	0.9714	0.9541	0.9122	0.9336
	FCM	0.9720	0.9552	0.9143	0.9350
	RKM	**0.9880**	**0.9810**	**0.9627**	**0.9723**
5	KM	0.9738	0.9659	0.9341	0.9453
	FCM	0.9793	0.9729	0.9473	0.9565
	RKM	**0.9838**	**0.9788**	**0.9585**	**0.9661**

Table 3: Average numerical values of segmentation quality parameters and execution time

Method	Accuracy	Dice	Jaccard	MCC	Time (Sec.)
KM	0.9638	0.9470	0.9001	0.9200	**3.78**
FCM	0.9593	0.9417	0.8909	0.9112	4.56
RKM	**0.9662**	**0.9531**	**0.9124**	**0.9293**	8.26

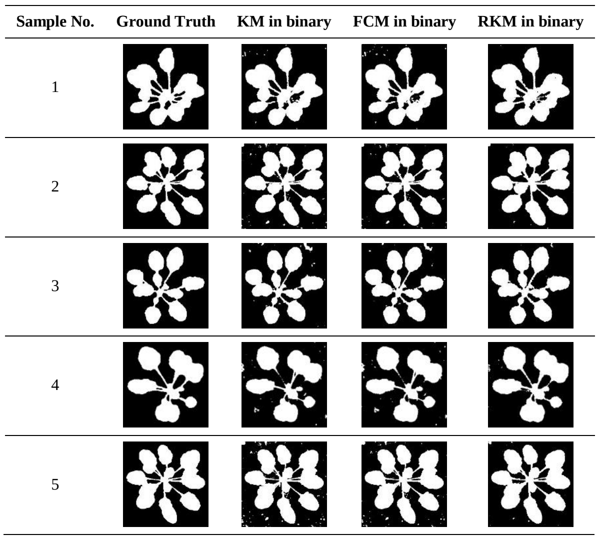


Figure 3: Ground truth images and segmentation results of clustering techniques in binary format over five sample images

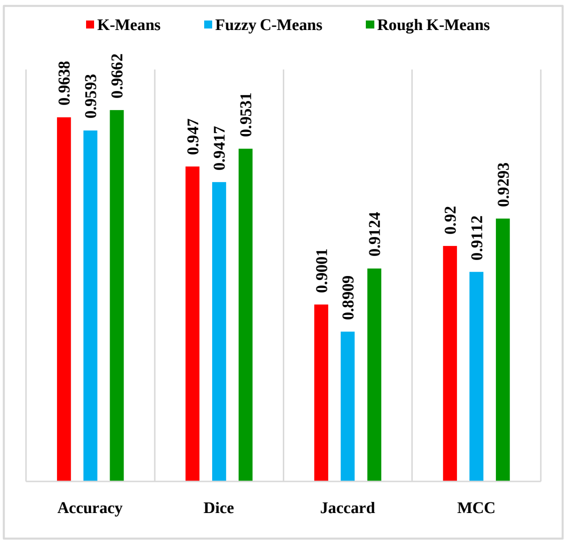


Figure 4: Graphical analysis of average quality parameters for clustering techniques

The three clustering algorithms i.e., KM, FCM, and RKM have been utilized to segment the leaves of the rosette plants. Figure 2 and 3 represents the original color plant image, the ground truth images of the leaf segmentation provided by the experts, their segmented leaf part by the three utilized algorithms binary segmented leaf part provided by the employed clustering algorithms. Figures 2 and 3 here clearly show that the RKM provides the best leaf-based segmentation results. Not only visual analysis, the segmentation efficacy of the clustering algorithms has been analyzed by computing four well-known segmentation quality parameters which are presented in Table 2. The values of the segmentation quality parameters regarding five plant samples presented in Table 2. The average values of the segmentation quality parameters over 30 images are given in Table 3. The best numerical values of the Tables 2 and 3 are given in bold. Most of the values of the quality parameters clearly reveal that RKM provides superior outcomes to other three tested clustering algorithms. The graphical representation of the average quality parameters (recorded in Table 3) is also showed in Figure 4. The average execution times of the four clustering algorithms over 30 images are also presented in Table 3. KM needs least computational effort. RKM takes the largest execution time in the same environment.

4 Conclusion

This paper presents a Rough K-Means (RKM) based clustering algorithm in CIELab color space for leaf image segmentation of rosette plants. The proposed RKM based technique is compared against two well-known conventional clustering algorithms namely K-Means and Fuzzy C-Means. An entire dataset of 30 images have been used for this experiment. Experimental results here reveals that the RKM based clustering algorithm outperforms the others and delivers the best outcomes in both the visual as well as numerical analysis for the utilized segmentation parameters. The main three limitations of the proposed method are noise sensitivity, local optima trapping and large computational time. If researched further, it can be possible to analyze the plant's growth or to detect any visually identifiable signs of disease or damage, or any possible visual anomaly. These results encourage further research in the improvement of RKM for image segmentation such as incorporation of nature-inspired optimization algorithms to overcome local optima trapping problem [8] [4].

References

[1] Aich, S., and Stavness, I. Leaf counting with deep convolutional and deconvolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision Workshops* (2017), pp. 2080–2089.

[2] Arbelaez, P., Maire, M., Fowlkes, C., and Malik, J. Contour detection and hierarchical image segmentation. *IEEE transactions on pattern analysis and machine intelligence 33*, 5 (2010), 898–916.

[3] Bezdek, J. C., Ehrlich, R., and Full, W. Fcm: The fuzzy c-means clustering algorithm. *Computers & geosciences 10*, 2-3 (1984), 191–203.

[4] Dhal, K. G., Das, A., Gálvez, J., Ray, S., and Das, S. An overview on nature-inspired optimization algorithms and their possible application in image processing domain. *Pattern Recognition and Image Analysis 30*, 4 (2020), 614–631.

[5] Dhal, K. G., Das, A., Ray, S., and Das, S. A clustering based classification approach based on modified cuckoo search algorithm. *Pattern Recognition and Image Analysis 29*, 3 (2019), 344–359.

[6] Dhal, K. G., Das, A., Ray, S., and Gálvez, J. Randomly attracted rough firefly algorithm for histogram based fuzzy image clustering. *Knowledge-Based Systems 216* (2021), 106814.

[7] Dhal, K. G., Fister Jr, I., Das, A., Ray, S., and Das, S. Breast histopathology image clustering using cuckoo search algorithm. In *Proceedings of the 5th student computer science research conference* (2018), pp. 47–54.

[8] Dhal, K. G., Fister Jr, I., and Das, S. Parameterless harmony search for image multi-thresholding. In *4th student computer science research conference (StuCosRec-2017)* (2017), pp. 5–12.

[9] Dhal, K. G., Gálvez, J., Ray, S., Das, A., and Das, S. Acute lymphoblastic leukemia image segmentation driven by stochastic fractal search. *Multimedia Tools and Applications* (2020), 1–29.

[10] Dobe, O., Sarkar, A., and Halder, A. Rough k-means and morphological operation-based brain tumor extraction. In *Integrated Intelligent Computing, Communication and Security*. Springer, 2019, pp. 661–667.

[11] Ghosal, D., Das, A., and Dhal, K. G. A comparative study among clustering techniques for leaf segmentation in rosette plants. *Pattern Recognition and Image Analysis 31*, 4 (2021).

[12] Inbarani H, H., Azar, A. T., et al. Leukemia image segmentation using a hybrid histogram-based soft covering rough k-means clustering algorithm. *Electronics 9*, 1 (2020), 188.

[13] Koornneef, M., Hanhart, C., van Loenen-Martinet, P., and Blankestijn de Vries, H. The effect of daylength on the transition to flowering in phytochrome-deficient, late-flowering and double mutants of arabidopsis thaliana. *Physiologia Plantarum 95*, 2 (1995), 260–266.

[14] Kumar, D. M., Satyanarayana, D., and Prasad, M. G. An improved gabor wavelet transform and rough k-means clustering algorithm for mri brain tumor image segmentation. *Multimedia Tools and Applications 80*, 5 (2021), 6939–6957.

[15] Kumar, J. P., and Domnic, S. Rosette plant segmentation with leaf count using orthogonal transform and deep convolutional neural network. *Machine Vision and Applications 31*, 1 (2020), 1–14.

[16] Lingras, P., and West, C. Interval set clustering of web users with rough k-means. *Journal of Intelligent Information Systems 23*, 1 (2004), 5–16.

[17] Minervini, M., Fischbach, A., Scharr, H., and Tsaftaris, S. A. Finely-grained annotated datasets for image-based plant phenotyping. *Pattern recognition letters 81* (2016), 80–89.

[18] Ning, J., Zhang, L., Zhang, D., and Wu, C. Interactive image segmentation by maximal similarity based region merging. *Pattern Recognition 43*, 2 (2010), 445–456.

[19] Orlando, F., Napoli, M., Dalla Marta, A., Natali, F., Mancini, M., Zanchi, C., and Orlandini, S. Growth and development responses of tobacco (nicotiana tabacum l.) to changes in physical and hydrological soil properties due to minimum tillage.

[20] Pape, J.-M., and Klukas, C. 3-d histogram-based segmentation and leaf detection for rosette plants. In *European Conference on Computer Vision* (2014), Springer, pp. 61–74.

[21] Peters, G. Some refinements of rough k-means clustering. *Pattern Recognition 39*, 8 (2006), 1481–1491.

[22] Raj, A., and Minz, S. Spatial rough k-means algorithm for unsupervised multi-spectral classification. In *International Conference on Information and Communication Technology for Intelligent Systems* (2020), Springer, pp. 215–226.

[23] Sarkar, S., Das, S., and Chaudhuri, S. S. A multilevel color image thresholding scheme based on minimum cross entropy and differential evolution. *Pattern Recognition Letters 54* (2015), 27–35.

[24] Scharr, H., Minervini, M., French, A. P., Klukas, C., Kramer, D. M., Liu, X., Luengo, I., Pape, J.-M., Polder, G., Vukadinovic, D., et al. Leaf segmentation in plant phenotyping: a collation study. *Machine vision and applications 27*, 4 (2016), 585–606.

[25] Telfer, A., Bollman, K. M., and Poethig, R. S. Phase change and the regulation of trichome distribution in arabidopsis thaliana. *Development 124*, 3 (1997), 645–654.

[26] Thanh, D. N., Prasath, V. S., Hien, N. N., et al. Melanoma skin cancer detection method based on adaptive principal curvature, colour normalisation and feature extraction with the abcd rule. *Journal of digital imaging* (2019), 1–12.

[27] Vukadinovic, D., and Polder, G. Watershed and supervised classification based fully automated method for separate leaf segmentation. In *The Netherland Congress on Computer Vision* (2015), pp. 1–2.

[28] Walter, A., and Schurr, U. The modular character of growth in nicotiana tabacum plants under steady-state nutrition. *Journal of Experimental Botany 50*, 336 (1999), 1169–1177.

Adversarial Image Perturbation with a Genetic Algorithm

Rok Kukovec
University of Maribor,
Faculty of Electrical
Engineering and Computer Science,
Koroška cesta 46, 2000 Maribor, Slovenia
rok.kukovec@student.um.si

Špela Pečnik
University of Maribor,
Faculty of Electrical
Engineering and Computer Science,
Koroška cesta 46, 2000 Maribor, Slovenia
spela.pecnik@um.si

Iztok Fister Jr.
University of Maribor,
Faculty of Electrical
Engineering and Computer Science,
Koroška cesta 46, 2000 Maribor, Slovenia
iztok.fister1@um.si

Sašo Karakatič
University of Maribor,
Faculty of Electrical
Engineering and Computer Science,
Koroška cesta 46, 2000 Maribor, Slovenia
saso.karakatic@um.si

Abstract

The quality of image recognition with neural network models relies heavily on filters and parameters optimized through the training process. These filters are different compared to how humans see and recognize objects around them. The difference in machine and human recognition yields a noticeable gap, which is prone to exploitation. The workings of these algorithms can be compromised with adversarial perturbations of images. This is where images are seemingly modified imperceptibly, such that humans see little to no difference, but the neural network classifies the motif incorrectly. This paper explores the adversarial image modification with an evolutionary algorithm, so that the AlexNet convolutional neural network cannot recognize previously clear motifs while preserving the human perceptibility of the image. The experiment was implemented in Python and tested on the ILSVRC dataset. Original images and their recreated counterparts were compared and contrasted using visual assessment and statistical metrics. The findings suggest that the human eye, without prior knowledge, will hardly spot the difference compared to the original images.

Keywords adversarial perturbation, AlexNet, CNN, computer vision, evolutionary algorithms

1 Introduction

Computer vision algorithms are already used widely in every day applications, but the safety concerns persist regarding their reliability. Leaving vital decisions to them can cause dire consequences in cases of error. Therefore, additional caution is necessary in most use cases. Such algorithms have to be tested extensively before they are allowed to make such decisions on their own.

Deep neural networks are currently the state-of-the-art technology for recognizing motifs from an image. Computer vision achieves near-human-level accuracy in recognition, and the question arises of the key differences between human and computer vision. They return predicted labels and their corresponding certainties. The problem arises when there are high certainties for wrong labels [2].

This paper presents an approach for adversarial image perturbation with evolutionary algorithms, with the goal of misguiding the AlexNet convolutional neural network (CNN). The implemented approach demonstrates how simple and effective adversarial perturbation is, and how vulnerable every day image recognition models are. The implemented approach aims to recreate the image as similar to the original image as possible, keeping the human perception of the motif intact, while maximizing the error of the image recognition model. Pixel values in certain places are changed such that computer vision fails to classify them correctly.

2 Related work

The inspiration for this paper derives from [9], where the authors implemented an adversarial perturbation deceiving computer vision with only changing one pixel in the original image. This attack was carried out on images of very low resolution, which is the reason for its success.

In the paper authored by Fawzi et al. [1], an analysis was made of the resistance of computer vision algorithms to adversary disturbances. The existence of adversarial examples was confirmed, as there is an upper bound to robustness. The goal was to find the correlation between robustness against random and adversarial noise. As long as the boundary is so high that the recreated image has to be completely distorted, it does not indicate a problem. A problem arises if the image is human-recognizable and the recognition algorithm fails its prediction with high certainty. Several different models of machine learning, including CNNs, misclassify adversarial examples consistently. These are intentionally created, small interfer-

DOI https://doi.org/10.18690/978-961-286-516-0.6
ISBN 978-961-286-516-0

ences that are detrimental for the recognition algorithm [8]. The paper [13] shows that a universal adversarial perturbation is possible. One adversarial noise filter can be applied routinely to many different images. In the paper [2], there are examples of specifically produced images in which the human eye only sees random noise, yet the algorithm is near certain that there is a motif. The paper [4] shows that a successful adversarial perturbation against one neural network is likely to succeed against a variety of network architectures trained on different data sets.

A distinctive quality of this paper is that it is readily accessible to non-experts. It shows that implementations of adversarial perturbations are not limited only to teams of advanced researchers supported by both technical and financial capabilities. The attacker requires only a basic understanding of machine learning. The experiment uses only open-source libraries and a small amount of understandable custom code. Despite the straightforward approach, results are comparable to the work mentioned above.

3 Implementing adversarial perturbation on AlexNet CNN

The main objective of the approach is that the solution image is modified in accordance with two objectives: (1) Similarity to the original image, and (2) AlexNet's certainty during misrecognition. The optimization method pursuing these objectives is a genetic algorithm. The goal is that no change could be noticed by the human eye in the reproduced image without prior knowledge.

The following Python libraries were used:

- NumPy [11] is used for numerical calculations,
- OpenCV [3] and Pillow [6] for image preprocessing,
- Scikit-image [14] for the structural similarity index measure metric,
- PyTorch [12] is used for a pre-trained AlexNet CNN.
- GARI - Genetic Algorithm for Reproducing Images is used for the EA (Evolutionary algorithm) [7].

The proposed approach is divided into the following interconnected parts:

- Generation of sets of candidate solutions,
- Evaluation of the image similarity between the candidate solution and the original image using the normalized average of absolute pixel differences,
- Classification of the candidate solution using the AlexNet image recognition model,
- Computation of the fitness value for the candidate solution,
- Selection of the fittest candidate images for further reproduction.

Figure 1 shows the initial idea of the implementation of the adversarial perturbation. The start block represents the execution of the experiment with any given original image. The evolutionary algorithm creates candidate solutions, which are evaluated using two separate criteria, which, together, form a fitness function. It combines both results using fuzzy logic's operator AND ($\wedge$). The value of normalized average absolute pixel difference is multiplied with AlexNet's certainty into a wrong prediction. The process is repeated iteratively until the termination condition is reached.

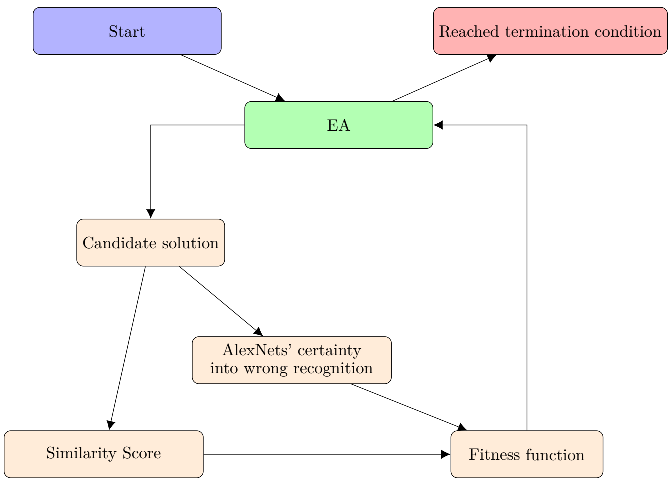

Start
Reached termination condition
EA
Candidate solution
AlexNets' certainty into wrong recognition
Similarity Score
Fitness function

Figure 1: Diagram of the proposed adversarial perturbation.

```
Data: Original image
Result: Recreated image
1 initialization;
2 create first candidate solution;
3 while termination goal not reached do
4     calculate similarity score;
5     check AlexNet's certainty into wrong
        prediction;
6     calculate fitness value;
7     send score to evolutionary algorithm;
8     create new candidate solutions;
9 end
```

Algorithm 1: Algorithm in pseudo-code

It was shown that the combination of AlexNet's predictions, genetic algorithm and evaluation of the fitness function was very time-consuming. Thus, it was not possible to recreate the image within the set time frame to the point of recognition by the human eye. The bottleneck appeared in the time-consuming evaluation of candidate solutions by AlexNet. It renders the attack infeasible for use cases where real-time solutions are needed.

We bypassed this bottleneck somewhat by not running AlexNet before the starting 80,000 iterations at all, since the first recreated images are random noise, which was optimized towards our goal. Initially, the only feedback given to the EA was the similarity score. It turned out that the recreated image was recognizable to the human eye much earlier than to AlexNet. AlexNet's predictions were only calculated after the candidate image was sufficiently similar. Once AlexNet recognizes the image, the evolutionary algorithm can start calling our final fitness function. It comprises of both the similarity score, as well as AlexNet's predicted class and its corresponding certainty.

Figure 2 shows a working version of the experiment. Presented is the detailed control flow dictating the entry of AlexNet into fitness value calculation. The experiment is divided into two phases. Phase one consists mainly of quick operations. No phase transition conditions are checked in the first 80,000 generations. Depending on the image, AlexNet started giving the first correct classification at about 30,000 generations. Towards the end of the first phase, correct classification is checked. If the prediction is correct, we advance to the second phase. It aims to create an adversarial perturbation. The output of AlexNet is an array of sorted certainties with labels. For the calculation of fitness function, the value is taken from the incorrect label which has the highest certainty and is combined with the similarity score.

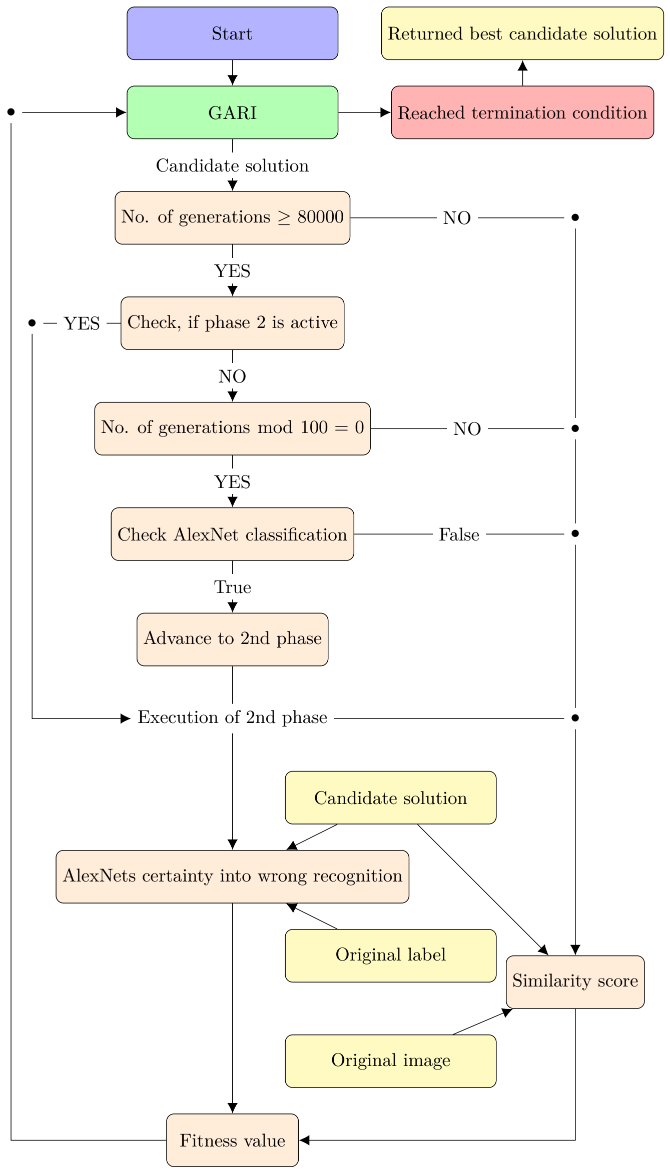

Start
Returned best candidate solution
GARI
Reached termination condition
Candidate solution
No. of generations ≥ 80000
NO
YES
YES
Check, if phase 2 is active
NO
No. of generations mod 100 = 0
NO
YES
Check AlexNet classification
False
True
Advance to 2nd phase
Execution of 2nd phase
Candidate solution
AlexNets certainty into wrong recognition
Original label
Similarity score
Original image
Fitness value

Figure 2: Flowchart of the final implementation of the proposed approach.

4 Results

The results of the experiment are evaluated visually and using statistical metrics. Terminating conditions were set as follows:

- Time limit of 2 hours reached,
- Calculated fitness exceeded 0.99,
- Algorithm finished both phases.

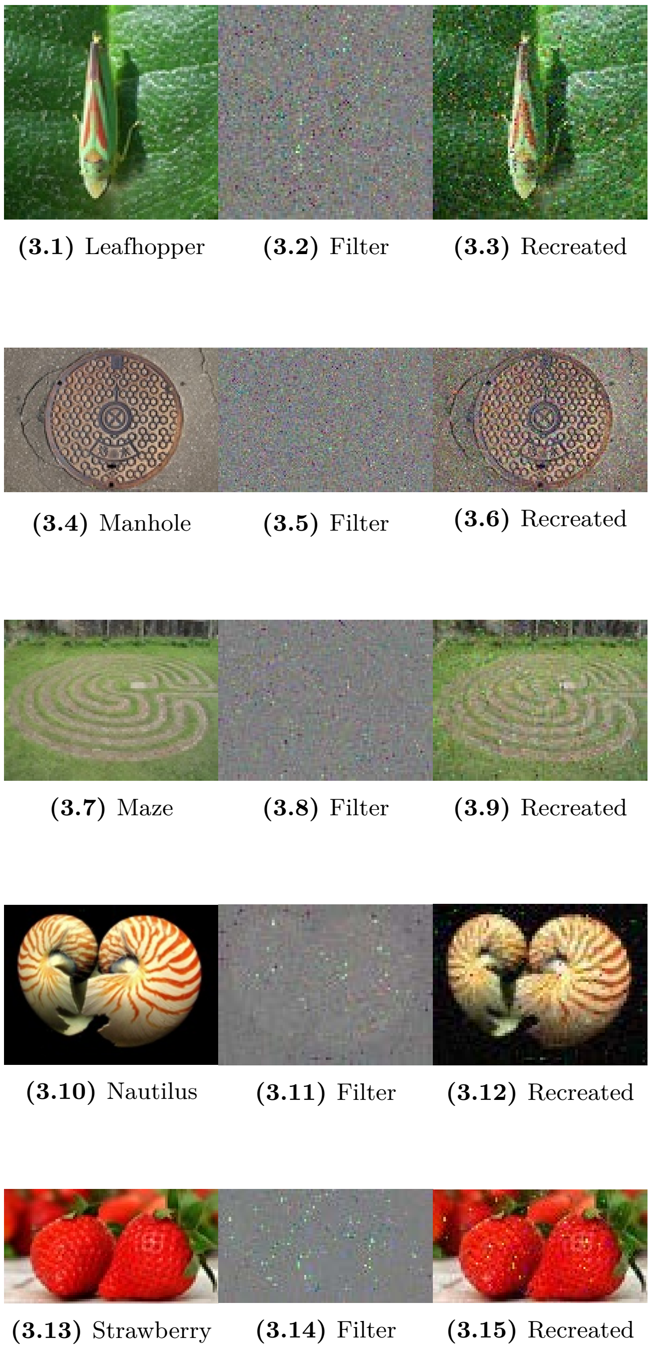


Figure 3: Original images, adversarial filters and recreated images.

The benchmark value was set to 0.99, since it was forcing both factors, normalized average of absolute pixel differences and AlexNet's certainty, into wrong prediction to be above 0.99. The product of two numbers between 0 and 1 is smaller than either factor.

Since images are difficult to evaluate qualitatively and the normalized mean of sum of absolute errors was already used in the evaluation process, new statistical metrics were introduced:

- Mean Squared Error (MSE),
- Peak signal-to-noise ratio (PSNR), and
- Structural similarity index measure (SSIM).

Results showed a promising direction, but they were not optimized fully due to operational limitations. The compromise was agreed upon deceiving AlexNet's prediction to the closest label in the feature space.

4.1 Examples of missclassified images

The results of the experiment are shown in Table 1. Recreated images are shown in Figure 3.

Original category	Category after attack	Certainty into missclassified label
Leafhopper	Lacewing	99.97%
Manhole-cover	Electric ray	99.98%
Maze	Hay	99.97%
Nautilus	Brain coral	99.98%
Strawberries	Bell pepper	99.97%

Table 1: Results of images in Figure 3

The calculated metrics on different recreated images achieve relatively high values. The human eye recognizes the motif of the image. The attack was carried out successfully and results are shown in Table 2.

Picture	MSE	PSNR	SSIM
Agama	768.69	29.51dB	0.62
Baseball	956.16	29.04dB	0.56
LeafHopper	666.89	29.52dB	0.77
Manhole cover	642.42	29.53dB	0.77
Maze	270.50	31.11dB	0.79
Nautilus	396.71	30.58dB	0.81
Nautilus 2	667.03	29.65dB	0.73
Panda	908.09	29.10dB	0.75
Rosehip	944.62	29.29dB	0.68
Strawberry	394.05	31.52dB	0.83
Sulphur butterfly	1015.85	28.82dB	0.61
Upright piano	975.15	29.00dB	0.66

Table 2: Calculated metrics of recreated images

One of the goals set was to recreate images in the input resolution of AlexNet (meaning 224·224 pixels). This goal was not reached because the time-complexity growth rate was non-linear. Recreated images were around $100 \cdot 100$ pixels in resolution. Figure 3 shows recreated images that, without prior-knowledge, it is hard to spot the difference, taking into account that the images are relatively small.

5 Discussion

Despite the limitations of the experiment, we showed that adversarial perturbations are possible to implement in a relatively short time with the help of genetic algorithms. Future research may point to one of the following six directions:

- Speeding up the process of optimization,
- Deceiving computer vision into a custom label,
- Selecting a more complex CNN,
- Testing other optimization methods (i.e. even other nature-inspired algorithms [5]),
- Testing with only using some features [10] to speed up the optimization process, and
- Protection against adversarial noise.

References

[1] Alhussein Fawzi, e. a. Analysis of classifiers' robustness to adversarial perturbations. *CoRR abs/1502.02590* (2015).

[2] Anh Mai Nguyen, e. a. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. *CoRR abs/1412.1897* (2014).

[3] Bradski, G. The OpenCV Library. *Dr. Dobb's Journal of Software Tools* (2000).

[4] Christian Szegedy, e. a. Intriguing properties of neural networks, 2014.

[5] Fister Jr, e. a. A brief review of nature-inspired algorithms for optimization. *arXiv preprint arXiv:1307.4186* (2013).

[6] Fredrik Lundh. Pillow - Pillow (PIL Fork) 8.2.0 Documentation, 2021.

[7] Gad, A. F. ahmedfgad/GARI: GARI (Genetic Algorithm for Reproducing Images) reproduces a single image using Genetic Algorithm (GA) by evolving pixel values.

[8] Ian J. Goodfellow, e. a. Explaining and harnessing adversarial examples, 2015.

[9] Jiawei Su, e. a. One pixel attack for fooling deep neural networks. *CoRR abs/1710.08864* (2017).

[10] Karakatič, S. Evopreprocess—data preprocessing framework with nature-inspired optimization algorithms. *Mathematics 8*, 6 (2020), 900.

[11] NumPy. What is NumPy? — NumPy v1.20 Manual, 2021.

[12] Paszke, A. e. a. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035.

[13] Seyed-Mohsen Moosavi-Dezfooli, e. a. Universal adversarial perturbations. *CoRR abs/1610.08401* (2016).

[14] van der Walt, e. a. scikit-image: image processing in Python. *PeerJ 2* (6 2014), e453.

Fast Recognition of Some Parametric Graph Families

Nina Klobas
Durham University,
Department of Computer Science,
Upper Mountjoy Campus, Stockton Road,
Durham DH1 3LE,United Kingdom
`nina.klobas@durham.ac.uk`

Matjaž Krnc
University of Primorska,
Faculty of Mathematics, Natural Sciences
and Information Technologies,
Glagoljaška ulica 8, 6000 Koper, Slovenia
`matjaz.krnc@famnit.upr.si`

Abstract

Recognizing graphs with high level of symmetries is hard in general, and usually requires additional structural understanding. In this paper we study a particular graph parameter and motivate its usage by devising efficient recognition algorithm for the family of *I*-graphs.

For integers ℓ, λ, m a simple graph is $[\ell, \lambda, m]$-cycle regular if every path of length ℓ belongs to exactly λ different cycles of length m. We identify all $[1, \lambda, 8]$-cycle regular *I*-graphs and, as a consequence, describe linear recognition algorithm for the observed family.

Similar procedure can be used to devise the recognition algorithms for Double generalized Petersen graphs and folded cubes. Besides that, we believe the structural observations and methods used in the paper are of independent interest and could be used for solving other algorithmic problems.

Keywords *I*-graphs, double generalized Petersen graphs, folded cubes, recognition algorithm, cycle regularity.

1 Introduction

Important graph classes such as bipartite graphs, (weakly) chordal graphs, perfect graphs and forests are defined or characterized by their cycle structure. A particularly strong description of a cyclic structure is the notion of *cycle-regularity*, introduced by Mollard [11]:

> *For integers l, λ, m a simple graph is $[l, \lambda, m]$-cycle regular if every path on $l + 1$ vertices belongs to exactly λ different cycles of length m.*

It is perhaps natural that cycle-regularity mostly appears in the literature in the context of symmetric graph families such as hypercubes, Cayley graphs or circulants.

Understanding the structure of subgraphs of hypercubes which avoid all 4-cycles does not seem to be easy. Indeed, a question of Erdős regarding how many edges can such a graph contain remains open after more than 30 years [5].

In this paper we study cycle-regularity and more general cyclic aspects of a family of *I*-graphs, with the focus of devising an efficient recognition algorithm. Similar approach can be extended also to two other graph families, namely Double generalized Petersen graphs and folded cubes. Due to the space constraints the study of these two families is not covered here, therefore we defer interested readers to the full version of this paper [9].

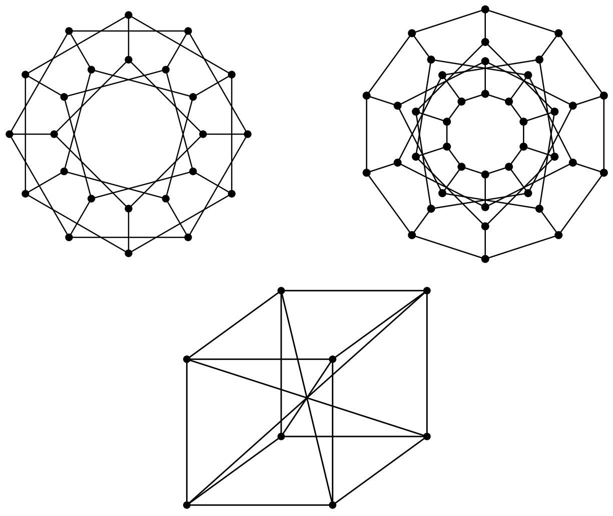


Figure 1: *I*-graph $I(12, 2, 3)$, double generalized Petersen graph $\mathrm{DP}(10, 2)$, and folded cube FQ_4.

***I*-graphs** were introduced in the Foster census [6], and are trivalent or cubic graphs with edges connecting vertices of two star polygons. They form a natural generalization of the well-known *generalized Petersen graphs* introduced in 1950 by Coxeter [3] and later named by Watkins in 1969 [14]. The family of *I*-graphs has been studied extensively with respect to their automorphism group and isomorphisms [1, 7, 13], Hamiltonicity [2], spectrum [12], and independence number [4, 8].

Our first result identifies all $[1, \lambda, 8]$-cycle regular members and determines the corresponding values of λ.

Theorem 1. *An arbitrary I-graph is never $[1, \lambda, 8]$-cycle regular, except when isomorphic to $I(n, j, k)$ where $j = 1$ and $(n, k) \in \{(3, 1), (4, 1), (5, 2), (8, 3), (10, 2), (10, 3), (12, 5), (13, 5), (24, 5), (26, 5)\}$.*

These structural results are used to devise the recognition algorithm for *I*-graphs.

Theorem 2. *I-graphs can be recognized in linear time.*

If the input graph is a member of the observed family, we

DOI https://doi.org/10.18690/978-961-286-516-0.7
ISBN 978-961-286-516-0

not only provide its parameters but also give a certificate of correctness, i.e. we give an exact isomorphism.

1.1 Preliminaries

Unless specified otherwise, all graphs in this paper are finite, simple, undirected and connected. For a given graph G we use a standard notation for a set of vertices $V(G)$ and a set of edges $E(G)$. A k-cycle C in G, on vertices $v_1, v_2, \ldots, v_k$ from $V(G)$ is denoted as $(v_1, \ldots, v_k)$. For integers a and b we denote with $\gcd(a, b)$ the greatest common divisor of a and b respectively.

Definition 3. Let l, λ, m be positive integers. A simple graph G is $[l, \lambda, m]$-cycle regular if every path on $l+1$ vertices of G belongs to exactly λ different m-cycles of G.

It is easy to see that $[1, \lambda, 8]$-cycle regular cubic graphs are also $[0, 3\lambda/2, 8]$-cycle regular, but the converse does not hold. Related to this we define a function $\sigma : E(G) \mapsto \mathbb{N}$, where $\sigma(e)$ corresponds to the number of distinct 8-cycles an edge e belongs to. We call $\sigma(e)$ an *octagon value* of an edge e, and we say that a graph G has a *constant octagon value* if σ is a constant function.

2 Structural analysis

Before we start with the analysis of the observed graph family we need to formally define it and present some of its basic properties.

Definition 4. Let n, j, k be positive integers for which $n \geq 3$ and $n \geq j, k \geq 1$. An I-graph $I(n, j, k)$ is a graph on vertices $\{u_0, u_1, \ldots, u_{n-1}, w_0, w_1, \ldots, w_{n-1}\}$, with the edge set consisting of outer edges $u_i u_{i+j}$, inner edges $w_i w_{i+k}$ and spoke edges $u_i w_i$, where the subscripts are taken modulo n.

Without loss of generality we always assume that $j, k < n/2$. Since $I(n, j, k)$ is isomorphic to $I(n, k, j)$, we restrict ourselves to cases when $j \leq k$. It is well known [1] that an I-graph $I(n, j, k)$ is disconnected whenever $d = \gcd(n, j, k) > 1$. In this case it consists of d copies of $I(n/d, j/d, k/d)$. Therefore, throughout the paper we consider only graphs $I(n, j, k)$ where $\gcd(n, j, k) = 1$. We also know [7] that two I-graphs $I(n, j, k)$ and $I(n, j', k')$ are isomorphic if and only if there exists an integer a, which is relatively prime to n, for which either $\{j', k'\} = \{aj \pmod n, ak \pmod n\}$ or $\{j', k'\} = \{aj \pmod n, -ak \pmod n\}$. Throughout the paper, whenever we discuss I-graphs with certain parameters, we consider only the lexicographically smallest possible parameters by which the graph is uniquely determined.

2.1 Equivalent 8-cycles

A particular member of automorphism group of every I-graph is a *rotation* defined as: $\rho(u_i) = u_{i+1}$, $\rho(w_i) = w_{i+1}$. Clearly, applying n times the rotation ρ yields an identity automorphism. When acting on I-graphs with ρ we get 3 edge orbits: orbit of outer edges E_J, orbit of spoke edges E_S and orbit of inner edges E_I. Edges from the same orbit E_J, E_S, or E_I have the same octagon value, which we denote by σ_J, σ_S and σ_I, respectively. Therefore the *octagon value of an I-graph* is said to be a triple $(\sigma_J, \sigma_S, \sigma_I)$.

We say that two 8-cycles of an I-graph are *equivalent* if we can map one into the other using rotation ρ. Let $G \simeq I(n, j, k)$ be an arbitrary I-graph and let C be one of its 8-cycles. With $\gamma(C)$ we denote the number of equivalent 8-cycles to C in G. Each 8-cycle contributes to the octagon value of an I-graph. We denote the contributed amount with $\tau(C)$, defined as the triple $(\delta_j, \delta_s, \delta_i)$, where we calculate $\delta_j, \delta_s, \delta_i$ by counting the number of outer, spoke and inner edges of a cycle and multiply these numbers with $\gamma(C)/n$. If a graph G admits m non-equivalent 8-cycles $C_1, C_2, \ldots, C_m$, one may calculate its octagon value $(\sigma_J, \sigma_S, \sigma_I)$ as $\sum_{i=1}^{m} \tau(C_i)$.

The following claim serves also as an example of the above-mentioned definitions.

Claim 5. *For $I(n, j, k)$ where $n > 3$ and integers $k, j < n/2$ there always exists an 8-cycle.*

Indeed, if $k \neq j$ it is of the form

$$C^* = (w_0, w_{\pm k}, u_{\pm k}, u_{\pm k \pm j}, w_{\pm k \pm j}, w_{\pm j}, u_{\pm j}, u_0).$$

If $k = j$ it is of the form

$$C_7 = (u_0, u_k, u_{2k}, u_{3k}, w_{3k}, w_{2k}, w_k, w_0).$$

2.2 Characterization of non-equivalent 8-cycles

Our aim is to identify all possible 8-cycles that can appear in an arbitrary I-graph and determine their contribution towards the octagon value of the graph. It is easy to see that an arbitrary 8-cycle can have either 4, 0 or 2 spoke edges, so we obtained this list by distinguishing 8-cycles by the number of spoke edges they admit. In the case of the 8-cycle admitting 2 spoke edges, we further distinguish cases by the number of outer and inner edges within a given 8-cycle.

Due to space constraints we present the analysis of 8-cycles admitting 4 spoke edges only, as it is the easiest case to deal with (for remaining cases see [9]). This analysis leads to complete characterisation of 8-cycles for the family of I-graphs, presented in the Table 1.

8-cycles with 4 spoke edges In addition to 4 spoke edges the 8-cycle must also have two inner and two outer edges. When using the spoke edge there are two options for choosing an inner (outer) edge. After considering all cases it is easy to see that there can be just two such 8-cycles, C^* (see Claim 5), which exists whenever $j \neq k$, and C_0, which is of the following form:

$$C_0 = (w_0, w_{\pm k}, u_{\pm k}, u_{\pm k \pm j}, w_{\pm k \pm j}, w_{\pm 2k \pm j}, u_{\pm 2k \pm j}, u_{\pm 2k \pm 2j}).$$

Cycle C_0 exists whenever $2k + 2j = n$. One can verify easily, that n applications of the rotation ρ to C^* and $n/2$ applications of the rotation ρ to cycle C_0 maps the cycle back to itself. Therefore there are n equivalent cycles to C^* and $n/2$ equivalent cycles to C_0 in an I-graph $I(n, j, k)$ and they contribute $(2, 4, 2)$ and $(1, 2, 1)$, respectively, to the graph octagon value.

2.3 Obtaining constant octagon value

Every 8-cycle of an I-graph contributes to the octagon value of each edge partition. It turns out that if we can

Label	**A representative of an 8-cycle**	**Existence conditions**	$\boldsymbol{\tau(C)}$	$\boldsymbol{\gamma(C)}$
C^*	$(w_0, w_{\pm k}, u_{\pm k}, u_{\pm k\pm j}, w_{\pm k\pm j}, w_{\pm j}, u_{\pm j}, u_0)$	$k \neq j$ and $n > 4$	$(2, 4, 2)$	n
C_0	$(w_0, w_{\pm k}, u_{\pm k}, u_{\pm k\pm j}, w_{\pm k\pm j}, w_{\pm 2k\pm j}, u_{\pm 2k\pm j}, u_{\pm 2k\pm 2j})$	$2k + 2j = n$	$(1, 2, 1)$	$n/2$
C_1	$(u_0, u_j, u_{2j}, u_{3j}, u_{4j}, u_{5j}, u_{6j}, u_{7j})$	$8j = n$ or $3n$	$(0, 0, 1)$	$n/8$
C_2	$(w_0, w_k, w_{2k}, w_{3k}, w_{4k}, w_{5k}, w_{6k}, w_{7k})$	$8k = n$ or $3n$	$(1, 0, 0)$	$n/8$
C_3	$(w_0, w_k, w_{2k}, w_{3k}, w_{4k}, w_{5k}, u_{5k}, u_{5k+j})$	$5k + j = n$ or $2n$	$(5, 2, 1)$	n
	$(w_0, w_k, w_{2k}, w_{3k}, w_{4k}, w_{5k}, u_{5k}, u_{5k-j})$	$5k - j = n$ or $2n$		
C_4	$(u_0, u_j, u_{2j}, u_{3j}, u_{4j}, u_{5j}, w_{5j}, w_{5j+k})$	$k + 5j = n$ or $2n$	$(1, 2, 5)$	n
	$(u_0, u_j, u_{2j}, u_{3j}, u_{4j}, u_{5j}, w_{5j}, w_{5j-k})$	$5j - k = 2n$ or n or 0		
C_5	$(w_0, w_k, w_{2k}, w_{3k}, w_{4k}, u_{4k}, u_{4k+j}, u_{4k+2j})$	$4k + 2j = n$ or $2k + j = n$	$(4, 2, 2)$	n
	$(w_0, w_k, w_{2k}, w_{3k}, w_{4k}, u_{4k}, u_{4k-j}, u_{4k-2j})$	$4k - 2j = n$		
C_6	$(u_0, u_j, u_{2j}, u_{3j}, u_{4j}, w_{4j}, w_{4j+k}, w_{4j+2k})$	$2k + 4j = n$ or $k + 2j = n$	$(2, 2, 4)$	n
	$(u_0, u_j, u_{2j}, u_{3j}, u_{4j}, w_{4j}, w_{4j-k}, w_{4j-2k})$	$4j - 2k = n$ or 0		
C_7	$(w_0, w_k, w_{2k}, w_{3k}, u_{3k}, u_{3k+j}, u_{3k+2j}, u_{3k+3j})$	$3k + 3j = n$ or $2n$	$(3, 2, 3)$	n
	$(w_0, w_k, w_{2k}, w_{3k}, u_{3k}, u_{3k-j}, u_{3k-2j}, u_{3k-3j})$	$3k - 3j = n$ or 0		

Table 1: All non-equivalent 8-cycles of I-graphs, their existence conditions, their contribution towards the octagon value of an I-graph τ, number of their equivalent cycles in an I-graph γ.

identify at least one edge partition of a graph, we can easily determine its parameters. Therefore, we want to find graphs with constant octagon value. These are graphs for which all edges touch the same number of 8-cycles. They are called $[1, \lambda, 8]$-cycle regular graphs. We consider all possible collections of 8-cycles and determine octagon values of I-graphs admitting those 8-cycles. Since I-graphs are defined with 3 parameters and all 8-cycles give constraints for these parameters, it is enough to consider collections of at most 4 cycles, to uniquely determine all $[1, \lambda, 8]$-cycle regular graphs. After a thorough analysis (see [9]) we see that the list of $[1, \lambda, 8]$-cycle regular I-graphs consists of 10 members (see Figure 2). Surprisingly, it turns out that all $[1, \lambda, 8]$-cycle regular I-graphs are in the family of generalized Petersen graphs. This proves Theorem 1.

3 Recognition algorithm

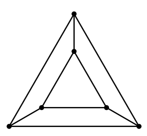
The recognition algorithm relies on the fact that there is just a small number of I-graphs (ten) with the constant octagon value. In particular, whenever the input graph G of the Algorithm 1 is a member of the family of I-graphs and is not $[1, \lambda, 8]$-cycle regular, we can immediately identify one of its edge orbits (E_I, E_J, or E_S), of size $|V(G)|/2$. Since the octagon value of each edge is computed in constant time and there is a finite number of the $[1, \lambda, 8]$-cycle regular I-graphs the Theorem 2 holds.

```
   Data: connected cubic graph $G$
   Result: parameters $(n, j, k)$ and isomorphism to
           $I(n, j, k)$, if they exists.
 1 $\mathcal{P} \leftarrow$ an empty dictionary
 2 for $e \in E(G)$ do
 3     $s = $ octagonValue$(e)$ ;   /* calculate $\sigma(e)$ */
 4     $\mathcal{P}[s]$.append$(e)$
 5 end
 6 $U \leftarrow$ an item of $\mathcal{P}$ with minimum positive cardinality
 7 if $G[U]$ is a 2-factor ;   /* 2-factor is a 2-regular graph */
 8 then
 9     $U \leftarrow \{e \mid e \in E(G)$, $e$ is adjacent to an edge
           of $U\}$ /* a perfect matching in $G$ */
10 end
11 return Extend$(G, U)$
```

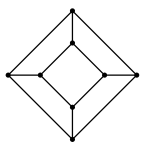
Algorithm 1: Recognition procedure for I-graphs

Correctness and time complexity of the algorithm. We first note, that if G is not cubic then it does not belong to observed graph families. Since checking whether a graph is cubic takes linear time we simply assume that the input graph is cubic. Furthermore, if G is not connected then it can only be a member of the family of I-graphs whenever it consists of multiple copies of a smaller I-graph G'. However, this case can easily be resolved by separately checking each part, so we can assume that the input graph is connected. Algorithm 1 consists of the following 3 parts.

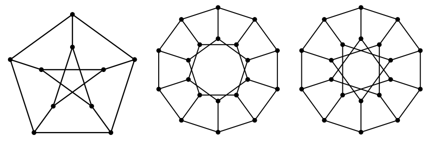
1. **Partitioning the edges with respect to the octagon value.**
   The algorithm determines the octagon value of each edge $e \in E(G)$ and builds a partition set $\mathcal{P}$ of graph edges (see lines 1 – 4). Since G is cubic and all 8-cycles containing edge e consist of edges which are at distance at most 4 from e, it is enough to check a subgraph H of G of order at most 62, to calculate the octagon value of an edge e. Therefore, calculation of octagonValue(e) takes $O(1)$ time for each edge e and this whole part is performed in $\Theta(|E(G)|)$ time.
2. **Identifying the edge-orbit which corresponds to the set of spokes.**
   Throughout lines 6 – 9 we determine the edge-orbit which corresponds to the set of spokes. It is easy to see that this requires additional $O(|E(G)|/3)$ time.
3. **Using set U for determining parameters of a given graph.**
   The algorithm uses computed set U to determine exact isomorphism between G and an I-graph or a double generalized Petersen graph, if it exists. This procedure differentiates regarding the graph family we are considering. The related procedure Extend(G, U) is performed in $\Theta(|E(G)|)$ time.



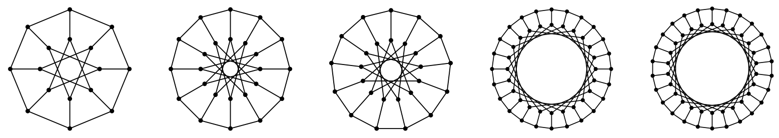
(a) *Triangular prism.*



(b) 3-*cube.*



(c) *Petersen graph, Dodecahedral, and Desargues graph.*



(d) *Möbius-Kantor graph, Nauru graph,* $G(13,5)$*,* $F_{048} \cong G(24,5)$*, and* $G(26,5)$*.*

Figure 2: All $[1,\lambda,8]$-cycle regular I-graphs. That is, (a) the graph on no 8-cycles; (b) the graph containing C_0 and C_7; (c) graphs containing C_5, C_6 and C^*; and (d) graphs containing C_3, C_4 and C^*.

3.1 Subprocedure Extend(G,U)

It is easy to see that Extend(G,U) can safely reject G if $|V(G)|$ is not divisible by 2. For this subprocedure set $n = |V(G)|/2$ and denote by H the subgraph of G induced by the vertices of U. There are two possibilities.

***H* = *G*.** In this case graph G has a constant octagon value. Since there are just ten such I-graphs checking G against them takes constant time.

***H* is of order 2*n* and is 1-regular.** Since U is a perfect matching of G the set $E(G) \setminus U$ is a collection of cycles. If G is an I-graph, then there exist positive integers i, j, l_1, l_2 with $j \le i$ such that there are j cycles of length l_1 and i cycles of length l_2. It remains to determine parameter k and check whether $G \simeq I(n,j,k)$. This procedure depends on the structure of I-graphs and the 8-cycle C^*. It is performed in $\Theta(|E(G)|)$ time (see [9] for details).

4 Conclusion

Studying the cyclic structure as described in this paper led to the construction of fast recognition algorithms for three parametric families. To the best of our knowledge, in addition to this work, such a procedure was so far only used in [10] for the family of generalized Petersen graphs. We believe that a similar approach should give interesting results for other parametric graph families of bounded degree, such as Johnson graphs, rose window graphs, Tabačjn graphs, Y-graphs, or H-graphs.

References

[1] Boben, M., Pisanski, T., and Žitnik, A. *I-graphs and the corresponding configurations. J. Comb. Des. 13*, 6 (2005), 406–424.

[2] Bonvicini, S., and Pisanski, T. Hamiltonian cycles in I–graphs. *Electronic Notes in Discrete Mathematics 40* (2013), 43–47. Combinatorics 2012.

[3] Coxeter, H. S. M. Self-dual configurations and regular graphs. *Bull. Amer. Math. Soc. 56* (1950), 413–455.

[4] Dods, M. S. *Independence number of specified I-graphs.* PhD thesis, Monterey, CA; Naval Postgraduate School, 2020.

[5] Erdős, P. Some of my favorite solved and unsolved problems in graph theory. *Quaestiones Math. 16*, 3 (1993), 333–350.

[6] Foster, R. M. *The Foster census.* Charles Babbage Research Centre, Winnipeg, MB, 1988. R. M. Foster's census of connected symmetric trivalent graphs, With a foreword by H. S. M. Coxeter, With a biographical preface by Seymour Schuster, With an introduction by I. Z. Bouwer, W. W. Chernoff, B. Monson and Z. Star, Edited and with a note by Bouwer.

[7] Horvat, B., Pisanski, T., and Žitnik, A. Isomorphism checking of I-graphs. *Graphs Combin. 28*, 6 (2012), 823–830.

[8] Klein, Z. J. *Structural properties of I-graphs: their independence numbers and Cayley graphs.* PhD thesis, Monterey, CA; Naval Postgraduate School, 2020.

[9] Klobas, N., and Krnc, M. Fast recognition of some parametric graph families. *arXiv e-prints* (Aug. 2020), arXiv:2008.08856.

[10] Krnc, M., and Wilson, R. J. Recognizing generalized Petersen graphs in linear time. *Discrete Applied Mathematics 283* (2020), 756 – 761.

[11] Mollard, M. Cycle-regular graphs. *Discrete mathematics 89*, 1 (1991), 29–41.

[12] Oliveira, A. S. S., and Vinagre, C. T. M. The spectrum of an I-graph, 2015.

[13] Petkovšek, M., and Zakrajšek, H. Enumeration of I-graphs: Burnside does it again. *Ars Math. Contemp. 2*, 2 (2009), 241–262.

[14] Watkins, M. E. A theorem on Tait colorings with an application to the generalized Petersen graphs. *J. Combinatorial Theory 6* (1969), 152–164.

Interactive Evolutionary Computation Approach to Permutation Flow Shop Scheduling Problem

Vid Keršič
University of Maribor,
Faculty of Electrical Engineering and Computer Science,
Koroška cesta 46, 2000 Maribor, Slovenia
vid.kersic@um.si

Abstract

Artificial intelligence and its subfields have become part of our everyday lives and efficiently solve many problems that are very hard for us humans. But in some tasks, these methods struggle, while we, humans, are much better solvers with our intuition. Because of that, the question arises: why not combine intelligent methods with human skills and intuition? This paper proposes an Interactive Evolutionary Computation approach to the Permutation Flow Shop Scheduling Problem by incorporating human-in-the-loop in MAX-MIN Ant System through gamification of the problem. The analysis shows that combining the evolutionary computation approach and human-in-the-loop leads to better solutions, significantly when the complexity of the problem increases.

Keywords scheduling problems, interactive evolutionary computation, ant colony optimization, metaheuristic optimization

1 Introduction

Computers have become an essential (and sometimes the only) tool to approach complex real-world problems. But many problems are still too hard to solve, even for computers. Many of these problems are NP-hard, and their solution cannot be found by using exact algorithms [1]. Thus, many of these problems are solved with approximation and different stochastic nature-inspired population-based algorithms, where suboptimal solutions are found. However, they are still considered good enough to use in practice. Some of these problems include Travelling Salesman Problem (TSP), scheduling problems, etc. This paper focuses on the Permutation Flow Shop Scheduling Problem (PFSP), one of the scheduling problems.

PFSP is the scheduling problem where there are m machines and n jobs [2]. Each job consists of m operations, where i-th operation must be executed on the i-th machine. The order of jobs must be the same on all machines, and each machine can perform only a single operation at a time. Each operation has a specified execution time, and the goal is to minimize total job execution time, called makespan. The problem was proven to be NP-hard when $m \geq 3$, which means no efficient algorithm exists [3].

Different approximation approaches were presented in the last decades and shown to be effective in finding suboptimal solutions. Firstly, different approximation algorithms were introduced to reduce time to solve the problems and find suitable solutions [4]. Afterwards, researchers applied and adjusted different optimization algorithms, especially nature-inspired and metaheuristic optimization algorithms, to improve results in shorter time execution [5].

In the last years, a new approach emerged to solve these problems. Inspired by human intuition of problem-solving, the Interactive Evolutionary Computation (IEC) approach combines metaheuristic optimization algorithms, e.g., nature-inspired algorithms, with human-in-the-loop through some process, e.g., gamification of the problem [6]. This paper proposes an IEC approach based on the MAX-MIN Ant System (MMAS), one of the Ant Colony Optimization (ACO) algorithms, to PFSP through gamification of the problem.

The structure of the paper is as follows: Section 2 provides an overview of related work. Section 3 describes the IEC approach to PFSP. Section 4 describes the experiment, shows the results, and provides an analysis of the results. Section 5 concludes the paper.

2 Related Work

Researchers started solving PFSP almost 70 years ago, when Johnson proposed the exact algorithm, but it was restricted to two machines [2]. Many exact algorithms with different strategies were proposed in the following decades, such as the branch-and-bound algorithm [7]. Despite that, the exact algorithms could not scale to bigger instances of the problem.

To overcome the NP-hard time complexity of the problem, researchers started to develop approximation and heuristic algorithms, where different approaches and heuristics were considered, such as [8]. Methods based on the Nawaz-Enscore-Ham (NEH) heuristic proved to be one of the best performers to the problem [4]. NEH-based methods consist of two steps: first, order the jobs based on some characteristic and then insert them one by one. But the approximation algorithms still have several drawbacks, such as being dependent on certain characteristics of the specific setting of the problem and still being too slow.

Metaheuristic optimization algorithms were also consid-

DOI https://doi.org/10.18690/978-961-286-516-0.8
ISBN 978-961-286-516-0

ered for PSFP, and they achieved more favorable results than approximation algorithms, especially time-wise. Many algorithms were applied to the problem, such as Simulated Annealing, Differential Evolution, and many nature-inspired algorithms, such as ACO [9, 5]. In many benchmarks, metaheuristic algorithms achieve comparable or even better results than approximation algorithms [10].

As mentioned in the introduction, researchers tried to incorporate human intuition into metaheuristic algorithms, which proved beneficial [11]. Authors included human-in-the-loop to TSP through gamification, where a user affects pheromone level in MMAS algorithm when playing a game [6]. Several other researchers also focused on gamification of different NP-hard problems like Bin-packing problem, Job Shop Scheduling, and Open Shop Scheduling [12, 13]. Based on the related work, the proposed approach incorporates human intuition to solve the PFSP problem through the gamification process where a user affects the pheromone trail in the MMAS algorithm.

3 IEC Approach to PFSP

The goal of PFSP is to find a permutation of n jobs that minimize the total makespan. Thus, the optimization task for the problem is defined as:

$$\min C(\pi) \tag{1}$$

where π is a permutation of n jobs and C is the cost function that calculates makespan.

Based on the techniques and approaches discussed in the previous section, the solution consists of three modules[1]: video game, back-end server, and MMAS algorithm. The video game was implemented in Unity, while the back-end server and MMAS algorithm, which runs on the back-end, are written in Python.

The video game contains blocks (jobs) and a board constructed from several rows (machines). The user tries to move and stack blocks together while complying with the limitations of the problem. For each game, a new instance of the MMAS algorithm starts. Each movement performed by the user is sent to the back-end server, where the MMAS algorithm runs in iterations. For each block placement, one iteration of the algorithm is launched. Pheromone of the ants is presented as 2D matrix $\mathbf{A}$ of size $N \times N$, where N is the number of jobs. Matrix's element $\mathbf{A}_{i,j}$ is the pheromone level of positioning job i on the position j. The user's placement of blocks affects the pheromone level of the algorithm; in the proposed approach, the value is multiplied by a scalar. This change of the pheromone level is the human-in-the-loop part. After each user's movement, the solution for the current problem is recommended to the user based on the outcome and current state of the MMAS algorithm (of course, users can freely decide which move they will make next). The player is also encouraged to perform well and achieve the highest score as fast as possible since the game contains a time counter. The final game score is calculated based on the combination of makespan and the time duration of the game. The algorithm of the MMAS-IEC approach is shown in Algorithm 1.

[1] Source code available at: https://github.com/Vid201/flow-shop-scheduling-iec

```
1  start the game
2  init MMAS
3  launch 1 iteration (MMAS)
4  while game not finished do
5      wait for the user to position a job (block)
6      $i \leftarrow positioned\_job$
7      $j \leftarrow position\_index$
8      $\mathbf{A}_{i,j} \leftarrow \mathbf{A}_{i,j} \times 3$
9      launch 1 iteration (MMAS)
10 end
11 return best found solution
```

Algorithm 1: Algorithm of the MMAS-IEC aporach.

4 Experiments and Results

The experiment aims to show that incorporating human-in-the-loop can positively affect the search process of metaheuristic algorithms, which means better solutions can be found in fewer iterations. Thus the analysis is mainly oriented on comparing MMAS without human-in-the-loop results with MMAS-IEC.

The operation times of the jobs are randomly generated on each new instance of the game since standard benchmark datasets are too big to be visualized in the game. Problem sizes used in the experiment are shown in Table 1. An example of the video game can be seen in Figure 1. The comparison of brute-force algorithm, MMAS algorithm without human-in-the-loop, and MMAS-IEC approach is performed. The results are also compared to the one variant of the Genetic Algorithm (GA) [14].

Table 1: Sizes of the problem in the experiment.

Number of jobs	Number of machines
5	3
10	2
10	4
12	2
12	4
12	6

In both executions of the MMAS algorithm, with and without human-in-the-loop, the parameters were set as follows: $n = 5$ (number of ants), $p_0 = 0.9$ (probability to select the job with highest pheromone), $mmr = 5$ (min-max ratio), $\rho = 0.75$ (persistence of the trail), $max\_iter = 100$ (maximum number of iterations) and $max\_stag = 5$ (maximum number of iterations of stagnation - iterations with the same solution). The parameters of MMAS were set according to the literature [5]. Readers are advised to read the referenced paper for an

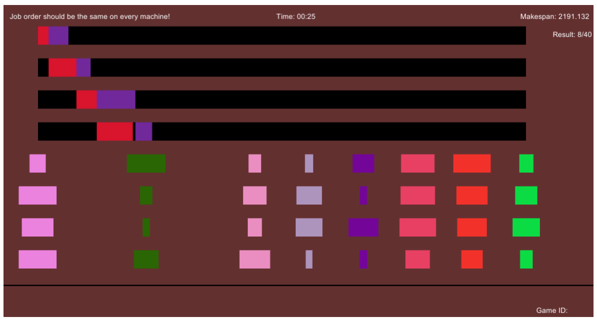


Figure 1: The gameplay of the video game.

in-depth description of the MMAS algorithm. For GA, the parameters were adapted from the used implementation [14].

The MMAS algorithm without human-in-the-loop was run 30 times to get solutions characterized by the algorithm and not randomness. In the MMAS-IEC approach, the multiplier of the pheromone was set to 3, which means element $\mathbf{A}_{i,j}$ is increased if the user places the job i on position j in the game. The size of the multiplier was chosen empirically.

To measure the performance of algorithms, the suboptimal solutions of metaheuristic algorithms were compared to optimal solutions by calculating relative percentage deviation (RPD), which tells how much suboptimal solutions are worse than the optimal solution. RPD was calculated according to the following equation:

$$RPD = \frac{Suboptimal_{sol} - Optimal_{sol}}{Optimal_{sol}} \times 100 \qquad (2)$$

The number of iterations to find the best solution was also compared since time execution was much longer in the interactive approach due to playing the game.

4.1 Analysis and Discussion

The video game and other algorithms were launched five times for each problem size, and the average/mean makespan with standard deviation was calculated. The brute force algorithm was run only on smaller problem sizes since its execution time was too long for the other instances. Where brute force results were available, RPD was also calculated. Results for the experiment are shown in Table 2.

While in the smallest instances of the problem, the MMAS algorithm outperformed the MMAS-IEC approach, the latter achieved better results in all the other problem sizes. This implies that when the search space becomes enormous and complex, human intuition positively affects the algorithm and leads to a better solution. Since the user affected the pheromone level and directed the algorithm towards better solutions, the average number of iterations was less than with only the MMAS algorithm. It must be noted that the best solution in the MMAS-IEC is usually not the same as the end state of the blocks in the user's game. The user only affected the pheromone level of the job on the selected position, and the game's order does not directly change the solution. The best solution is still the solution found by the MMAS algorithm, while the user only changed the algorithm's internal state, i.e., pheromone matrix. Figure 2 shows that when the problem sizes increase, MMAS-IEC tends to find solutions with a lower makespan.

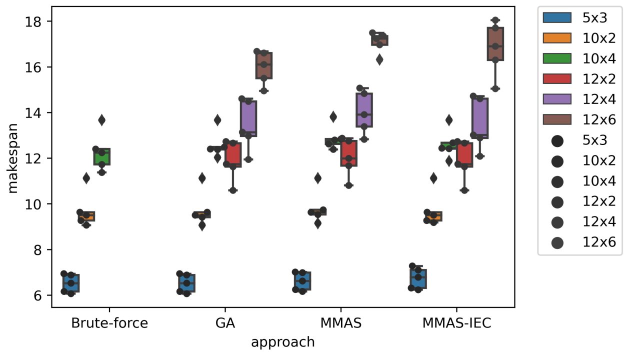


Figure 2: Box plots of makespan for different sizes of the problem with all four approaches. Colors represent different sizes of the problem, while dots show makespans for all five settings per each problem size for each approach.

Results of the MMAS-IEC approach are very dependent on the way the user plays the game. In the first played games, the results of the MMAS-IEC were not better

Table 2: Results of the experiment, where N is number of jobs, M is number of machines, $\overline{M}$ is average makespan, σ is standard deviation, $\overline{IT}$ is average number of iterations for best solution and $\overline{RPD}$ is average relative percentage deviation.

Problem sizes		Approach													
N	M	Brute force		MMAS				GA				MMAS-IEC			
		$\overline{M}$	σ	$\overline{M}$	σ	$\overline{IT}$	$\overline{RPD}$	$\overline{M}$	σ	$\overline{IT}$	$\overline{RPD}$	$\overline{M}$	σ	$\overline{IT}$	$\overline{RPD}$
5	3	6.514	0.402	6.605	0.399	31.0	1.396	6.515	0.402	25.8	0.015	6.742	0.463	3.6	3.500
10	2	9.719	0.814	9.834	0.753	27.8	1.183	9.749	0.797	45.0	0.309	9.742	0.792	2.8	0.236
10	4	12.281	0.876	12.890	0.543	28.4	4.958	12.594	0.625	61.2	2.549	12.614	0.657	5.2	2.711
12	2	/	/	12.019	0.845	26.0	/	11.870	0.877	42.4	/	11.869	0.877	2.6	/
12	4	/	/	14.005	0.949	29.8	/	13.430	1.118	49.2	/	13.463	1.152	5.4	/
12	6	/	/	17.075	0.464	19.8	/	15.970	0.739	84.2	/	16.801	1.195	8.2	/

or were even worse than MMAS alone. But after some games, strategies for the game come into mind, and results get better, which means the makespan decreases. One of the strategies that tend to work well is to position the job with increasing operations times by the machine number in the first place and then putting jobs with as little space between operations as possible on all machines. While the number of iterations is much less in the MMAS-IEC approach, time to find the best solution depends on how fast the user plays the game, e.g., MMAC alone can find the suboptimal solution in 1 second, while the user cannot play the game so fast. For bigger instances, around 10 seconds are needed to find the best suboptimal solution (the game does not have to be played to the end to find the best solution).

Comparing to the GA, both MMAS and MMAS-IEC approaches achieve worse results, except when $N = 10$ and $M = 2$, where the MMAS-IEC approach achieves the best ones. According to the obtained results, combining an interactive process, i.e., gamification, and GA, looks like a promising direction to explore.

The open problem for this gamification process is how to scale the game to bigger problem sizes since the game can become too complex, and it is also hard to visualize blocks and boards. The experiments were successfully conducted on the problems with up to 6 machines and 12 jobs, but scaling the game to more machines and jobs is open for further research.

5 Conclusion

This work shows how human intuition for problem-solving can be incorporated into metaheuristics algorithms for NP-hard problem PFSP. MMAS-IEC approach tends to find better solutions than fully autonomous metaheuristic algorithm MMAS, especially in bigger settings of the problem. Solutions are also found in a fewer number of iterations. But still, there are some open questions to the proposed approach, as bigger instances are hard to visualize, and playing the game usually takes more time than the MMAS algorithm alone. Both of these drawbacks are good starting points for further research.

Further research could be dedicated to selecting a metaheuristic algorithm since this study only accounts for the MMAS algorithm for PSFP. At the same time, several other algorithms proved to be suitable for this problem, e.g., the GA from the analysis. The research goal was to use the same metaheuristic algorithm with and without human interaction and not to compare different metaheuristic algorithms. On the MMAS algorithm itself, other techniques to change the pheromone level could be explored. Various strategies to encourage the player to perform well could also be introduced, e.g., online scoreboards or rewards, which could increase the player's performance with a positive loop. It would also be interesting to compare and try the MMAS-IEC approach on one of the benchmarks for PSFP, e.g., Taillard's benchmark problems [15].

References

[1] D. E. Knuth. Postscript about np-hard problems. *SIGACT News*, 6(2):15–16, Apr. 1974. ISSN: 0163-5700. DOI: `10.1145/1008304.1008305`.

[2] S. M. Johnson. Optimal two- and three-stage production schedules with setup times included. *Naval Research Logistics Quarterly*, 1(1):61–68, 1954.

[3] M. R. Garey, D. S. Johnson, and R. Sethi. The complexity of flowshop and jobshop scheduling. *Mathematics of operations research*, 1(2):117–129, 1976.

[4] M. Nawaz, E. E. Enscore Jr, and I. Ham. A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega*, 11(1):91–95, 1983.

[5] T. Stützle et al. An ant approach to the flow shop problem. In *Proceedings of the 6th European Congress on Intelligent Techniques & Soft Computing (EUFIT'98)*, volume 3, pages 1560–1564, 1998.

[6] A. Holzinger, M. Plass, M. Kickmeier-Rust, K. Holzinger, G. C. Crişan, C.-M. Pintea, and V. Palade. Interactive machine learning: experimental evidence for the human in the algorithmic loop. *Applied Intelligence*, 49(7):2401–2414, 2019.

[7] E. Ignall and L. Schrage. Application of the branch and bound technique to some flow-shop scheduling problems. *Operations research*, 13(3):400–412, 1965.

[8] S. Suliman. A two-phase heuristic approach to the permutation flow-shop scheduling problem. *International Journal of production economics*, 64(1-3):143–152, 2000.

[9] I. Osman and C. Potts. Simulated annealing for permutation flow-shop scheduling. *Omega*, 17(6):551–557, 1989. ISSN: 0305-0483.

[10] E. Vallada, R. Ruiz, and J. M. Framinan. New hard benchmark for flowshop scheduling problems minimising makespan. *European Journal of Operational Research*, 240(3):666–677, 2015. ISSN: 0377-2217.

[11] A. Holzinger. Interactive machine learning for health informatics: when do we need the human-in-the-loop? *Brain Informatics*, 3(2):119–131, 2016.

[12] N. D. Ross, M. B. Johns, E. C. Keedwell, and D. A. Savic. Human-evolutionary problem solving through gamification of a bin-packing problem. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 1465–1473, 2019.

[13] M. Vargas-Santiago, R. Monroy, J. E. Ramirez-Marquez, C. Zhang, D. A. Leon-Velasco, and H. Zhu. Complementing solutions to optimization problems via crowdsourcing on video game plays. *Applied Sciences*, 10(23):8410, 2020.

[14] Suyunu. Github - suyunu/flow-shop-scheduling: genetic algorithm for flow shop scheduling. URL: `https://github.com/suyunu/Flow-Shop-Scheduling` (visited on 07/23/2021).

[15] E. Taillard. Benchmarks for basic scheduling problems. *european journal of operational research*, 64(2):278–285, 1993.

Towards Representative Web Performance Measurements with Google Lighthouse

Tjaša Heričko
University of Maribor,
Faculty of Electrical
Engineering and Computer Science,
Koroška cesta 46, 2000 Maribor, Slovenia
tjasa.hericko@um.si

Boštjan Šumak
University of Maribor,
Faculty of Electrical
Engineering and Computer Science,
Koroška cesta 46, 2000 Maribor, Slovenia
bostjan.sumak@um.si

Saša Brdnik
University of Maribor,
Faculty of Electrical
Engineering and Computer Science,
Koroška cesta 46, 2000 Maribor, Slovenia
sasa.brdnik@um.si

Abstract

Web performance testing with tools such as Google Lighthouse is a common task in software practice and research. However, variability in time-based performance measurement results is observed quickly when using the tool, even if the website has not changed. This can occur due to variability in the network, web, and client devices. In this paper, we investigated how this challenge was addressed in the existing literature. Furthermore, an experiment was conducted, highlighting how unrepresentative measurements can result from single runs; thus, researchers and practitioners are advised to run performance tests multiple times and use an aggregation value. Based on the empirical results, 5 consecutive runs using a median to aggregate results reduce variability greatly, and can be performed in a reasonable time. The study's findings alert to potential pitfalls when using single run-based measurement results and serve as guidelines for future use of the tool.

Keywords websites, web performance, performance testing, performance variability, tool, Google Lighthouse

1 Introduction

In software engineering, performance tests are often conducted by software researchers and practitioners to audit a website's quality. The former commonly use web performance measurements to assess web performance on the observed websites [12, 14, 15], to investigate factors (positively or negatively) affecting performance [4, 6, 13], and to improve performance testing [10], while the latter use performance measurements for improving a website's quality to provide a better overall user experience, as web performance influences website traffic, user attrition, user engagement, online revenue, and even rankings in search results greatly [2, 16, 17].

Performance testing can be conducted using various tools, among which Google Lighthouse has gained increasing attention in recent years. It is an open-source tool, providing audits for performance, as well as for accessibility, search engine optimization, and progressive web apps, with indicators on how to improve these aspects of websites if needed [8]. However, when dealing with time-based measurements, the results of such testing can often be inconsistent, as several factors can interfere with the measures and may introduce fluctuations, even if the website has not changed. Most commonly, results tend to vary due to variability in the network, web server, client hardware, and client resource contention [7]. Lighthouse addresses variability by providing vague strategies and recommendations on how to reduce them, though results can still vary. Besides isolating external factors, e.g., using a dedicated device for testing, using a local deployment or a machine on the same network, the most straightforward strategy is to run Lighthouse multiple times and use aggregate values instead of single tests [7].

The research objectives of this paper are: (i) To study how the research community has addressed the challenge of variability in performance measurements when using the tool; and (ii) To demonstrate the strategy of performing multiple runs empirically with their aggregation into a single-value result. To achieve these objectives, we performed a literature review and conducted an experiment.

Our work is broadly related to previous research providing a better understanding and managing of variations in measurements, testing, and benchmarking for timebased performance measurements [3, 5, 10, 11]. These studies are focused primarily on suggesting recommendations for robust testing in the presence of environmental fluctuations, and, as such, are quite different in aim from ours, which is to gain insight into how a specific tool – Google Lighthouse – is used in research for web performance measurement, further investigated with empirical research alerting software researchers and practitioners to potential pitfalls in the future use of the tool.

DOI https://doi.org/10.18690/978-961-286-516-0.9
ISBN 978-961-286-516-0

The contributions of the paper are: (i) Presenting an overview of existing studies using Lighthouse for measuring performance, with an emphasis on how the tool is used, what measuring strategies are employed, and how the authors addressed possible inconsistencies in results; (ii) Providing analysis of the effects of repeating performance measurements to prevent single run‘s outliers; (iii) Highlighting potential pitfalls for research and practice using single run-based results provided by Lighthouse; and (iv) Serving as a base for research studies on mitigating unrepresentative web performance measurements.

2 Literature review

A literature review was performed to find existing research utilizing Lighthouse as the tool for estimating web performance. A full-text search was conducted using the search string *»Google Lighthouse«* in the following digital libraries: ACM Digital Library[1], Google Scholar[2], IEEE Xplore[3], and Web of Science[4]. The search was carried out on July 28, 2021, and altogether 134 studies were retrieved from the search. Inclusion and exclusion criteria guided the study selection process. Only journal and conference papers were considered. Materials not accessible in English were excluded. Any research that only described Lighthouse theoretically was excluded, and all papers where Lighthouse was not used for performance measurements were excluded as well. After the review process, 8 primary studies were selected.

The list of primary studies is available in Table 1. All primary studies were published in conference proceedings in recent years, in 2018 or later. From the performance measurements made with Lighthouse, primary studies used the Performance Score (S1-S3, S6) most commonly, a single-value indicator of websites' overall performance, for their further analysis. The following more specific time metrics were also used commonly: Speed Index (S2, S4, S7, S8), First Meaningful Paint (S1, S2, S4, S7) and Estimated Input Latency (S1, S2, S7). Researchers observed between 1 and 21 websites in each study. Less than half of the primary studies (S1, S4, S7) have noted some variance between runs when auditing the same website due to uncontrollable variables, and employed some strategies to mitigate this problem. In two studies (S4, S7), the authors repeated runs consequently, while in one study (S1), researchers ran performance audits multiple times trough the day. The number of runs varied from 5 to 100. Two studies (S1, S4) then used mean for aggregating multiple runs into a single value, while one study (S7) used median.

Table 1: A list of primary studies selected in the literature review with their performance audit strategies.

ID	Year	Performance audit strategy	Ref
S1	2018	5 repetitions trough the day	[6]
S2	2019	1 run	[12]
S3	2019	1 run	[15]
S4	2019	100 consecutive repetitions	[13]
S5	2019	1 run	[10]
S6	2020	1 run	[4]
S7	2020	30 consecutive repetitions	[14]
S8	2021	1 run	[18]

3 Experiment

An experiment was performed to demonstrate further how single performance audits can be unrepresentative in some scenarios, and investigate how the number of runs affects variability. In the experiment, 10 real websites were used, selected randomly from the Alexa Top 500 list, which includes top-ranked websites on the web [1]. The selected websites were: AliExpress[5], Amazon India[6], Bola[7], Freepik[8], IKEA[9], Mercari[10], Shopify[11], Unspash[12], Wix[13], and Zendesk[14], further referred to as *W1*, *W2*, *W3*, *W4*, *W5*, *W6*, *W7*, *W8*, *W9*, and *W10*, respectively. The selected websites ranged from simple static presentation websites to dynamic websites, i.e., e-commerce websites and news sites.

All websites were audited with Google Lighthouse v7.3.0 on a device with macOS 10.15.7 using Headless Chrome. A desktop device was emulated using a broadband network connection. For each website, performance audits were performed 4 times, each time with an increasing number of repeated independent runs (N=1,5,10,100). During the auditing process, external factors were isolated as much as possible. The experiment was conducted on July 29, 2021 between 4:42 and 8:37 PM (GMT+2).

From the collected results, we used the Performance Score for analysis, as this value captures the overall web performance of a website. It is calculated as a weighted average of six metric scores, each metric representing some aspect of a website's performance. The Performance Score can have the following values: *Poor* (0-50), *Needs improvement* (50-89) and *Good* (90-100) [9].

Data analysis was conducted using IBM SPSS Statistics v27. Descriptive statistics were used to present the characteristics of sets of data collected with a different number of consecutive runs, including a description and spread of the data in each set. Mood's median test was performed to estimate if the medians of data sets from different runs on the same website were equal.

[1]https://dl.acm.org/
[2]https://scholar.google.com/
[3]https://ieeexplore.ieee.org/
[4]https://apps.webofknowledge.com/

[5]https://www.aliexpress.com/
[6]https://www.amazon.in/
[7]https://www.bola.com/
[8]https://www.freepik.com/
[9]https://www.ikea.com/
[10]https://www.mercari.com/
[11]https://www.shopify.com/
[12]https://unsplash.com/
[13]https://www.wix.com/
[14]https://www.zendesk.com/

4 Results and Discussion

The distribution of Performance Scores of each website for N=100 is presented with boxplots in Figure 1. It can easily be observed that, for almost all websites (except for *W4*), some performance measurements occurred that were not a typical representative of a website's performance. Suppose one of these outlier measurements was the only assessment run, this can lead to an unrepresentative result. Consequently, wrongful conclusions and decisions can be made, e.g., a developer may think a change he implemented into a code recently made performance worse, yet, instead, this occurred due to fluctuation in the network, web, or client device. An interesting observation is that, due to variability in the measurement results, a website can be interpreted in a different score group, e.g., *W1*, *W2*, *W4*, and *W9* results are dispersed between score groups *Good* and *Needs improvement*, and *W3* between score groups *Poor* and *Needs improvement*.

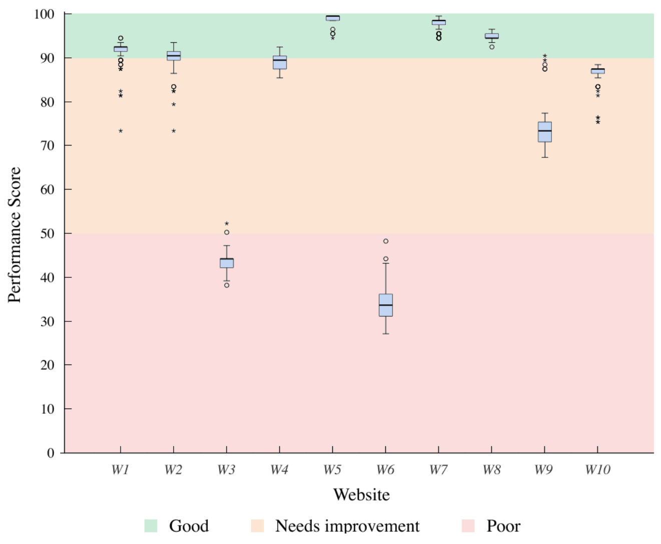


Figure 1: Data distribution of Performance Scores (N=100).

Detailed results for all sets of data are presented in Table 2, providing insight into how the data are spread, how much repeating the test reduces variability and how results stabilize as the number of tests run increases. These results further illustrate the differences between single and multiple runs, which can provide a more reliable estimate of a website's performance; therefore, providing a rationale why addressing intrinsic fluctuations when dealing with time-based metrics should be considered, and why a single run can (in some cases) not be representative enough to provide reliable measurements. We argue that the use of a median value for aggregation is preferred over other measures of central tendency to minimize the impact of outliers.

Mood's median test, performed for N=5, N=10, and N=100, showed that the medians of the Performance Score were the same across all three categories of runs for all websites, except *W5*, where the test could not be performed. These results, presented in Table 3, indicate that 5 runs in comparison to 10 and 100 runs are sufficient to eliminate possible outliers while still performing web performance testing in a reasonable time.

Table 2: Performance scores and their statistics across different numbers of runs.

	Statistics	**N=1**	**N=5**	**N=10**	**N=100**
W1	Mean	87	91	91.2	90.9
	Median	87	92	92	92
	Range	/	6	6	21
	SD	/	2.3	1.9	2.9
W2	Mean	86	86	88.3	89.5
	Median	86	90	90	90
	Range	/	18	20	20
	SD	/	7.5	5.7	2.8
W3	Mean	40	42.8	44.3	43.3
	Median	40	43	44	44
	Range	/	5	10	14
	SD	/	1.9	2.8	1.9
W4	Mean	89	88.4	88.3	88.6
	Median	89	89	88	89
	Range	/	5	5	7
	SD	/	1.9	1.5	1.5
W5	Mean	99	98.6	98.5	98.6
	Median	99	99	98.5	99
	Range	/	1	1	5
	SD	/	0.5	0.5	0.9
W6	Mean	28	35	32.9	33.9
	Median	28	34	31	33.5
	Range	/	15	16	21
	SD	/	7.3	6.1	3.9
W7	Mean	98	98	97.6	97.3
	Median	98	98	98	98
	Range	/	0	2	5
	SD	/	0	0.7	1.2
W8	Mean	94	94.4	94.4	94.5
	Median	94	94	94	94
	Range	/	1	1	4
	SD	/	0	0.5	0.6
W9	Mean	77	72.4	72.1	73.3
	Median	77	72	72	73
	Range	/	7	8	23
	SD	/	2.9	2.5	4.2
W10	Mean	75	83.4	85.3	86.1
	Median	75	86	87	87
	Range	/	12	13	13
	SD	/	4.9	3.9	2.5

5 Conclusion

Several strategies can be employed to reduce random noise, measurement bias and errors when using Lighthouse for web performance measurements. In the paper, we performed a literature review in which we selected studies using Lighthouse for estimating web performance. The results show that more than half of the primary studies did not employ any specific strategy to address variability in web performance measurements. Others use a reasonably straightforward approach to repeat the Lighthouse audit multiple times and summarize repeated runs using a mean or median. However, a large discrepancy was noticed in these works in the number of runs and measures of central tendency used to aggregate multiple

Table 3: Mood's median test by website, comparing groups of N=5, N=10, and N=100.

	W1	*W2*	*W3*	*W4*	*W5*	*W6*	*W7*	*W8*	*W9*	*W10*
Asymp.Sig	0.996	0.723	0.137	0.557	*	0.744	0.927	0.879	0.211	0.758

*All values are less than or equal to the median. Mood's median test could not be performed.

runs into a single-value result. Thus, we investigated this further empirically by conducting an experiment on real popular websites, to demonstrate how the number of runs affects variability and prevents single-run outliers. With this, we highlighted how measurement results from single runs could be misleading and unrepresentative; therefore, we recommend for research and practice to run performance tests multiple times and use an aggregation value. Based on our results, performing performance audits 5 times reduces variability in results greatly in a reasonable time. Our study provides a base for future research studies addressing outliers in web performance testing, and for guidelines for future studies on how to perform representative web performance measurements with Lighthouse.

Acknowledgment

The authors acknowledge the financial support from the Slovenian Research Agency (Research Core Funding No. P2-0057).

References

[1] Alexa Internet, Inc. The top 500 sites on the web, 2021.

[2] Arapakis, I., Bai, X., and Cambazoglu, B. B. Impact of response latency on user behavior in web search. In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval* (New York, 2014), SIGIR '14, ACM, pp. 103–112.

[3] Beyer, D., Löwe, S., and Wendler, P. Reliable benchmarking: Requirements and solutions. *International Journal on Software Tools for Technology Transfer 21*, 1 (2019), 1–29.

[4] Chan-Jong-Chu, K., Islam, T., Exposito, M. M., Sheombar, S., Valladares, C., Philippot, O., Grua, E. M., and Malavolta, I. Investigating the correlation between performance scores and energy consumption of mobile web apps. In *Proceedings of the Evaluation and Assessment in Software Engineering* (New York, 2020), EASE '20, ACM, pp. 190–199.

[5] Chen, J., and Revels, J. Robust benchmarking in noisy environments, 2016.

[6] Gambhir, A., and Raj, G. Analysis of cache in service worker and performance scoring of progressive web application. In *2018 International Conference on Advances in Computing and Communication Engineering (ICACCE)* (2018).

[7] Google Developers. Variability, 2019.

[8] Google Developers. Lighthouse, 2021.

[9] Google Developers. Lighthouse performance scoring, 2021.

[10] Johnston, O., Jarman, D., Berry, J., Zhou, Z. Q., and Chen, T. Y. Metamorphic relations for detection of performance anomalies. In *2019 IEEE/ACM 4th International Workshop on Metamorphic Testing (MET)* (2019), pp. 63–69.

[11] Kalibera, T., and Jones, R. Rigorous benchmarking in reasonable time. In *Proceedings of the 2013 International Symposium on Memory Management* (New York, 2013), ISMM '13, ACM, pp. 63–74.

[12] Nurshuhada, A., Yusop, R. O. M., Azmi, A., Ismail, S. A., Sarkan, H. M., and Kama, N. Enhancing performance aspect in usability guidelines for mobile web application. In *2019 6th International Conference on Research and Innovation in Information Systems (ICRIIS)* (2019), pp. 1–6.

[13] Rahman, S., and Wittie, M. P. MR-DNS: Multi-resolution Domain Name System. In *Internet and Distributed Computing Systems* (Cham, 2019), R. Montella, A. Ciaramella, G. Fortino, A. Guerrieri, and A. Liotta, Eds., Springer International Publishing, pp. 191–202.

[14] Riet, J. v., Paganelli, F., and Malavolta, I. From 6.2 to 0.15 seconds – an industrial case study on mobile web performance. In *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)* (2020), pp. 746–755.

[15] Rojas-Mora, J., Lincolao-Venegas, I., and Schneeberger-Leon, F. S3e2: a web-based gis for the visualization and analysis of socioeconomic segregation in chile's elementary education system. In *Proceedings of the 1st International Conference on Geospatial Information Sciences* (2019), O. S. Siordia, J. L. S. Cardenas, A. Molina-Villegas, G. Hernandez, P. Lopez-Ramirez, R. Tapia-McClung, K. G. Zuccolotto, and M. C. Colunga, Eds., vol. 13 of *Kalpa Publications in Computing*, EasyChair, pp. 12–20.

[16] Shivakumar, S. K. *Modern Web Performance Optimization.* Apress, Berkeley, Ca, 2020.

[17] Szalek, K., and Borzemski, L. Conversion Rate Gain with Web Performance Optimization. A Case Study. In *Information Systems Architecture and Technology: Proceedings of 39th International Conference on Information Systems Architecture and Technology – ISAT 2018* (Cham, 2019), Springer, pp. 312–323.

[18] Yu, A., and Benson, T. A. Dissecting performance of production quic. In *Proceedings of the Web Conference 2021* (New York, 2021), WWW '21, ACM, pp. 1157–1168.

Transformer-based Sarcasm Detection in English and Slovene Language

Matic Rašl
University of Maribor,
Faculty of Electrical
Engineering and Computer Science,
Koroška cesta 46, 2000 Maribor, Slovenia
matic.rasl@student.um.si

Mitja Žalik
University of Maribor,
Faculty of Electrical
Engineering and Computer Science,
Koroška cesta 46, 2000 Maribor, Slovenia
mitja.zalik1@student.um.si

Vid Keršič
University of Maribor,
Faculty of Electrical
Engineering and Computer Science,
Koroška cesta 46, 2000 Maribor, Slovenia
vid.kersic@um.si

Abstract

Sarcasm detection is an important problem in the field of natural language processing. In this paper, we compare performances of the three neural networks for sarcasm detection on English and Slovene datasets. Each network is based on a different transformer model: RoBERTa, DistilBert, and DistilBert – multilingual. In addition to the existing Twitter-based English dataset, we also created the Slovene dataset using the same approach. An F1 score of 0.72 and 0.88 was achieved in the English and Slovene dataset, respectively.

Keywords natural language processing, sarcasm detection, transformers, RoBERTa, DistilBert

1 Introduction

Language is the essential tool for communication in real life and online in the digital world. With the fast growth of the internet in the last two decades, an enormous amount of text data is available to everyone, which is one of the main reasons natural language processing (NLP) has become one of the fastest-growing fields in computer science and artificial intelligence. While the most commonly used NLP application is text translation, many other applications are being researched and applied, e.g., text summarization, emotion recognition, sarcasm, and irony detection [1]. In this paper, we focus on the sarcasm detection problem.

Sarcasm detection is defined as a binary classification problem, where the goal is to detect if the given text is sarcastic [2]. The most common places to find sarcastic comments are social media platforms, e.g., Twitter, where people often express their opinions and views on different topics. While in some examples, e.g., *"I work 40 hours a week for us to be this poor"*, it is easy to spot, sometimes, e.g., *"Great, that's just what I needed!"* is harder to perceive at first sight. Detection of sarcasm is essential because not understanding and detecting it can lead to substantial miscommunication errors and disagreements. Automatic sarcasm detection is also crucial in other NLP problems, such as sentiment analysis, where undetected sarcasm can negatively affect an analysis. Therefore, there is a need for automatic detection of sarcastic comments and text.

This paper compares performances of three neural networks for sarcasm detection on English and Slovenene datasets. Each neural network is based on a transformer model. In the following sections, we overview the related work, describe the used datasets, present the experiment, analyze the results, and conclude the paper emphasizing future work.

2 Related Work

Automatic sarcasm detection dates back to 2006 [3], but it has gotten momentum in the past few years with advancements in the fields of neural networks and NLP. In general, sarcasm can be detected in three different ways [2]. Rule-based approaches use specific evidence, such as words or phrases, for identification. Such techniques were often used in earlier systems, such as [4]. Statistical approaches either use text features or learning algorithms to find sarcasm. Statistical methods were used in works, such as [5], where combinations of positive verbs and negative situation phrases were used as classification features. The most common approach today is by using deep learning techniques. For example, in [6], the model can learn user-specific context and thus achieve better results than previous state-of-the-art models.

Significant advancements in NLP tasks were achieved with transformers. They are a new form of neural network that does not use convolution and recursion. Instead, they use attention to find correlations between words in the text. Transformers can process text in parallel, allowing much faster learning than sequential methods [7]. They also achieve better results than previous methods.

DOI https://doi.org/10.18690/978-961-286-516-0.10
ISBN 978-961-286-516-0

With the increasing number of learning parameters, neural networks need a larger training dataset to prevent overfitting. While building large labeled datasets can be demanding, it is easy to construct large unlabelled corpora. Therefore, large models can be trained on unlabelled text data to create a good language model, i.e., expressive word embeddings. Afterwards, these representations can be used for different NLP-related tasks [5]. The mainstream architecture of the pre-trained models is Bidirectional Encoder Representations from Transformers (BERT). The initial model was pre-trained on BooksCorpus and English Wikipedia, which advanced state-of-the-art for eleven NLP tasks [8]. Nowadays, many BERT-based architectures exist. For example, RoBERTa (A Robustly Optimized BERT Pre-training Approach) [9] optimizes the way of masking tokens and thus improving the performance of the model. Another common architecture is DistilBert [10], which has reduced the number of training parameters. That makes its training 60 % faster while retaining 97 % of BERTs language understanding capabilities.

BERT has been widely and successfully utilized for sarcasm detection [11]. In [12], the accuracy is even more improved by also considering the context of sarcastic comments. The authors in [13] use RoBERTa to detect sarcasm with even higher accuracy. Although BERT-based architectures are very successful, their pre-training still has some drawbacks. Sarcasm is present primarily in informal communication (e.g., social networks such as Reddit, Twitter, etc.), which was not part of the training set. Therefore, in [14] BERT was outperformed by the context-independent GloVe embeddings model, which was pre-trained on Twitter data.

3 Datasets

Constructing a dataset for the sarcasm detection problem is not a straightforward task since the perception of sarcasm is difficult even for people. A general approach to dataset creation is to scrap the data from different social media platforms, e.g., Twitter, Reddit, and use user-specified labels, i.e., hashtags on Twitter and */s* on Reddit [11, 15, 16]. But this approach has several drawbacks, like users not annotating sarcasm with tags or misusing labels to express their opinion better. The Headlines dataset was introduced to solve the mentioned problem. The dataset contains headlines from two news websites: one, where real-world events are reported, and the other with sarcastic descriptions of events, including sarcastic headlines [17]. The third common way is to manually label data, but this is time-consuming and still requires the annotator with a good sense of sarcasm.

Since no dataset for sarcasm detection in the Slovenian language exist, and manual labeling is time-consuming, we created the Slovene dataset with the user-specified labels. As a knowledge base for our task, tweets (i.e., posts on Twitter) were selected. Tweets, annotated by users with specific hashtags (e.g., #sarcasm, #sarkazem), were considered sarcastic (i.e., positive) examples, while other tweets were non-sarcastic (i.e., negative) examples. For the English dataset, we selected the one from the 2nd Workshop on Figurative Language Processing [11] because it was constructed in the same way as the Slovene dataset. Before training, datasets were split into the training and the test sets, as shown in Table 1.

Table 1: Train-Test split.

Set	**English**	**Slovene**
Train	5000	759
Test	1800	272

4 Method

Although transformers can be fine-tuned to specialize in a specific task, the process takes a long time on common hardware. However, as shown in [13], fine-tuning can be avoided by utilizing other networks to find correlations in transformer embeddings. Since the transformer's weights are not changing, its output can be calculated only once and then saved before learning the second part of the network. This approach significantly improves learning times.

For the experiment, we implemented the neural network model similar to the one used in [13]. As some details about the network were missing in the mentioned paper, we also relied on implementation in [18]. The architecture of the neural network is shown in Figure 1. During the experiment, three different transformers were explored, RoBERTa [9], pre-trained on English dataset, and two DistilBert [10] transformers, one pre-trained on the English language and the other one on multiple languages (DistilBert mult). DistilBert transformers are smaller than RoBERTa (66 M vs. 125 M training parameters), which translates to significant speedup embedding generation time.

As mentioned before, tokenized inputs were transformed to embeddings at the beginning of the training. Embeddings were saved to the *Transformer output cache.* Then they were used as input to the Bidirectional LSTM layer, whose outputs were concatenated with original embeddings before the pooling layer. Before flattening the data, 1D spatial dropout was applied. After dense and another dropout layer, a dense layer with softmax activation was applied.

For the training, the Google Colab [19] environment with Google TPU was used. Neural networks were trained for 25 epochs with a 10 % validation split. In the end, weights of epoch with the smallest validation loss were restored.

5 Experiments and Results

The accuracy of the models was tested on the test datasets with the embedding of length 20. Additionally, we tested the model using RoBERTa transformer with the embeddings of length 100 to find out how embeddings length affects the results. However, since the results with larger embeddings were similar to those obtained with smaller ones, we did not train DisitlBert based models on larger embeddings due to the hardware limitations. Results are shown in Table 2.

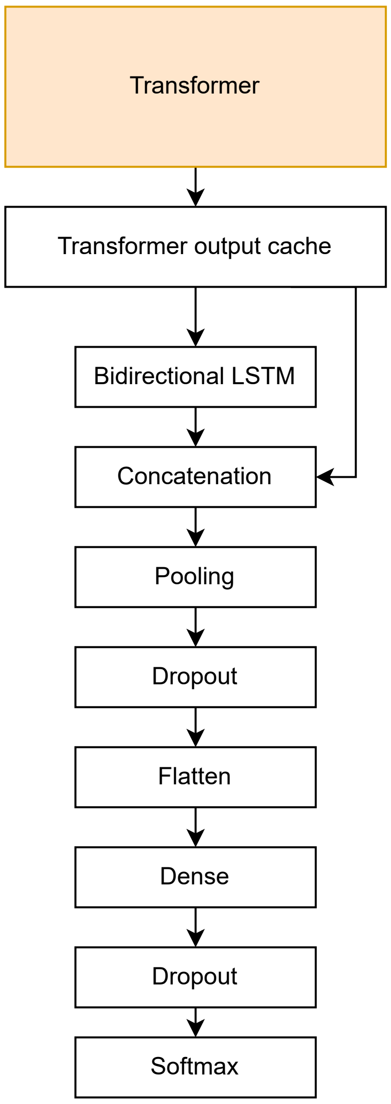


Figure 1: The architecture of the neural network.

For all tested models, the results on the Slovene data were significantly better than on the English data (ranging from 11 % to 18 % improvement). This means that in the used Slovene dataset, sarcasm was more clearly expressed. The best results on both datasets were achieved using the RoBERTa transformer with an embedding length of 100. Even when using shorter embeddings, the RoBERTa transformer performed the best. However, the difference between embeddings of length 100 and 20 was small (only 2 % difference in F1 score). Additionally, the difference between using RoBERTa and DistilBERT transformer is also relatively small (3 % to 6 % difference in F1 score), which implies that the usage of DistilBERT can be a good alternative to RoBERTa on low-cost hardware. When using a multilingual transformer, the results on the English dataset were close to the English-only transformer. However, on the Slovene dataset, the multilingual dataset provided slightly better results.

In [13], the performance of the RCNN-RoBERTa model

Table 2: Results of the experiment. The first three columns contain *transformer's name* (**M**), *used dataset* (**L**), and *embeddings length* (**EL**). The dataset is denoted by its language (*slo* for Slovene and *en* for English). In the last four columns, *accuracy* (**A**), *precision* (**P**), *recall* (**R**), and *F1 norm* (**F1**) are presented. The results are rounded to two decimal places.

M	**L**	**EL**	**A**	**P**	**R**	**F1**
RoBERTa	en	100	0.70	0.67	0.78	0.72
RoBERTa	en	20	0.66	0.63	0.79	0.70
DistilBert	en	20	0.63	0.60	0.74	0.67
DistilBert - mult	en	20	0.61	0.58	0.80	0.67
RoBERTa	slo	100	0.83	0.82	0.95	0.88
RoBERTa	slo	20	0.81	0.83	0.89	0.86
DistilBert	slo	20	0.74	0.83	0.78	0.80
DistilBert - mult	slo	20	0.79	0.89	0.78	0.83

was measured on various datasets. The F1 score was between 78 % and 90 %, which is considerably better than the results in English, but comparable to the Slovene dataset. However, since different datasets were used in the study, the results are hard to compare. The same English dataset as applied here was used in [11], where participants presented 13 different solutions, ranging between an F1 score of 0.58 and 0.83 Only three solutions were better than the F1 score of 0.72 , which we achieved with RoBERTa transformer and embeddings length of 100

Dataset construction is one of the most critical parts of the experiment since it provides the knowledge base for the transformer models. According to the obtained results, models were able to detect sarcasm in the given examples, despite several drawbacks explained in section 3. The used approach is good enough for uncomplicated use cases where sarcasm is meant to be detected since users also annotate messages with #sarcasm. But for more complicated use cases with complex and more challenging examples, alternative methods to the dataset construction should also be explored.

6 Conclusion

In this paper, we compare how the utilization of different transformers combined with the BiLSTM model affects the accuracy of sarcasm prediction. RoBERTa, English-based DistilBERT, and multilingual DistilBERT transformers were used in the experiment. All three transformers were combined with the same BiLSTM model and trained on English and Slovene datasets. Afterwards, the accuracy of the models was obtained using test datasets in English and Slovene language. In the best case, F1 scores of 0.72 and 0.88 were achieved on English and Slovene datasets, respectively.

In the future, more work could be done on dataset creation. Different dataset construction approaches, as described in section 3, can be explored and adjusted for the Slovene language. Furthermore, current datasets can be expanded by adding more Tweets (especially Slovene) or data from different sources, e.g., Reddit. Another interesting direction for future work is exploring transfer learning to reuse models on different languages, e.g., languages similar to Slovene.

References

[1] G. G. Chowdhury. Natural language processing. *Annual Review of Information Science and Technology*, 37(1):51–89, 2003. DOI: 10.1002/aris.1440370103.

[2] A. Joshi, P. Bhattacharyya, and M. J. Carman. Automatic sarcasm detection: a survey. *ACM Comput. Surv.*, 50(5), Sept. 2017. DOI: 10.1145/3124420.

[3] J. Tepperman, D. Traum, and S. Narayanan. "Yeah Right": sarcasm recognition for spoken dialogue systems. In *Ninth international conference on spoken language processing*, 2006.

[4] T. Veale and Y. Hao. Detecting ironic intent in creative comparisons. In *Proceedings of 19th European Conference on Artificial Intelligence - ECAI 2010*, pages 765–770, Amsterdam, The Netherlands. IOS Press, 2010. DOI: 10.3233/978-1-60750-606-5-765.

[5] X. Qiu, T. Sun, Y. Xu, Y. Shao, N. Dai, and X. Huang. Pre-trained models for natural language processing: A survey. *Science China Technological Sciences*, 63(10):1872–1897, 2020. DOI: 10.1007/s11431-020-1647-3.

[6] S. Amir, B. C. Wallace, H. Lyu, P. Carvalho, and M. J. Silva. Modelling context with user embeddings for sarcasm detection in social media. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 167–177, Berlin, Germany. Association for Computational Linguistics, Aug. 2016. DOI: 10.18653/v1/K16-1017.

[7] R. Kulshrestha. Transformers. 2020. URL: https://towardsdatascience.com/transformers-89034557de14 (visited on 06/20/2021).

[8] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, 2018. arXiv: 1810.04805.

[9] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. RoBERTa: A Robustly Optimized BERT Pretraining Approach, 2019. arXiv: 1907.11692.

[10] V. Sanh, L. Debut, J. Chaumond, and T. Wolf. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter, 2019. arXiv: 1910.01108.

[11] D. Ghosh, A. Vajpayee, and S. Muresan. A report on the 2020 sarcasm detection shared task. In *Proceedings of the Second Workshop on Figurative Language Processing*, pages 1–11, Online. Association for Computational Linguistics, July 2020. DOI: 10.18653/v1/2020.figlang-1.1.

[12] H. Srivastava, V. Varshney, S. Kumari, and S. Srivastava. A Novel Hierarchical BERT Architecture for Sarcasm Detection. In *Proceedings of the Second Workshop on Figurative Language Processing*, pages 93–97, Stroudsburg, PA, USA. Association for Computational Linguistics, 2020. DOI: 10.18653/v1/2020.figlang-1.14.

[13] R. A. Potamias, G. Siolas, and A. G. Stafylopatis. A transformer-based approach to irony and sarcasm detection. *Neural Computing and Applications*, 32(23):17309–17320, Dec. 2020. DOI: 10.1007/s00521-020-05102-3.

[14] A. Khatri and P. P. Sarcasm detection in tweets with BERT and GloVe embeddings. In *Proceedings of the Second Workshop on Figurative Language Processing*, pages 56–60, Online. Association for Computational Linguistics, July 2020. DOI: 10.18653/v1/2020.figlang-1.7.

[15] M. Bouazizi and T. Otsuki Ohtsuki. A pattern-based approach for sarcasm detection on twitter. *IEEE Access*, 4:5477–5488, 2016. DOI: 10.1109/ACCESS.2016.2594194.

[16] M. Khodak, N. Saunshi, and K. Vodrahalli. A Large Self-Annotated Corpus for Sarcasm, 2017. eprint: 1704.05579.

[17] R. Misra and P. Arora. Sarcasm Detection using Hybrid Neural Network, 2019. arXiv: 1908.07414.

[18] L. Famiglini. Irony-Sarcasm-Detection-Task. URL: https://github.com/lorenzofamiglini/Irony-Sarcasm-Detection-Task (visited on 06/25/2021).

[19] Google Colab. URL: https://colab.research.google.com/notebooks/intro.ipynb (visited on 06/06/2020).

Extraction and Analysis of Sport Activity Data Inside Certain Area

Luka Lukač
University of Maribor,
Faculty of Electrical
Engineering and Computer Science,
Koroška cesta 46, 2000 Maribor, Slovenia
luka.lukac@student.um.si

Abstract

Nowadays, sport data analysis is one of the crucial factors, used to enhance the athletes' performance, which can depend upon many different circumstances. One of those is the area of an exercise, which can dramatically impact on an athlete's performance. Since not enough devotion has been given to this topic, this study focuses on extracting and analysing parts of exercises, which take place inside of a specific area, using principles from another part of Computer Science, Computational Geometry.

Keywords sports, area, extraction, analysis

1 Introduction

As long as professional sport competitions have been organised, professional, as well as amateur athletes have been constantly trying to enhance their performance. In order to know in which aspects their exercises can be improved, tracking and analysing data are crucial. One of important factors, which can influence the performance, is the area where the training is performed. This study focuses on the extraction of the data inside certain areas, which allows thorough analysis of the effectiveness of an athlete in those regions.

2 Materials and Methods

2.1 Materials

In these days, a lot of athletes track their performance during the training using modern sport trackers, which allow them to capture a lot of indicators of their exercise. Since there are many indicators that are not visible directly, but can be extracted as a product of an extensive data analysis, a sport activity can be analysed more thoroughly and deeply. A great tool in analysing the data is Machine Learning, efficient computational methods of which enabled rising the researches in automatic planning of sport training sessions. [1]

Nowadays, many athletes monitor their performance using modern sport equipment, such as sport watches or sport wrists. There are websites that are focused on collecting that kind of data, however, it is very likely that the data does not have a public access [3] and that the databases require a lot of preprocessing skills. An example of a toolbox for extracting features from a sport activity is 'sport-activities-features', which is available in GitHub repository at: `https://github.com/firefly-cpp/sport-activities-features`. It is focused on one of the most intensive and demanding tasks in Machine Learning: data preprocessing. [2]

The overall load indicators that are parts of sport activity datasets (total distance, total duration, average heart rate, etc.) have some disadvantages, such as that details are not expressed sufficiently, only a general outlook of the training is captured and different phases of the training are not recognized directly or are not recognized at all. Thus, more hidden indicators, like intervals, topographic maps and weather data are extracted in the 'sport-activities-features' toolbox. [2]

One of the crucial features that allows retrieving a lot of interesting and useful data is area detection, which is the main part of this study. It is well known that doing exercises in hilly areas can be a lot more difficult and demanding. Despite apparently worse performance in those regions (if only overall indicators are observed), an athlete can put more power in that activity compared to the one which is set in a flat area. As different areas may contain distinct characteristics, it is important to be able to analyse the performance of an athlete in a certain area.

Datasets that are used to extract this kind of data must include positions of an athlete at a certain time. If a dataset is corrupt or some data is missing, the analysis can output wrong results, therefore it is very important to have datasets that have no such issues.

2.2 Methods

In order to unambiguously detect points that are located inside the chosen area, inclusion test is a necessity. There are three major algorithms one can use to determine, whether a point is inside the area:

- Algorithm of the same signs,
- Algorithm of the sum of angles,
- Algorithm with rays.

To start with, the first algorithm that was implemented and seemed to be working was the algorithm of the same signs. Its principles are quite simple: if the sign of the cross product between the hull line and the line from

DOI https://doi.org/10.18690/978-961-286-516-0.11
ISBN 978-961-286-516-0

the hull to the point is the same for all the hull lines, the point lies within the polygon that represents the area with great certainty. However, there are two significant disadvantages regarding this algorithm, namely, it does not allow holes inside the area, and secondly, it does not allow concave polygons (representing area). [4]

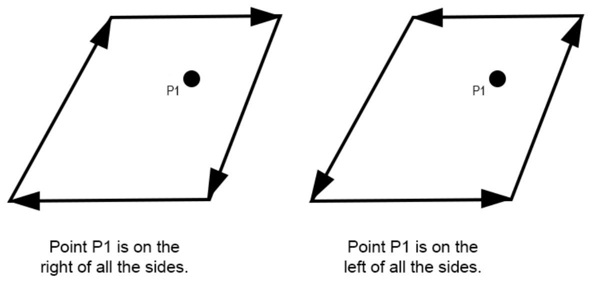

P1
P1
Point P1 is on the right of all the sides.
Point P1 is on the left of all the sides.

Figure 1: Algorithm of the same signs.

The next algorithm that was considered was the algorithm of the sum of angles. To each point that is located on the hull, a ray from the point, which has been chosen for the inclusion test, is made. The next step is to calculate the sum of the angles between the rays, constructed in the previous step. If the sum is 360°, the point is in the area and if the sum is 0°, the point lies outside the area. Although this algorithm allows concave angles on the border of the area, the holes are still not permitted, this is why it has not been implemented to the package as an inclusion test. [4]

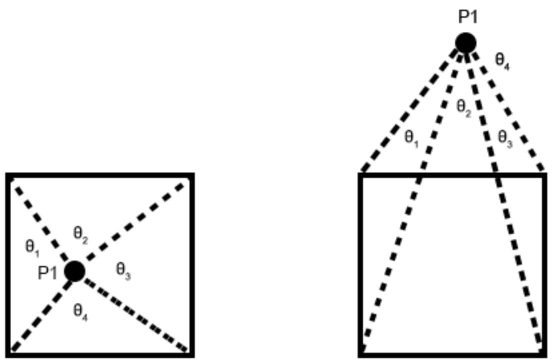


Figure 2: Algorithm of the sum of angles.

Finally, the third algorithm that uses rays to determine whether a point lies inside an area has been implemented. A ray from the point, which we want to check if inside or outside the area, to any direction is being constructed. If the ray intersects with the area border even times, the point is not a part of the area. On the other hand, if there is an odd number of intersections between the ray and the hull, the point lies inside the area. This algorithm is much less error-prone than the aforementioned algorithms, as it produces the right results even if there are holes or concave angles present in the area, this is why this algorithm has been chosen for determining which parts of an exercise were inside the given area. [5]

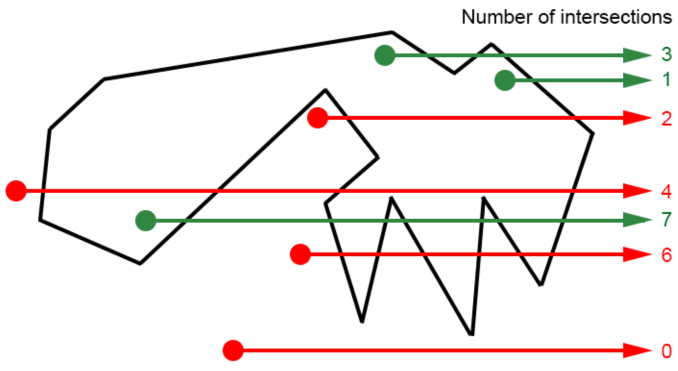


Figure 3: Ray algorithm.

```
1 function GET-POINTS-INSIDE-AREA(positions, area)
2     points_in_area ← {};
3     for position ← 1 to positions.count do
4         n ← number_of_intersections(position, area);
5         if $n \% 2 = 1$ then
6             points_in_area ← points_in_area + position;
7         end
8     end
9     return points_in_area;
```

Algorithm 1: Detection of points of an exercise inside the given area

3 Results

After the implementation of the algorithm that detects parts of an exercise inside area, visualisation has been made in order to be able to test the algorithm properly. To start with, the map sectors, where the exercise is taking place, are downloaded from OSM (OpenStreetMap) and merged into an image, represented as a map. Then, the area that has been used for the extraction of data is plotted on the map, and lastly, the identified and unidentified points of an exercise are being drawn on the map according to the coordinates. An example of the visualization can be found at 5, where the wider area of the Slovenian Littoral is being displayed on the map and the exercise is plotted.

However, not only are parts inside the area extracted, but the data inside the area is also analysed. As already pointed out, the performance of an athlete inside different areas can vary significantly. This is why it is important to be able to retrieve data, specific to a certain region. One important factor about areas is also, what the performance of an athlete was like not only in one training, but in more exercises that took place at different times. This is why data, such as whole distance, average speed, maximum speed, average heart rate, etc. is being extracted from the activities or their parts, which took place inside the same area.

Despite the algorithm working flawlessly, there can be issues with obtaining datasets which would suit the analysis. As this kind of data is usually not publicly available, a user is often limited to analysing their own activities, which is one of the reasons why there is only a limited amount of exercises visualized and displayed in this paper.

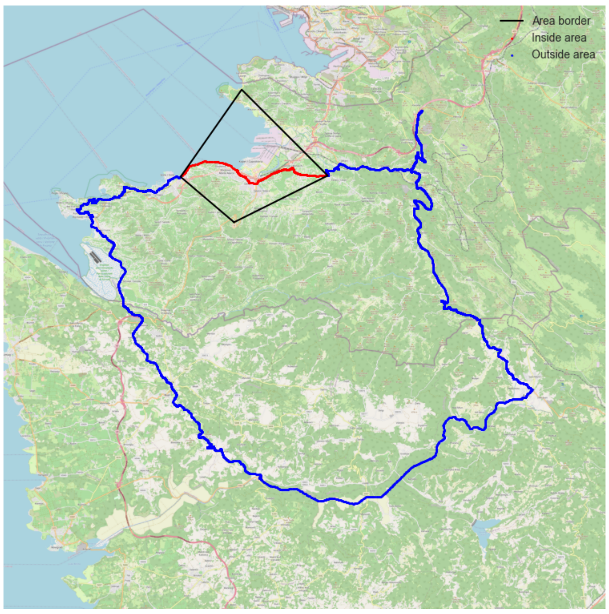


Figure 4: Visualization of the area detection.

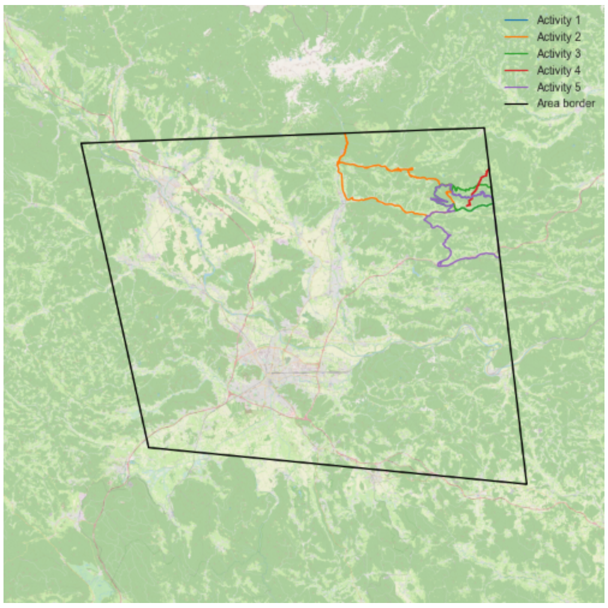


Figure 5: Visualization of more exercises inside area.

4 Discussion

To conclude, area identification is one of the crucial factors in the sport data analysis. Since different areas have different difficulties, it is crucial to be able to analyse regions separately. This study portrays how this task can be done using algorithms from Computational Geometry. In the beginning, parts of exercises inside desired areas should be extracted and then the extracted data can be used for analysis of the performance of an athlete. The next step, which has not been a part of this study, could be the Machine Learning phase, during which the extracted data could be analysed even further.

Acknowledgment

The author of this paper would like to kindly thank Iztok Fister Jr., who provided a lot of motivation and assistance during the writing of this study.

References

[1] Fister, I., Fister Jr, I., and Fister, D. *Computational intelligence in sports.* Springer, 2019.

[2] Fister Jr., I., Lukač, L., Rajšp, A., Fister, I., Pečnik, L., and Fister, D. A minimalistic toolbox for extracting features from sport activity files. In *2021 IEEE 25th International Conference on Intelligent Engineering Systems (INES)* (2021), IEEE, pp. 121–126.

[3] Rajšp, A., and Fister Jr., I. A systematic literature review of intelligent data analysis methods for smart sport training. *Applied Sciences 10*, 9 (2020), 3013.

[4] Topiwala, A. Is the point inside the polygon? *Towards Data Science* (2020).

[5] Vacek, L., Atter, E., Rizo, P., and Nam, B. suas for deployment and recovery of an environmental sensor probe.

Table 1: List of extracted features inside area.

ID	*Feature*	*Description*
1	Distance	Total distance inside area.
2	Time	Total time inside area.
3	Maximum speed	Maximum speed inside area.
4	Average speed	Average speed inside area.
5	Minimum heart rate	Minimum heart rate inside area.
6	Maximum heart rate	Maximum heart rate inside area.
7	Average heart rate	Average heart rate inside area.

Methodology for the Assessment of the Text Similarity of Documents in the CORE Open Access Data Set of Scholarly Documents

Ivan Kovačič
University of Maribor,
Faculty of Electrical
Engineering and Computer Science,
Koroška cesta 46, 2000 Maribor, Slovenia
ivan.kovacic@um.si

David Bajs
University of Maribor,
Faculty of Electrical
Engineering and Computer Science,
Koroška cesta 46, 2000 Maribor, Slovenia
david.bajs@student.um.si

Milan Ojsteršek
University of Maribor,
Faculty of Electrical
Engineering and Computer Science,
Koroška cesta 46, 2000 Maribor, Slovenia
milan.ojstersek@um.si

Abstract

This paper describes the methodology of data preparation and analysis of the text similarity required for plagiarism detection on the CORE data set. Firstly, we used the CrossREF API and Microsoft Academic Graph data set for metadata enrichment and elimination of duplicates of documents from the CORE 2018 data set. In the second step, we used 4-gram sequences of words from every document and transformed them into SHA-256 hash values. Features retrieved using hashing algorithm are compared, and the result is a list of documents and the percentages of coverage between pairs of documents features. In the third step, called pairwise feature-based exhaustive analysis, pairs of documents are checked using the longest common substring.

Keywords data preparation for text similarity analysis, text similarity, CORE data set, metadata enrichment

1 INTRODUCTION

Text similarity is a measure of the degree of content similarity between two textual documents. It is used as a metric which assists for plagiarism detection. In general, plagiarism is a term that is used to describe acts of intellectual theft and violating the material and moral rights of the authors of intellectual property. However, there are several definitions of plagiarism that may be applied, depending on the circumstance of each individual case of plagiarism. Maurer et. al. [9] have compiled a list of possible definitions of plagiarism. They include:

- presenting other people's work as your own
- copying other people's work (or ideas) without giving credit
- improper quotation
- copying sentence structures with minor modification without giving credit
- copying the majority of your work from others, regardless of giving credit [9]

In order to efficiently assert the existence and degree of plagiarism, electronic detection is necessary along with manual analysis. Using electronic detection, the similarity index [1] is a metric which is reliable for identifying potential cases of plagiarism, but it does not necessarily indicate that plagiarism had occurred in an individual case. Another important step is to obtain a sufficient amount of metadata about any given document which is analyzed. Using a sufficient amount of metadata, it is easier to detect duplicate records of the same document, which allows for a higher quality assessment of the degree of plagiarism on a given document. If a journal article is recorded twice in the same data set with inconsistent metadata, there exists a pair of documents with a high degree of similarity. Without good quality metadata, the pair of documents may appear as totally different journal articles, although they represent the same instance of intellectual work. In the following sections, we describe our methodology of data preparation for text similarity, metadata enrichment and the computation of text similarity metrics on the CORE data set of scholarly documents as well as an analysis of the run time complexity of our methodology.

2 RELATED WORK

Eaton and Crossman did a review of literature in the field of self-plagiarism in social sciences research databases and found that only 5.8 % of a sample of search results consisted of primary research done in this field, which indicates that the field of self-plagiarism is a relevant topic of research in social sciences, we may also infer a higher relevance of this research topic for computational methods of detecting self-plagiarism [2].

DOI https://doi.org/10.18690/978-961-286-516-0.12
ISBN 978-961-286-516-0

In [8], a study was described in which a sample of articles that were published in the Scientific Periodicals Electronic Library (SPELL) was analyzed for plagiarism. A comparison was made between two samples taken from the years 2013 and 2018. The former sample contained literal reproductions in 65.9 % of the total sample size, while the latter sample from 2018 contained evidence of plagiarism in 44 % of the total sample size. A reduction of the similarity index was noticeable, the difference being attributed to the fact that the sample of articles from 2018 contained guidelines related to plagiarism and self-plagiarism. The methodology of this study consisted of random sampling of the data set and inputting the samples into iThenticate plagiarism detector. García-Romero and Estrada-Lorenzo have described the analysis of the Déjà vu database using citation analysis and found that cases of plagiarism are more frequently published in journals with lower visibility and also confirmed that duplicate documents not citing the original document show a higher degree of full text similarity than those citing it [5].

3 METHODOLOGY

In this section, we present our methodology of applying plagiarism detection algorithms and procedures on the CORE data set, including metadata enrichment techniques and the plagiarism detection methods used in this study.

3.1 The CORE data set

CORE is one of the leading aggregation services of open-access scholarly documents publicly available on the web. Currently, CORE has aggregated metadata of over 200 million open-access research articles and other scholarly documents, harvested from over 10.000 open-access repositories. A full list of harvested repositories may be seen in [12]. In our study, we used the 2018-03-01 CORE data set. The data set contains 9.767.152 unique documents with full text. Each individual document is represented as a JSON record containing both the full text of the document as well as the extracted metadata. The schema of the JSON records is described in Table 1.

Table 1: CORE data set JSON schema.

JSON attribute	Description
coreId	unique CORE identifier
doi	DOI identifier (*if available*)
title	Title of the document
authors	Authors (array)
contributors	Contributors (array)
datePublished	Date of publication
abstract	The abstract of the document
downloadUrl	URL of the PDF document
language	Language of the document
fullTextIdentifier	Full text unique ID
pdfHashValue	PDF signature hash
journals	List of journals or journal aliases
rawRecordXml	Harvested OAI-PMH XML response
year	Year of publication
relations	Related entities (journals,journal aliases)
topics	OAI-PMH dc:topic value
subjects	OAI-PMH dc:subject value
issn	ISSN (*if available*)
fullText	Full text of the document

Metadata is important for plagiarism/self-plagiarism analysis because it allows an expert who analyses an individual case of plagiarism to evaluate the authorship, year of publication and other data of a document which is being analyzed in comparison to a document's potential candidates for document similarity. The CORE data set does provide a rich aggregation of metadata for scholarly documents, but it is also limited by the quality of metadata provided by the individual servers in harvested OAI-PMH repositories. The CORE team actively develops means of enriching/improving the metadata associated with each individual document, since there exist issues with the metadata provided. These issues include:

- **Duplicate document entries** - there exist several duplicates in the CORE data set. These are present because of multiple publications of the same document in several different repositories that are harvested by CORE. The individual repositories have inconsistent metadata, which makes detection of duplicates a non-trivial task.
- **Unstructured metadata** - most documents in the 2018 CORE data set contain insufficiently structured metadata, since the aggregation process makes it difficult to ascertain the data like the journal name, publisher. This information gets missing in the dc:topic and dc:subject fields and mapping the data in these fields to their respective metadata fields is a non-trivial task.
- **Non-defined document types**
- **Non-defined fields of study**
- **Non-normalized author names**

The issues can be elaborated with a concrete example. The document with CORE ID 47847252 is a book section written in French. The document type, found in the subjects field, is "Book sections", which is a non-standardized document type, originating from the terminology used by the digital library that hosts the OAI server which provided this document. The authors' names are recorded with the last name written first, followed by the initial of the first name. In contrast, the document with CORE ID 6871221 has authors written with the first name written firstly, followed by the last name. Such inconsistencies, which are the result of different bibliographic notations, increase the difficulty of automated detection of self-plagiarism and/or duplication of documents.

3.2 Microsoft Academic Graph (MAG)

In order to tackle the issue of incomplete and inconsistent metadata, we utilized the mapping to the Microsoft Academic Graph done by the CORE team with the help of their published 2019-04-01 MAG mapping data set. The Microsoft Academic Graph is a document graph that contains metadata records about scholarly documents, citation relationships between them, as well as normalized data about authors, journals, fields of study and other metadata. By connecting CORE to MAG by using the DOI persistent identifier, an intersection of over 1.200.000 million articles was made in 2016 [6]. The MAG data set contains structured metadata of scholarly documents. It is organized the same way a classical relational database is, with individual schemas representing entities and connections between them represented as foreign keys. A full description of the MAG schema is available in [3]. We emphasize some of the more useful metadata entries in MAG which allow for metadata enrichment of CORE:

- **Normalized authors** - the authors in MAG are normalized in a way that makes them uniquely identifiable and distinguishable from other authors. The displayed author name is separated from the author's normalized name. Where available, authors are also associated with an affiliation.
- **Fields of study** - papers documented in MAG have a mapping to fields of study of that particular paper. The fields of study are organized in a hierarchical structure which allows us to construct a tree structure of fields of study for each paper in MAG.
- **Journal data** - in MAG journal articles are associated with journals, which have their names normalized and other metadata associated to them, where available. Namely, a citation count, the publisher of the journal and the ISSN, where available [11].

3.3 CrossREF API

CrossREF is a service that aggregates metadata from publishers and provides tools for several use cases related to metadata, among other things it is an agent of the DOI foundation (DF) and it allows users to register DOIs for their scholarly documents. One of the most useful features of CrossREF is the CrossREF public API, which allows for metadata enrichment by resolving a DOI which is contained in the CrossREF database into the metadata of the associated document. The metadata provided by CrossREF is rich and reliable, with information about the journal and publisher, authors with affiliations (where available), document types (e.g. journal article or conference paper), reference counts, the volume and issue, among others. The API itself is free to use, with no signup or authentication required, the client who consumes the REST API must only adhere to throttling mechanisms specified by CrossREF which limit the number of requests per second available to the client. The API returns a requested sleep interval for the client in the `X-Rate-Limit-Interval` HTTP header.

3.4 Metadata enrichment of CORE using CrossREF and MAG

Our approach of metadata enrichment utilized the CORE MAG mapping with some additional enhancements. We imported a published MAG dump available in [10]. When a document in CORE had a DOI available, the document's DOI was used for pairing with MAG, where the mapping was not previously present in the 2019-04-01 MAG mapping data set. In addition, we normalized the document titles and looked for matches in MAG, excluding the pairings where anomalies had been detected (non-matching year of publication, non-matching authors). We further enriched the metadata of these documents by consuming the CrossREF API and acquiring the ISSN, the proper volume, issue and publisher for the documents. Over 30 % of articles in CORE that have a DOI have no ISSN associated with them in the MAG dump provided in [10]. Our method of metadata enrichment is summarized by Algorithm 1.

```
Data: set of CORE documents
Result: set of CORE documents with enriched
        metadata
1  for document in set of CORE documents do
2      if document in CORE MAG mapping and
         document has DOI then
3          consume CrossREF API;
4          map result to metadata properties of the
             CORE document;
5      else if normalized document title and year
         match then
6          enrich document using new MAG
             mapping;
7          consume CrossREF API;
8          map result to metadata properties of
             CORE document;
9      else
10         mark document as incomplete;
11     end
12 end
```

Algorithm 1: CORE metadata enrichment with MAG and CrossREF.

3.5 Text matching in the CORE data set

In this section, we describe how plagiarism detection was implemented on the CORE data set. We serialized the entire CORE 2018-03-01 data set of JSON records into a PostgreSQL database, creating a relational model which corresponds to the JSON schema of CORE. To implement a plagiarism detector, we firstly had to implement a candidate search, followed by a pairwise text matching algorithm.

3.5.1 Candidate retrieval

For finding plagiarism candidates for each individual document, we computed n-gram models for each document in CORE. The full text of the documents is tokenized with a minimum sentence length of 40 characters, using rules for mapping ordinal numbers, URLs, email addresses to tokens, removing stop words and computing lexicographically sorted n-gram sequences with $n = 4$. The minimum length of 40 characters was chosen because it is used in the Slovenian open access infrastructure. Sequences of 4-grams have been determined as more efficient for candidate retrieval in previous studies [4]. A space delimited sequence of sentence n-grams is transformed into an SHA-256 hash value which is stored in a database. The set of all n-gram hashes represents the set of features of the entire full text which can be matched to other documents, thus acquiring lists of potential candidates for plagiarism. In relational databases, this approach can be implemented efficiently by performing a table join of the table containing the SHA-256 hashes with itself, with the table itself being structured as a key-value map, where the key is a unique ID of the document which is mapped to the hashed n-gram value. In order for such a table join query to be time-efficient, indexes are necessary on the column containing the n-grams hashes, which drastically increases the space complexity of the described method. Once the self-join query had been performed, a hash coverage index was computed:

$$hashCoverage = \frac{\text{unique covered hashes in candidate}}{\text{total number of hashes in document}}$$

The candidates are then sorted in descending order by the *hashCoverage* metric and stored into the database as a candidate list. The procedure of candidate search is summarized in Algorithm 2.

```
  Data: fileid - CORE document ID
  Result: set of candidate documents in CORE
1 for document in set of CORE documents do
2     perform self-join on the table of hashes
        where document ID = fileid;
3     compute hashCoverage;
4     sort candidate list by hashCoverage
        descending;
5     store candidate list into database;
6 end
```

Algorithm 2: Candidate retrieval with hashed n-gram sequences.

3.5.2 Text matching of candidates

For each document, a maximum of 100 candidate documents is set as a limit for reasons of reducing the space complexity of the procedure. After lists of candidates are obtained for all documents in CORE, a fine text comparison algorithm is used by Kärkkäinen et. al. which constructs a permuted longest common prefix array efficiently [7]. The output of the algorithm is a sequence of longest common prefixes along with offsets at which a computed longest common prefix occurred in the text. Our implementation of the algorithm is written in C++ and wrapped by a REST API, which accepts two texts with configuration parameters and returns the longest common prefix array. The configuration parameters are the minimum length of the document (*minLen*), the minimum length of the longest common suffix (*lengthBoundary*) and the size of the window in which neighbor prefixes are searched (*windowSize* - for merging adjacent common prefixes into longer sequences). The REST API is bundled into a Docker image, which allows for an implemention of a distributed approach for calculating text matching on the CORE dataset. A workload of documents is generated, on which text matching is performed for each candidateas a pair (document ID, *status*) in a table in a relational database, with *status* being a flag described in Table 2.

Table 2: Distributed text matching workload status flags.

Status flag	Description
-1	unprocessed
0	processing
1	processed

Each node running the distibuted program performs a transactional `UPDATE` query on the workload table, setting the flags of N documents to 0 and simultaneously obtaining the document IDs from the table where the current flag value equals -1. It proceeds by obtaining all the full texts of the document and the candidates from the database and passing them on to a pairwise comparison performed by the Docker REST service which is running on the same node. This approach allows for efficient parallel computation of text matching between documents and their candidates. The parameter N is limited by the amount of free memory available to the node. Given an average number of candidates C per document and an average length L bytes of a document in CORE, the estimated space complexity in bytes iss given by $N * C * L$. If a node failed to perform the comparison, for example in the case of running out of free virtual memory, a transactional update of the status flag is made, setting the flag back to -1. This allows the document to be processed by another node. Using this approach, the procedure is fault-tolerant and partition tolerant. The results of the text matching comparisons are arrays of offsets and length of common prefixes, which are used in the implementation of a plagiarism detector text matching user interface. The prefixes are also used to compute the similarity indices of the documents:

$$similarityIndex = \frac{\Sigma \text{ length of matches}}{\text{total length of document}}$$

3.5.3 Deduplication

After computing pairwise text matching between documents and their candidates for plagiarism, the process of plagiarism detection in the CORE database is complete. We described the problem of duplicate entries in the CORE data set in 3.1. With overall and pairwise similarity indices computed, we were able to conduct deduplication by using the computed indices in conjunction with the enriched metadata we extracted from MAG and CrossREF. The process of removing duplicates begins by acquiring a list of all pairwise matches with a similarity index above 85 %. For each such pair, we compare the normalized titles of the two documents, the year of publication and ISSN. When a full match has been made, the second document is automatically marked as duplicated. When only a partial match has been made (e.g. only the normalized titles match), the document is marked as a potential duplicate. The potential duplicates are submitted for manual inspection, meanwhile the detected duplicates are excluded from candidate lists and the similarity indices of the documents which contained duplicates are corrected, respectively.

4 RESULTS

By utilizing our method for metadata enrichment, we enriched metadata for 3.457.071 documents in the 2018-03-01 CORE data set which contain a DOI persistent identifier value. We also acquired a complete set of metadata for a subset of 618.754 documents. This result is crucial for the goal of data preparation, since text similarity analysis is an instrument of plagiarism detection, which requires a sufficient amount of metadata in order to be useful to the expert conducting the plagiarism analysis. The latter, smallest subset contains ISSN values which allow for further studies on journal plagiarism and self-plagiarism statistics by implementing plagiarism detection on this subset. Using our candidate retrieval method, we have computed over 4.7 billion n-grams hashes for the 9.8 million documents in the 2018-03-01 CORE data set, yielding a total hash table size of 1478 GB in PostgreSQL, using b-tree indexes and document IDs in the form of SHA-256 hashes. On average, we found 87.37 candidates per document for each document in the CORE data set.

For our implementation of text matching comparison, we installed the Docker image containing the text matching REST API implementation on a cluster of 33 machines with Intel i5-8600 and 8 GB internal memory. Our configuration for the parameters equaled $minLen = 20$, $lengthBoundary = 40$ and $windowSize = 350$, respectively, as they are set used in the Slovenian open access infrastructure. The average document length in the CORE 2018-03-01 data set is 73.41 KB, we found setting the parameter N=20 allows for stable and efficient processing of the CORE data set using this approach. Table 3 contains benchmark statistics for a varible number of computing nodes running the text similarity Docker image. A random sample of 1000 documents is selected for a variable amount of nodes which process the workload in parallel. The efficiency of the approach tends to asymptotically decline as we increase the number of nodes, which is a result of the centralized data storage being used for synchronization of the compute nodes. The network infrastructure and the hardware and configuration of the database server represent a bottleneck. The table also contains a reference value for the time necessary to process all the 9.767.152 documents in the CORE 2018 data set for a given number of nodes. The benchmarks were performed on a subset of 10 nodes of the total 33 nodes used to process the entire CORE data set, but the results may be interpolated to a higher number of nodes.

Table 3: Distributed text similarity computation performance benchmarks.

#nodes	duration(s)	est. days (CORE 2018)
1	1590,806	179,83
2	753,917	85,23
3	547,659	61,91
4	425,046	48,05
5	374,564	42,34
6	306,461	34,64
7	294,918	33,34
8	285,888	32,32
9	233,364	26,38
10	217,015	24,53

5 CONCLUSION

The study described in this paper describes the methodology of establishing the largest plagiarism detection dataset in Slovenia. We have developed a framework of processing larger sets of documents into a pipeline for plagiarism detection, with means of metadata enrichment with the help of the CrossREF API and Microsoft Academic Graph. Our output allows for further study in the field of academic integrity, content similarity detection, stylometry and other fields of study. Utilizing our data set, which consists of enriched metadata entries for the 2018-03-01 CORE data set, including the similarity indices for each document in the data set, further research is possible.

6 ACKNOWLEDGMENTS

The authors thank Petr Knoth and the CORE team for providing the 2018-03-01 CORE data set and technical support during our efforts of obtaining data from CORE. Also, we thank David Schmid for publishing the MAG dump on Zenodo from April 2019.

References

[1] Bretag, T., and Mahmud, S. Self-plagiarism or appropriate textual re-use? *Journal of Academic Ethics 7*, 3 (Sep 2009), 193.

[2] Eaton, S. E., and Crossman, K. Self-plagiarism research literature in the social sciences: A scoping review. *Interchange 49*, 3 (Aug 2018), 285–311.

[3] Eide, D., and Huang, C. "microsoft academic graph schema". *Microsoft Academic Graph official website*. Accessed Jun 1, 2021. [Online] Available: https://docs.microsoft.com/en-us/academic-services/graph/reference-data-schema.

[4] Fartek, J. *Distributed generation of plagiarism detection reports*. J. Fartek, 2018. Bachelor thesis.

[5] García-Romero, A., and Estrada-Lorenzo, J. M. A bibliometric analysis of plagiarism and self-plagiarism through déjà vu. *Scientometrics 101*, 1 (Oct 2014), 381–396.

[6] Herrmannova, D., and Knoth, P. An analysis of the microsoft academic graph. *D-Lib Magazine 22* (09 2016).

[7] Kärkkäinen, J., Manzini, G., and Puglisi, S. J. Permuted longest-common-prefix array. In *Combinatorial Pattern Matching* (Berlin, Heidelberg, 2009), G. Kucherov and E. Ukkonen, Eds., Springer Berlin Heidelberg, pp. 181–192.

[8] Krokoscz, M. Plagiarism in articles published in journals indexed in the scientific periodicals electronic library (spell): a comparative analysis between 2013 and 2018. *International Journal for Educational Integrity 17*, 1 (Jan 2021), 1.

[9] Maurer, H., Kappe, F., and Zaka, B. Plagiarism - a survey. *Journal of Universal Computer Science 12* (01 2006), 1050–1084.

[10] Microsoft Academic. Microsoft academic graph. *Zenodo*. Accessed Jun 1, 2021. [Online] Available: `https://doi.org/10.5281/zenodo.2628216`, Apr. 2019.

[11] Sinha, A., Shen, Z., Song, Y., Ma, H., Eide, D., Hsu, B.-J. P., and Wang, K. An overview of microsoft academic service (mas) and applications. In *Proceedings of the 24th International Conference on World Wide Web* (New York, NY, USA, 2015), WWW '15 Companion, Association for Computing Machinery, p. 243–246.

[12] The CORE team. "core data providers". *CORE official website*. Accessed Jun 1, 2021. [Online] Available: `https://core.ac.uk/data-providers?q=&size=10`.

Embedding Non-planar Graphs: Storage and Representation

Đorđe Klisura
University of Primorska,
Faculty of Mathematics, Natural Sciences
and Information Technologies,
Glagoljaška ulica 8, 6 000 Koper, Slovenia
klisuradjordje10@gmail.com

Abstract

In this paper, we propose a convention for representing non-planar graphs and their least-crossing embeddings in a canonical way. We achieve this by using state-of-the-art tools such as canonical labelling of graphs, Nauty's Graph6 string and combinatorial representations for planar graphs. To the best of our knowledge, this has not been done before. Besides, we implement the mentioned procedure in a SageMath language and compute embeddings for certain classes of cubic, vertex-transitive and general graphs. Our main contribution is an extension of one of the graph data sets hosted on MathDataHub, and towards extending the SageMath codebase.

Keywords planar graph, graph representation, crossing number, graph database

1 Introduction

Computers are increasingly used by mathematicians to assist them in their studies. This includes most aspects of a researcher's work, from publishing and reading papers to computations in mathematical software. Perhaps surprisingly, mathematicians also generate and use data, and the production and processing of massive datasets are becoming increasingly important in several areas of mathematics.

The primary applications for these mathematical datasets and databases are exploratory in nature. They are used by researchers to test hypotheses or to discover patterns and counterexamples. It's not difficult to find such "datasets" that even predate computers: the *Atlas of Graphs* and the *Foster census* are two examples from graph theory. The Online Encyclopedia of Integer Sequences (OEIS) is a modern mathematical database that represents an online database of integer sequences. The OEIS now contains over 334000 sequences that are useful to both professional and amateur mathematicians, making it the largest database of its kind. The sequences in the database serve as fingerprints for the records they are associated with. A somewhat similar project in graph theory is the *House of Graphs*. An important notion is that of using mathematical objects, such as integer sequences, or graphs, to search for mathematical theorems. This has been introduced as theorem fingerprinting by Billey and Tenner [2] as a way to improve the efficiency of searching for mathematical knowledge. In a broader sense, fingerprints are used in many fields of science, ranging from computer science to chemistry, archaeology, and genetics. Computer documentation, reducing duplication in web search results, and surely DNA fingerprinting are a few examples.

Because of its applications in physics, biochemistry, biology, electrical engineering, astronomy, operations research, and computer science, graph theory is rapidly moving into the core of mathematics. The theory of planar graphs is based on Euler's polyhedral formula, which is related to the polyhedron edges, vertices and faces. Planar graphs are used in a variety of applications in the modern era, including constructing and organizing sophisticated radio electronic circuits, railway maps, planetary gearboxes, and chemical molecules. Pipelines, railway lines, subway tunnels, electric transmission lines, and metro lines are all vitally crucial for modelling an urban city. For further readings on this topic, look at Trudeau and Richard [13], and Barthelemy [1].

1.1 Related work and contributions

Some of the most significant projects that act as mathematical databases which are of assistance to researchers in their research projects are the SageMath platform [12], American Mathematical Society MathSciNet [11], above-mentioned Encyclopedia of Integer Sequences [6], House of Graphs [5], and Atlas of Graphs [10].

The above-named tools are far from perfect and are many times subject to important work of the open-source community. This project aims to provide another toolset for researchers, via improving the platform MathDataHub which will, in the future, provide our database containing planar embeddings minimising the number of crossings. Those embeddings are hard to compute and such a database of precomputed embeddings does not exist in any mathematical database.

The paper is structured as follows. In Section 2, we present the central technique and ideas of embedding non-planar graphs. Moreover, we give a concrete algorithm that uses them and evaluates them in terms of space and time complexity. In Section 3, we talk about the impact of our results so far. Finally, in Section 4, we present some output samples of the algorithm that has been evaluated before we make some concluding remarks in Section 5.

DOI https://doi.org/10.18690/978-961-286-516-0.13
ISBN 978-961-286-516-0

2 Embedding non-planar graphs

In this section, we present our main result, namely the algorithm for calculating the canonical embedding of non-planar graphs.

```
Data: Non-planar graph $G$
Result: Planar embedding, crossing number
        and added vertices of $G$
$V \quad V(G)$
$edgePairs \quad \{ab, cd$, where $a, b, c, d \in V(G)$ and
    $ab, cd \in E(G)\}$
$k \leftarrow 0$
while $G$ is not planar do
    $S \leftarrow$ all $k$-subsets from $edgePairs$
    for $\{a_1, a_2, \ldots, a_k\}$ in $S$ do
        $E(G) \leftarrow E(G) \setminus \bigcup_{i=1}^{k} a_i$
        for $a_i$ element in $\{a_1, a_2, \ldots, a_k\}$ do
            $\{\{a_i^1, a_i^2\}, \{a_i^3, a_i^4\}\} \leftarrow a_i$
            $v_i \leftarrow$ vertex such that $v_i \notin V(G)$
            $V(G) \leftarrow V(G) \cup \{v_i\}$
            $E(G) \leftarrow E(G) \cup \{a_i^1 v_i,\ v_i a_i^2,\ a_i^3 v_i,$
                $v_i a_i^4\}$
        end
        if $G$ is planar then
            return $G, k, V(G) \setminus V$
        end
    end
    $k \quad k + 1$
end
```

Algorithm 1: Algorithm for calculating the canonical embedding of non-planar graphs.

After investigating several non-planar graphs with up to five vertices in SageMath we came up with the notion of representing their embeddings. Combinatorial embedding is a key concept in the study of such graph embeddings. The significance stems from the fact that, when combined with canonical labelling, combinatorial embeddings can be utilized to generate a unique representation of (planar) embeddings for (planar) graphs. For further reading on combinatorial representations and planar embeddings, refer to Mutzel and Weiskircher [9], Didjev [8], Duncan, Goodrich and Kobourov [4], and to Hopcroft and Tarjan [7].

The concept is as follows: we first construct all non-incident pairs of edges of a graph; then, we go through those pairs of edges and for each crossing of two edges, we delete those edges and add a new vertex to which we connect vertices of deleted edges. We repeat the process until the graph is planar. Finally, if it is planar, in the end, we canonically reorient its vertices and save new embedding of a graph.

We show the approach and demonstrate how it works using a Petersen graph shown in Figure 1. The first step of Algorithm 1 is constructing all pairs of non-incident edges of G, meaning that the two different edges cannot share the same vertex. Those pairs of edges are $\{\{0,1\},\{2,3\}\}$, $\{\{4,9\},\{5,8\}\}$ etc. In the beginning, we initialise crossing number k to 0, to check if the graph is already planar if it is we return k and if it is not, k is increased by 1, and we are going through the set of pairs of non-incident edges. For each k we modify the graph until we get a planar embedding, in the following way: we take the first pair, in our example pair $\{\{0,1\},\{2,3\}\}$ and we delete edges $\{0,1\}$ and $\{2,3\}$. Then we add a new vertex v to which we connect the vertices of the deleted edges: $\{0\},\{1\},\{2\},\{3\}$. We check for planarity. If the checking confirmed a positive result, that is, confirmed that the graph is planar, the crossing number is returned and the algorithm terminates. However, if the checking confirmed a negative result, that is, confirmed that the graph is not planar we go to the next pair. If all pairs fail, we look for tuples of size 3 next, and so on.

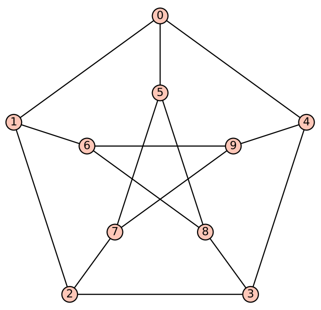


Figure 1: Petersen graph.

We get that graph G is planar after three iterations, hence, the crossing number of the Peterson graph is 2. In the end, we canonically relabel vertices and we obtain a planar embedding presented in Figure 2.

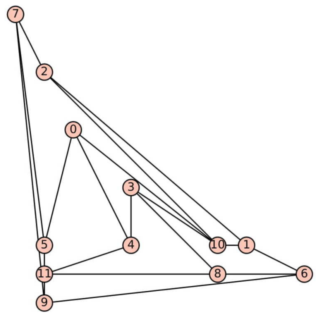


Figure 2: Planar embedding of Petersen graph after applying Algorithm 1.

See Section 3 for more details about the transformation.

2.1 Theoretical analysis of the algorithm

Consider first the space complexity of Algorithm 1. The amount of memory used by Algorithm 1 to execute and produce the result is linear with respect to the input instance. This is due to the fact that the input instance is a graph and most of the work on it is done in-place, by modifying it locally and not taking more space even after many manipulations. Hence, we can say that Algorithm 1 does not take too much memory.

Next, let us evaluate the time complexity of Algorithm 1. To determine the time complexity, we need to consider all of the SageMath integrated functions we called in our main function. Function *is_planar* that is implemented runs in linear time, concerning the graph as an input instance, meaning it runs in time $\mathcal{O}(n+m)$ where n is a number of vertices and m is a number of edges of the graph. For more reading on the time complexity of the planarity algorithm, refer to Boyer and Myrvold [3]. To remove a vertex in a graph, we first need to find the vertex in the data structure and the time complexity depends on the structure we use; if we use a $HashMap$, the time complexity will be $\mathcal{O}(1)$. Then we remove the vertex, and we do it in $\mathcal{O}(n)$ time. Adding and removing an edge of the graph is done in constant time, $\mathcal{O}(1)$, time while adding a vertex to a graph takes $\mathcal{O}(n)$ time. Checking if there is an edge between vertices is done in $\mathcal{O}(n)$ time since a vertex can have at most $\mathcal{O}(n)$ neighbours. The time complexity of getting an embedding of the graph and of finding the neighbours is linear, that is $O(n+m)$, since we needed to perform the Breadth-First Search algorithm.

Let us analyze the lines from Algorithm 1. Line 1 has complexity $\mathcal{O}(n)$, as it assigns to a set n vertices, while line 2 assigns m^2 edge pairs to a set and hence has complexity $\mathcal{O}(m^2)$. Line 3 has constant, $\mathcal{O}(1)$, complexity.

Let us now analyse the complexity of the while loop. Inside the while loop, we see that line 5 generates all subsets of size k from the set of size m^2. Hence, the assignment of the k-subsets to the set has complexity equal to the size of the set which is $\mathcal{O}(\binom{m^2}{k}) = \mathcal{O}(m^{2k})$. Now, the first for loop goes through all k elements of the k-subset, and has complexity $\mathcal{O}(m^{2k}(k+kn+m+n)) = \mathcal{O}(m^{2k}(nk+m))$ because the line 7 has complexity $\mathcal{O}(k)$, the inner for loop has complexity $\mathcal{O}(nk)$ and the if statement in line 14, that checks whether the graph is planar, has complexity $\mathcal{O}(n+m)$. Finally, the while loop is executed k times meaning that the complexity of the whole while loop is $\mathcal{O}(m^{2k}k(nk+m))$.

3 Applications

Drawing: One of the applications is in graph colouring. In Algorithm 1, we labelled newly added edges, then we use the method *plot*() within SageMath and with the property colour by label, we get different colours for the newly added edges. In Figure 3 we see an example of the transformed Petersen graph from Figure 2. As it can be seen, the original graph embedding is coloured red, while the newly inserted edges are coloured green and blue.

Storing graphs: Another application of our approach is related to the storing of graphs with their combinatorial embedding, added vertices (if the graph is non-planar) and with Graph6 string. We constructed a function that stores data about an individual graph in a single text file that a computer can comprehend (Graph6 string, calculated embedding and added vertices - separated by

(1).jpg (1).jpg (1).jpg

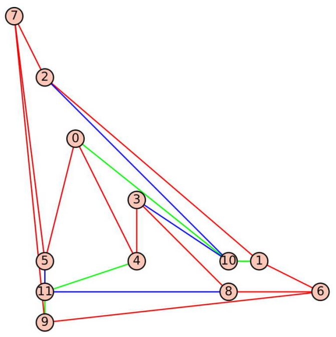


Figure 3: Colouring of planar embedding of Petersen graph.

semicolons) since we aim to store graphs in the database. Furthermore, we added the certificate flag *verbose* so that we may provide a detailed output (when set to *True*) for users - with output explanations.

By now, we processed cubic graphs with up to 21 edges, vertex-transitive graphs with up to 20 edges and all graphs with up to 13 edges. These files can be stored in any database since we created a non-verbose mode of writing into them. In Table 1 we present an overall of the processed families of graphs by now.

Table 1: Processed families of graphs

family of graphs	**# generated**	**up to edges**
cubic	752	21
vertex-transitive	16	20
general	376899	13

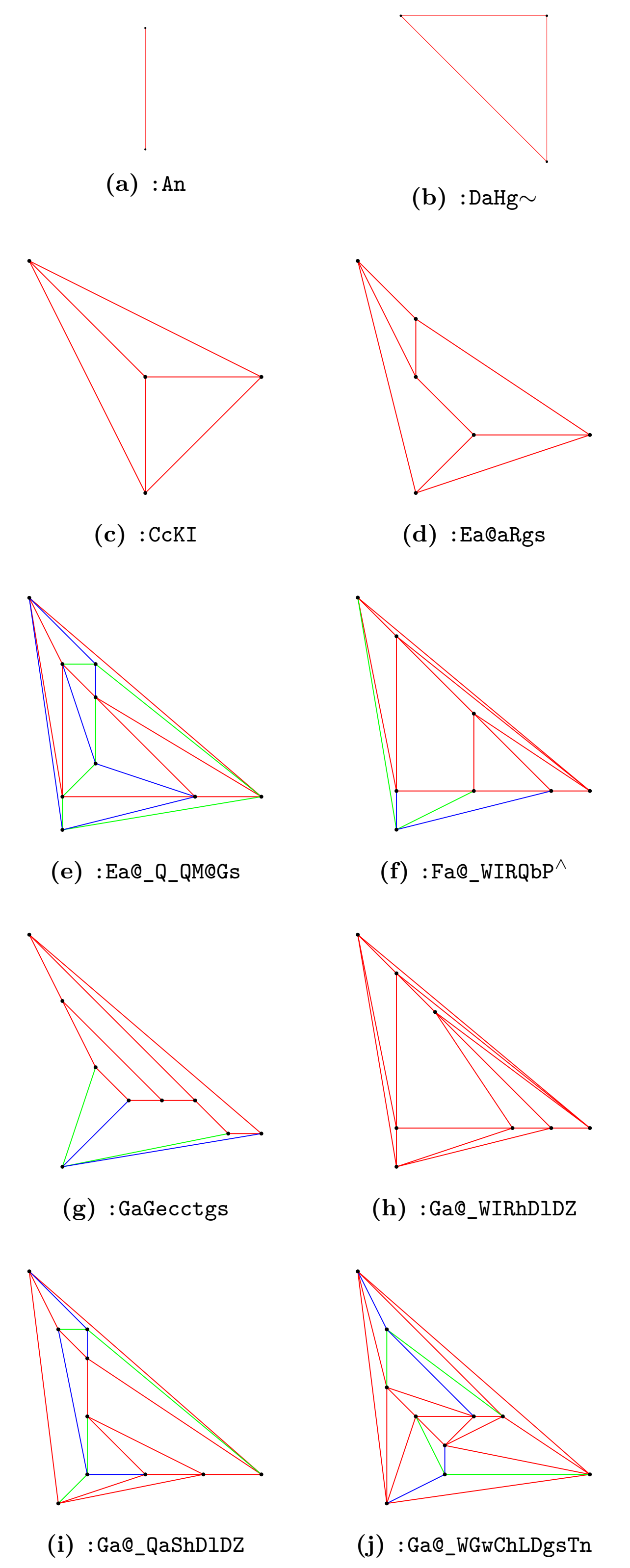
4 Output samples

Here we present some examples of the algorithm's final outputs, as previously detailed in the paper. We've successfully generated embeddings, saved them in files along with additional vertices and Graph6 strings, and plotted images of vertex-transitive graphs on less than 20 edges. In Figure 4 we present some of them individually, with the Graph6 string for each one written in the caption.

5 Conclusions

In the paper, we had a look at a non-planar graph embedding, its storage, and representation. We introduced an algorithm for constructing those embeddings and a function that writes them in both a human-readable way, or in the way suitable for the storage in the database. This contributes to the subject of representation theory because there was no standard way of encoding such embeddings.

We demonstrated how our method may be used to draw graphs and save graph data in various file formats. Our techniques can be used to enrich almost any graph



(a) `:An` **(b)** `:DaHg~`

(c) `:CcKI` **(d)** `:Ea@aRgs`

(e) `:Ea@_Q_QM@Gs` **(f)** `:Fa@_WIRQbP^`

(g) `:GaGecctgs` **(h)** `:Ga@_WIRhDlDZ`

(i) `:Ga@_QaShDlDZ` **(j)** `:Ga@_WGwChLDgsTn`

Figure 4: Coloured representations of several vertex-transitive graphs with up to 20 edges, with Graph6 string in captions.

database, and that is exactly what we were hoping to achieve. We've generated vertex-transitive graphs with up to 20 edges, all graphs with up to 13 edges, and all cubic graphs with up to 21 edges by now. In collaboration with Katja Berčič, PhD, our files will be uploaded to the *MathDataHub* database.

Currently, we are working on contributing our code to the SageMath project.

Acknowledgment

I'd like to thank Professor Matjaž Krnc for his advice and suggestions throughout the planning, development, and writing of this paper.

I'd like to thank Klemen Berkovič for his help in codifying this *Author's Guide*, and to Iztok Fister Jr. for his contribution to *Author's Guide* and `.tex` files.

References

[1] Barthelemy, M. *Morphogenesis of Spatial Networks.* New York: Springer, 2017.

[2] Billey, S. C., and Tenner, B. E. Fingerprint databases for theorems. *Notices of the AMS 60*, 8 (2013), 1034.

[3] Boyer, J. M., and Myrvold, W. J. On the cutting edge: Simplified o(n) planarity by edge addition. *Journal of Graph Algorithms and Applications 8*, 3 (2004), 241–273.

[4] Christian, D., T., G. M., and Stephen, K. Planar drawings of higher-genus graphs, graph drawing, 17th international symposium, gd 2009. vol. 5849, pp. 45–56.

[5] Goedgebeur, G. B. K. C. J., and Mélot, H. *House of Graphs: a database of interesting graphs, Discrete Applied Mathematics*, 2013.

[6] Inc., O. F. *The On-Line Encyclopedia of Integer Sequences*, 2021. `http://oeis.org`.

[7] John, H., and E., T. R. Efficient planarity testing. *Journal of the Association for Computing Machinery 21*, 4 (1974), 549–568.

[8] N., D. H. On drawing a graph convexly in the plane, graph drawing, dimacs international workshop, gd '94, princeton. vol. 894, pp. 76–83.

[9] Petra, M., and René, W. Computing optimal embeddings for planar graphs, computing and combinatorics, 6th annual international conference, cocoon 2000. vol. 1858, pp. 95–104.

[10] Read, R. C., and Wilson, R. J. *An Atlas of Graphs (Mathematics).* Oxford University Press, Inc., USA, 2005.

[11] TePaske-King, Bert; Richert, N. *The Identification of Authors in the Mathematical Reviews Database*, 2001.

[12] The Sage Developers. *SageMath, the Sage Mathematics Software System (Version 9.4)*, 2021. `https://www.sagemath.org`.

[13] Trudeau, and J., R. *Introduction to Graph Theory.* New York: Dover Pub, 1993.

Analiza ritmičnosti števnih podatkov z uporabo modela cosinor

Nina Velikajne
Faculty of Computer and Information Science,
University of Ljubljana,
Večna pot 113, SI-1000 Ljubljana, Slovenia
nv6920@student.uni-lj.si

Miha Moškon
Faculty of Computer and Information Science,
University of Ljubljana,
Večna pot 113, SI-1000 Ljubljana, Slovenia
miha.moskon@fri.uni-lj.si

Povzetek

Analiza ritmičnosti števnih podatkov je postala pomembna v mnogih vidikih znanosti, inženirstva in celo ekonomije. Obstajajo metode z namenom detekcije ritmičnosti zveznih podatkov, ki pa večinoma niso primerne za analizo števnih podatkov. V prispevku predstavimo metodologijo, ki omogoča analizo ritmičnosti v števnih podatkih. Metoda združuje metodo cosinor z uporabo različnih računskih regresijskih modelov, ki so primerni za analizo števnih podatkov. Omogoča tako detekcijo ritma kot tudi ocenitev parametrov ritma, primerjavo zgrajenih modelov in iskanje optimalnega števila komponent za metodo cosinor ter iskanje najbolj ustreznega tipa števnega modela. Vzpostavljena metoda omogoča primerjavo zaznanega ritma v odvisnosti od različnih parametrov ritmičnosti in izračun njihovih intervalov zaupanja. Celotno metodologijo smo testirali na tedenski periodičnosti realnih podatkov COVID-19 obolenj v Sloveniji.

Ključne besede metoda cosinor, analiza ritmičnosti, števni podatki, pojavnost dogodkov, regresija.

1 Uvod

Detekcija in analiza ritmičnih vzorcev v števnih podatkih ima pomembno vlogo pri mnogih vidikih znanosti. Periodični podatki so podatki, v katerih se vzorci ponavljajo z določeno periodo. Zelo pogost tip periodičnih procesov so procesi, ki odražajo cirkadiano nihanje – procesi s periodo 24-ur [3]. Regulira jih Zemljina rotacija in izmenjavanje dneva in noči, ki vpliva na vse organizme in posledično na njihovo obnašanje ter gibanje (glej [20]). Poseben tip podatkov predstavljajo števni podatki, ki opisujejo pojavnost izbranih dogodkov [3].

Števni podatki se pogosto pojavljajo periodično. V naravi in naši okolici so tovrstni podatki vseprisotni. Z njihovo analizo lahko pripomoremo k razumevanju različnih dogodkov in posredno k razumevanju delovanja organizmov in družbenih sistemov. Na primer, analiza števila porodov glede na uro v dnevu lahko pomaga pri organizaciji medicinskega in babiškega osebja v porodnišnici [14]. Analiza dnevnih vzorcev v prometu in njihovo spreminjanje skozi čas nam lahko pove veliko o gibanju in obnašanju populacije, posebej v času epidemije (glej [4]).

Za analizo števnih periodičnih podatkov potrebujemo posebne računske metode, ki upoštevajo in ohranjajo odnose med podatki. Metode morajo upoštevati diskretno porazdelitev, ki je omejena le na nenegativne cele vrednosti. Pri uporabi navadne linearne regresije so lahko napovedane vrednosti negativne, kar je teoretično nemogoče [8]. Za zaznavanje in analizo ritmičnosti v zveznih podatkih obstaja kar nekaj neparametričnih metod (glej [18, 12]). V primerjavi z omenjenimi metodami (glej [16]) nam uporaba trigonometričnih regresijskih metod v povezavi z različnimi cosinor modeli predstavlja številne prednosti, npr. ocenitev parametrov ritma [18]. Izkaže se tudi, da alternativne metode v določenih primerih odpovejo zaradi velikega števila osamelcev (angl. *outliers*), same velikosti podatkov, neuravnoteženosti podatkov in zbiranja podatkov brez ponovitev (glej [17]).

V tem prispevku predstavimo metodologijo, ki z združevanjem metode cosinor skupaj z različnimi števnimi računskimi modeli upošteva vse omejitve števnih periodičnih podatkov in tako omogoča tudi ocenitev parametrov ritma. Metoda omogoča izračun intervalov zaupanja za posamezen parameter ritma s pomočjo samovzorčenja (angl. *bootstrapping*). S F testom določimo optimalno število komponent za metodo cosinor, za iskanje najbolj ustreznega tipa števnega modela pa uporabimo Vuongov test.

Članek je razdeljen na pet poglavij. V drugem in tretjem poglavju so predstavljeni metoda cosinor za analizo periodičnih podatkov in pet računskih regresijskih modelov za delo s števnimi podatki. Sledi poglavje, ki opisuje postopek izbire najbolj ustreznega računskega modela. V poglavju Rezultati so predstavljeni rezultati testiranja vzpostavljene metode na realnih podatkih. V zadnjem poglavju so zajete ključne ugotovitve in postopki celotne analize.

2 Metoda cosinor

Pri periodičnih podatkih se opazovani vzorci ponovijo z določeno periodo. Njihovo analizo lahko naslovimo kot regresijski problem, pri katerem pa je potrebno upoštevati tudi karakteristike ritma, npr. fazo in amplitudo nihanja. Metoda cosinor se uporablja za analizo časovnih vrst in se posveča tako detekciji ritma kot tudi oceni parametrov ritma. Model v ozadju metode lahko opišemo z Enačbo 1, kjer je N število komponent, M srednja vrednost ritma (MESOR, angl. *Midline Estimating Statistic of Rhythm*), A amplituda, P perioda in $e(t)$ funkcija napake. Spremenljivka t označuje čas, i pa iterira po številu

DOI https://doi.org/10.18690/978-961-286-516-0.14
ISBN 978-961-286-516-0

komponent – od 1 do N [1, 15, 16].

$$Y(t) = M + \sum_{i=1}^{N} \left(A_{i,1} \cdot \sin\left(2\pi \frac{t}{P/i}\right) + A_{i,2} \cdot \cos\left(2\pi \frac{t}{P/i}\right) \right) + e(t), \quad (1)$$

Če je perioda ritma znana vnaprej, lahko enačbo metode cosinor poenostavimo v model linearne regresije:

$$Y(t) = M + \sum_{i=1}^{N} (A_{i,1} \cdot X_{i,1}(t) + A_{i,2} \cdot X_{i,2}(t)) + e(t), \quad (2)$$

kjer je $X_{i,1}(t) = \sin\left(2\pi \frac{t}{P/i}\right)$ in $X_{i,2} = \cos\left(2\pi \frac{t}{P/i}\right)$. V kolikor perioda ni znana, jo lahko ocenimo s pomočjo uporabe periodogramov (angl. *periodograms*) [1, 15, 16].

3 Analiza števnih podatkov z metodo cosinor

Za analizo in detekcijo ritma na izvornih očiščenih podatkih naprej uporabimo metodo cosinor. V primeru znane periode uporabimo Enačbo 2, ki podatke razdeli na poljubno število komponent in jih transformira do regresijske oblike. Na transformirane podatke nato apliciramo regresijske računske modele. Regresijski modeli omogočajo identifikacijo in karakterizacijo odnosov med mnogimi faktorji. Zaradi dela s števnimi podatki moramo izbrati ustrezne regresijske računske modele, ki upoštevajo vse lastnosti tovrstnih podatkov.

Podatki so diskretni, omejeni na nenegativne cele vrednosti in velikokrat tudi razpršeni (angl. *dispersed*). Srečamo se s pojmom povečane (angl. *overdispersion*) ali pa zmanjšane razpršitve (angl. *underdispersion*) [8]. Pri povečani razpršitvi imajo podatki večjo varianco, kot bi jo sicer pričakovali. Če je varianca večja kot povprečje podatkov, gre za povečano razpršitev. Obratno velja za zmanjšano razpršitev [7]. Z uporabo navadne linearne regresije bi dobili nepravilne rezultate, saj tak računski model ne bi upošteval omenjenih lastnosti [8].

V vzpostavljeni metodologiji smo se odločili za uporabo petih različnih računskih modelov, ki se najpogosteje uporabljajo za analizo števnih podatkov. Uporabili smo Poissonov model (angl. *Poisson model*), generaliziran Poissonov model (angl. *generalised Poisson model*), Poissonov model z inflacijo ničel (angl. *zero-inflated Poisson model*), negativen binomski model (angl. *negative binomial model*) in negativen binomski model z inflacijo ničel (angl. *zero-inflated negative binomial model*).

Poissonov model predpostavlja, da so podatki porazdeljeni s Poissonovo porazdelitvijo:

$$P(y = k) = \frac{\lambda^k e^{-\lambda}}{k!}, \quad (3)$$

kjer je λ povprečna pričakovana vrednost oz. število dogodkov na enoto časa. Povprečje podatkov μ je pri Poissonovi porazdelitvi enako povprečni pričakovani vrednosti λ. Varianca σ^2 je enaka povprečju, poenostavljeno velja, da je σ^2=λ. Model ne upošteva povečane ali pa zmanjšane razpršitve podatkov [9].

Generaliziran Poissonov model izhaja iz navadnega Poissonovega modela. Bistvena razlika tega modela v primerjavi s Poissonovim modelom je ta, da je generaliziran Poissonov model primeren tudi za podatke, ki imajo povečano ali zmanjšano razpršitev. Vpeljemo dodaten parameter α, ki opisuje stopnjo disperzije. Model torej ne zahteva, da je povprečje μ enako varianci σ^2. Obstajata dve različici generaliziranega Poissonovega modela – GP-1 in GP-2 [6]. V vzpostavljeni metodologiji smo uporabili različico GP-1.

Poissonov model z inflacijo ničel je razširitev navadnega Poissonovega modela. Tovrstni model za razliko od navadnega in generaliziranega Poissonovega modela upošteva, da so ničelne vrednost bolj pogoste kot ostale vrednosti. Izhaja iz tega, da obstajata dva dejavnika, ki vplivata na izid, ali je vrednost ničelna ali neničelna [13].

Negativen binomski model predpostavlja, da so podatki porazdeljeni z negativno binomsko porazdelitvijo. Uporabljata se dve verziji negativnega binomskega modela, tj. NB-1 in NB-2. Različica NB-1 se je izkazala kot bolj primerna, prilagojena krivulja se je namreč vidno lepše prilegala izvornim podatkom, zato smo v metodologiji uporabili verzijo NB-1. Varianca takega modela je definirana kot $\sigma^2 = \mu + \alpha \cdot \mu$, kjer je α parameter disperzije in μ povprečje. Povprečje je enako povprečni pričakovani vrednosti λ. Model je zato primeren tudi za podatke s povečano ali pa zmanjšano razpršitvijo [2, 11]. Ob večanju parametra α varianca konvergira k povprečju in negativna binomska porazdelitev postane Poissonova [2].

Negativen binomski model z inflacijo ničel je razširitev negativnega binomskega modela. Podobno kot Poissonov model z inflacijo ničel upošteva, da so ničelne vrednosti bolj pogoste kot ostale (neničelne) vrednosti. Ključna razlika v primerjavi s Poissonovim modelom z inflacijo ničel je, da ta model temelji na negativni binomski porazdelitvi. Model je zato primeren tudi za podatke, ki imajo povečano ali pa zmanjšano razpršitev [10, 11].

4 Izbira najustreznejšega modela

Izbiro najbolj ustreznega računskega modela razdelimo na dva nivoja. Na prvem nivoju iščemo optimalno število komponent za metodo cosinor. Na tem mestu smo izhajali iz tega, da so modeli gnezdeni. Dva modela sta gnezdena, če lahko prvi model izrazimo z drugim oz. če drugi model poleg vsaj enega dodatnega parametra vsebuje enake parametre kot prvi [5]. Implementirali smo F test. Na podlagi dveh zgrajenih modelov izračunamo F vrednost. F test temelji na razliki vsote kvadratov (angl. *sum of squares*) dveh modelov in upošteva število parametrov modela [5].

Na drugem nivoju smo vrednotili tip računskega modela. Uporabili smo Vuongov test, ki je primeren tako za gnezdene modele kot tudi za ne gnezdene in prekrivajoče se (angl. *overlapping*) modele. Vuongov test omogoča izračun Z vrednosti na podlagi logaritma največjega verjetja (angl. *maximum log-likelihood*) dveh modelov. Tudi ta test upošteva število parametrov modelov [19].

Oba testa sledita podobnemu postopku. Za dva modela, tj. model A in model B, izračunamo F oz. Z vrednost.

Model *A* zavržemo, če je izračunana vrednosti manjša od vnaprej določene meje, tj. statistične signifikance (angl. *statistical significance*) [5, 19].

5 Rezultati

Vzpostavljeno metodologijo smo preizkusili na realnih podatkih. Podatke smo pridobili s strani COVID-19 sledilnik[1] in analizirali število pozitivnih testov glede na dan v tednu. Upoštevali smo vse teste, tj. seštevek PCR (angl. *polymerase chain rection*) in hitrih antigenskih (HAGT) testov. Metodo smo izvedli na treh podatkovnih zbirkah. Prva zbirka beleži število pozitivnih testov od 20. oktobra 2020 – razglasitev 2. vala epidemije v Sloveniji, do 10. februarja 2021. Povprečje pozitivnih testov je 1340, varianca pa 276278. Druga zbirka vsebuje primere od 11. februarja 2021 do 26. aprila 2021. Povprečje podatkov je 796, varianca pa 90328. Tretja podatkovna zbirka združuje časovni obdobji prve in druge podatkovne zbirke. Povprečje zadnje zbirke je 1082, varianca pa 232224. Metodo smo preizkusili tudi za časovno obdobje 1. epidemije v Slovenji – pomlad 2020, vendar je teh podatkov premalo za smiselno analizo.

V podatkovni zbirki smo najprej odstranili osamelce (angl. *outliers*), tako da smo za posamezno uro odstranili vnose, kjer so bile vrednosti števila pozitivnih testov večje ali manjše od 0,15 kvantila. Nato smo izvedli metodo cosinor za posamezno število komponent – od 1 do 4, in zgradili posamezen tip računskega modela (glej Poglavje 1). Pri številu komponent 4 smo se ustavili, ker se računski modeli zaradi prevelikega števila komponent niso več prilagajali izvornim podatkom. Zgrajene modele smo nato ovrednotili, najprej smo poiskali optimalno število komponent na podlagi F testa in nato še najbolj ustrezen tip modela s pomočjo Voungovega testa. Opisan postopek se ponovi za posamezno podatkovno zbirko. Vse podatkovne zbirke imajo večjo varianco kot povprečje kar pomeni, da imajo podatki povečano razpršitev. Na podlagi porazdelitve izvornih podatkov (glej Sliko 1) lahko ugotovimo, da podatki nimajo ničelnih vrednosti.

Težave se pojavijo pri vseh modelih, v kolikor je število komponent pri metodi cosinor večje od 3. Opazimo, da se izvornim podatkom najbolje prilegata negativen binomski model in generaliziran Poissonov model. Oba modela, sta namreč primerna za podatke s povečano razpršitvijo. Za vse podatkovne zbirke smo dobili enak rezultat. Optimalno število komponent je 3, najbolj ustrezen tip modela pa je generaliziran Poissonov model (glej Sliko 1). Na podlagi izvedene analize smo lahko ovrednotili parametre ritmičnosti (glej Tabelo 1) in njihove intervale zaupanja (glej Tabelo 2). Celoten postopek analize z vsemi vmesnimi rezultati je dostopen v repozitoriju GitHub[2].

6 Diskusija in zaključek

Vzpostavljena in implementirana metodologija omogoča analizo števnih periodičnih podatkov. Metodologija je se-

[1]https://covid-19.sledilnik.org/sl/data

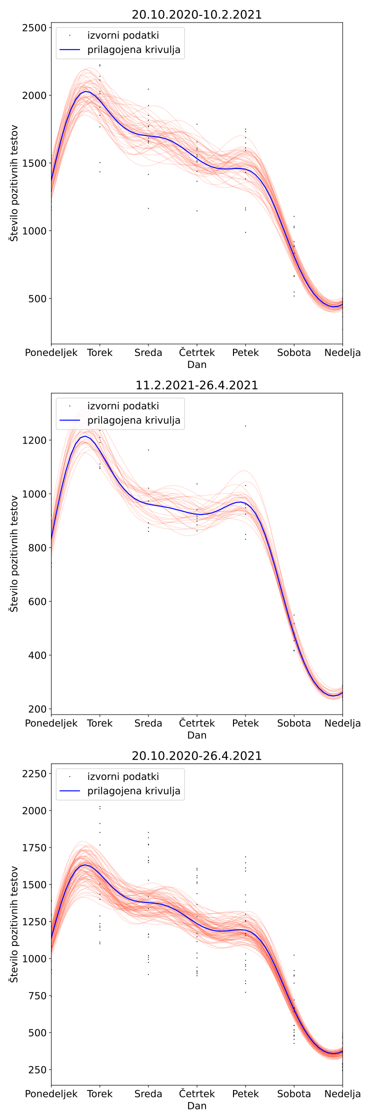
[2]https://github.com/ninavelikajne/Ritmicnosti-stevnih-podatkov

Tabela 1: Ocenjeni parametri ritmičnosti za posamezno podatkovno zbirko.

pod. zbirka	tip modela	št. komponent	amplituda	mesor	vrhovi	št. poz. testov
1.	gen_poisson	3.0	794.62	1233.67	0.7	2028.29
2.	gen_poisson	3.0	483.37	731.32	0.7 3.9	1214.69 969.59
3.	gen_poisson	3.0	637.66	995.12	0.7	1632.77

Tabela 2: Intervali zaupanja parametrov ritmičnosti za posamezno podatkovno zbirko.

pod. zbirka	amplituda	mesor	vrhovi	št. poz. testov
1.	[727.44 863.81]	[1164.72 1299.48]	[0.63 0.85]	[1897.05 2158.4]
2.	[452.11 514.36]	[701.36 757.49]	[0.56 0.83] [1.96 4.82]	[1154.14 1271.19] [902.71 1040.37]
3.	[596.96 691.66]	[955.36 1051.93]	[0.59 0.86]	[1556.13 1739.78]



Slika 1: Zmagovalni modeli za posamezno podatkovno zbirko. V vseh primerih je število komponent enako 3 in tip modela generaliziran Poissonov model. Oranže črte označujejo intervale zaupanja modela.

stavljena iz dveh delov. Prvi del predstavlja analizo periodičnih podatkov. Implementirana je metoda cosinor, ki ji lahko uporabnik nastavlja poljubno število komponent. Drugi del zajema analizo števnih podatkov. Uporabljeni so različni računski regresijski modeli, ki so primerni za delo s števnimi podatki. Implementirali smo pet tovrstnih modelov, tj. Poissonov model, generaliziran Poissonov model, Poissonov model z inflacijo ničel, negativen binomski model in negativen binomski model z inflacijo ničel. Vzpostavljena metodologija omogoča vrednotenje zgrajenih modelov. Tudi vrednotenje se tako kot grajenje modelov deli na dva dela. V prvem delu se osredotočimo na iskanje optimalnega števila komponent za metodo cosinor. Izhajamo iz dejstva, da so modeli gnezdeni in jih zato lahko ovrednotimo s F testom. V drugem delu iščemo najbolj ustrezen tip računskega modela. Uporabimo Vuongov test, ki je primeren tako za gnezdene, negnezdene in tudi prekrivajoče se modele.

Računsko metodo smo preizkusili na tedenski periodičnosti števila obolenj z boleznijo COVID-19 v Sloveniji. Podatke smo razdelili na 3 podatkovne zbirke, vsaka opisuje različno obdobje. Za vse podatkovne zbirke se kot najboljši tip modela izkaže generaliziran Poissonov model, optimalno število komponent pa je število 3 (glej Sliko 1). V podatkih se ritmičnost podatkov lepo izraža. Ocenili smo parametre ritma (glej Tabelo 1) in njihove intervale zaupanja (glej Tabelo 2). Z uporabo predstavljene metode na dobljenih rezultatih razberemo, da je število pozitivnih testov največje ob torkih nato pa skozi teden upada. Kot pričakovano je število pozitivnih testov najmanjše ob vikendih. Oblika zaznanega ritma je za vse podatkovne zbirke podobna (glej Sliko 1).

Celotna metoda je implementirana kot modul v jeziku Python in je prosto dostopna. Omogoča širok spekter funkcionalnosti za analizo tovrstnih podatkov. Potencialni uporabnik lahko direktno spreminja in prilagaja funkcionalnosti glede na svoje potrebe.

Literatura

[1] Bingham, C., Arbogast, B., Guillaume, G. C., Lee, J. K., and Halberg, F. Inferential statistical methods for estimating and comparing cosinor parameters. *Chronobiologia 9*, 4 (1982), 397–439.

[2] Cameron, A. C., and Trivedi, P. K. Econometric models based on count data: Comparisons and applications of some estimators and tests. *Journal of Applied Econometrics 1*, 1 (1986), 29–53.

[3] Cameron, A. C., and Trivedi, P. K. *Regression analysis of count data*, vol. 53. Cambridge University Press, 2013.

[4] Chang, S., Pierson, E., Koh, P. W., Gerardin, J., Redbird, B., Grusky, D., and Leskovec, J. Mobility network models of covid-19 explain inequities and inform reopening. *Nature 589*, 7840 (2021), 82–87.

[5] Clark, T. E., and McCracken, M. W. Tests of equal forecast accuracy and encompassing for nested models. *Journal of econometrics 105*, 1 (2001), 85–110.

[6] Consul, P., and Famoye, F. Generalized Poisson regression model. *Communications in Statistics - Theory and Methods 21*, 1 (1992), 89–109.

[7] Coxe, S., West, S. G., and Aiken, L. S. The analysis of count data: A gentle introduction to Poisson regression and its alternatives. *Journal of personality assessment 91*, 2 (2009), 121–136.

[8] Gardner, W., Mulvey, E., and Shaw, E. Regression analyses of counts and rates: Poisson, overdispersed Poisson, and negative binomial models. *Psychological bulletin 118* (12 1995), 392–404.

[9] Gardner, W., Mulvey, E., and Shaw, E. Regression analyses of counts and rates: Poisson, overdispersed Poisson, and negative binomial models. *Psychological bulletin 118* (12 1995), 392–404.

[10] Greene, W. H. *Accounting for excess zeros and sample selection in Poisson and negative binomial regression models.* Working Paper EC-94-10. Leonard N. Stern School of Business, New York University, 1994.

[11] Hilbe, J. M. *Negative binomial regression.* Cambridge University Press, 2011.

[12] Hutchison, A. L., Maienschein-Cline, M., Chiang, A. H., Tabei, S. A., Gudjonson, H., Bahroos, N., Allada, R., and Dinner, A. R. Improved statistical methods enable greater sensitivity in rhythm detection for genome-wide data. *PLoS Comput Biol 11*, 3 (2015), e1004094.

[13] Lambert, D. Zero-inflated Poisson regression, with an application to defects in manufacturing. *Technometrics 34*, 1 (1992), 1–14.

[14] Martin, P., Cortina-Borja, M., Newburn, M., Harper, G., Gibson, R., Dodwell, M., Dattani, N., and Macfarlane, A. Timing of singleton births by onset of labour and mode of birth in nhs maternity units in england, 2005–2014: A study of linked birth registration, birth notification, and hospital episode data. *PloS One 13*, 6 (2018).

[15] Nelson, W., LEE, J. K., et al. Methods for cosinor-rhythmometry. *Chronobiologia 6*, 4 (1979), 305–323.

[16] Refinetti, R., Cornélissen, G., and Halberg, F. Procedures for numerical analysis of circadian rhythms. *Biological rhythm research 38*, 4 (2007), 275–325.

[17] Ruben, M. D., Francey, L. J., Guo, Y., Wu, G., Cooper, E. B., Shah, A. S., Hogenesch, J. B., and Smith, D. F. A large-scale study reveals 24-h operational rhythms in hospital treatment. *Proceedings of the National Academy of Sciences 116*, 42 (2019), 20953–20958.

[18] Thaben, P. F., and Westermark, P. O. Detecting rhythms in time series with RAIN. *Journal of biological rhythms 29*, 6 (2014), 391–400.

[19] Vuong, Q. H. Likelihood ratio tests for model selection and non-nested hypotheses. *Econometrica 57*, 2 (1989), 307–333.

[20] Zhdanova, I. Melatonin. In *Encyclopedia of the Neurological Sciences (Second Edition)*, M. J. Aminoff and R. B. Daroff, Eds., second edition ed. Academic Press, Oxford, 2014, pp. 1030 – 1033.

Analiza sentimenta komentarjev hotelov z uporabo slovarjev in metode Naivni Bayes

Nina Murks
Univerza v Mariboru,
Fakulteta za elektotehniko, računalništvo
in informatiko,
Koroška cesta 46, 2000 Maribor, Slovenija
nina.murks@student.um.si

Anže Omerzu
Univerza v Mariboru,
Fakulteta za elektotehniko, računalništvo
in informatiko,
Koroška cesta 46, 2000 Maribor, Slovenija
anze.omerzu@student.um.si

Borko Bošković
Univerza v Mariboru,
Fakulteta za elektotehniko, računalništvo
in informatiko,
Koroška cesta 46, 2000 Maribor, Slovenija
borko.boskovic@um.si

Povzetek

V članku smo predstavili pristop k analizi sentimenta komentarjev hotelskih gostov s pomočjo slovarjev in metode Naivni Bayes. Najprej smo zgradili slovarja sentimenta, ki sta vsebovala n-grame, ter njihove verjetnosti, da pripadajo pozitivnemu ali negativnemu razredu. Nato smo s pomočjo zgrajenih slovarjev klasificirali komentarje hotelov, pri čemer smo uporabili metodo Naivni Bayes. Pri klasifikaciji k omentarjev s mo računali klasifikacijske v rednosti o z. v erjetnosti, da so posamezni komentarji pozitivni ali negativni. Komentarje smo klasificirali s p omočjo unigramov in bigramov, ter rezultate primerjali z rezultati iz literature. Pri unigramih smo dosegli natančnost 0,92, pri bigramih je natančnost znašala 0,80. Klasifikacijske v rednosti posameznih komentarjev smo si shranili, pri čemer smo pri komentarjih, ki smo jih klacificirali kot negativne, dodali negativen predznak. Predznačene klasifikacijske vrednosti smo nato sešteli, za vsak hotel ter na tak način izračunali hotelom pripadajoče točke. Točke hotelov so v našem primeru pokazatelj splošnega zadovoljstva hotelskih gostov, ki ga najdemo v komentarjih. Glede na točke smo hotele uredili po vrsti in prišli do lestvice hotelov, pri katerih najdemo najbolj pozitivne komentarje.

Ključne besede analiza sentimenta, komentarji hotelskih gostov, slovar sentimenta, n-grami, Naivni Bayes

1 Uvod

Preden se odpravimo na potovanje, je ena izmed pomembnih odločitev izbira nastanitve. Pri določitvi kraja namestitve so nam pomembne namestitvene zmožnosti – ali ima hotel bazen, organizirano varstvo, brezplačno parkiranje itd. Več o kvaliteti storitev, ki jih ponuja izbrana nastanitev, lahko izvemo neposredno od gostov, ki so storitve že koristili; pobrskamo po spletnih komentarjih, povezanih z mnenjem o nastanitvi.

Odločili smo se, da analizo sentimenta apliciramo na komentarje gostov hotelov. Sprva smo komentarje klasificirali kot pozitivne ali negativne. S pomočjo klasificiranih komentarjev smo ugotovili, ali je bil večini hotelskih gostov izbran hotel všeč ali ne. Cilj našega eksperimenta je bila aplikacija, ki omogoča takojšen vpogled v širše mnenje o hotelu, ki ga najdemo v komentarjih hotelskih gostov.

V članku smo predstavili različne pristope k analizi sentimenta, in sicer v poglavju Sorodna dela. V naslednjem poglavju sledi opis eksperimenta, ki smo ga izvedli. Znotraj tega smo najprej predstavili, na kakšen način smo pridobili podatke in kako smo jih predprocesirali. Nato smo predstavili način grajenja slovarjev, njihovo uporabo in klasifikacijo komentarjev hotelskih gostov. V zadnjem podpoglavju tega poglavja smo predstavili pristop računanja točk za posamezen hotel. Nazadnje podamo še zaključek in možnosti za nadaljnje delo.

2 Sorodna dela

Analiza sentimenta zajema računalniško analizo stališč govorca ali pisca. Stališče pisca, ki ga lahko prepoznamo v besedilu, je lahko pozitivno, negativno ali nevtralno. Danes je uporaba analize izjemno koristna pri spremljanju družbenih omrežij, saj nam omogoča vpogled v javno mnenje [2].

Uporaba analize sentimenta je dobro raziskano področje, kar kaže visoko število pojavitev znanstvenih in strokovnih člankov/prispevkov s tega področja. Znanih je več pristopov k analizi sentimenta. V nadaljevanju smo jih predstavili nekaj – po vzoru že objavljenih člankov [8][3].

Prvi pristop temelji na unigramih in vzorcih iz učne množice (ang. training set) [1][10]. Avtorji člankov so raziskovanje usmerili na vzorce komentarjev na družbenem omrežju Twitter. S pomočjo unigramov in vzorcev so razpoznavali, ali je komentar sovražen (žaljiv) ali čist (nev-

DOI https://doi.org/10.18690/978-961-286-516-0.15
ISBN 978-961-286-516-0

tralen) ali pozitiven [10]. V drugem članku so raziskovalci s pomočjo vzorcev iskali prisotnost sarkazma v komentarjih.

Tudi v naslednjem, za naše raziskovanje pomemben članku [11], so avtorji svoje raziskovanje usmerili na sentimente komentarjev na družbenih omrežjih. Njihov pristop uporablja utežene besedne vektorje, ki predstavljajo vhod v celico nevronske mreže (BiLSTM), ki zazna kontekstne informacije. Tako bolje predstavi vektorje komentarjev. Sentiment komentarjev se kasneje naprej določi s klasifikacijo nevronskih mrež.

V članku [8] so uporabili ordinalno regresijo z uporabo tehnik učenja. Za omenjeno tehnologijo so uporabili javno bazo komentarjev na Twitterju, ki jih je bilo treba vnaprej procesirati z uporabo metode ekstrakcije lastnosti. Za klasifikacijo analize sentimenta so uporabili različne algoritme: Multinomijska logistična regresija (Multinomial logistic regression - SoftMax), podpora vektorski regresiji (Support Vector Regression - SVR), odločitvena drevesa (Decision Trees - DTs) in naključni gozd (Random Forest - RF).

V članku [7] so raziskovalci za analizo sentimenta uporabili in nadgradili pristop k ansamblu lastnosti (ang. feature ensemble), pri katerem so upoštevali različne elemente, ki jih drugi raziskovalci pri uporabi te metode zanemarijo. Upoštevali so besedoslovje, besedno vrsto, jezikovno semantiko in položaj besed.

Do zdaj omenjeni pristopi uporabljajo analizo sentimenta nad angleškim jezikom. Zanimalo nas je, kako se analize sentimenta lotevajo v tujih jezikih, kjer je sentiment manj jasen. Prav to so storili v članku [4], v katerem so se posvetili nejasnosti kitajskih fraz, ki so izraženi v sentimentu. Tradicionalni pristop s strojnim učenjem ne more prikazati resničnega sentimenta, zato so predlagali uporabo multistrateške metode analize razpoloženja in prikazali, da hibridna analiza sentimenta dosega zadovoljive rezultate.

Članka [5] in [9] vsebujeta analizo sentimenta s pomočjo konvolucijskih nevronskih mrež. Prvi naveden članek uporablja konvolucijske mreže za analizo sentimenta komentarjev na Twitterju, drugi pa za zaznavanje sarkazma na socialnih omrežjih. Konvolucijske nevronske mreže so sestavljene iz več slojev, kjer vsak sloj opravlja nalogo pripravljanja, pretvarjanja ali popravljanja podatkov.

Analize sentimenta se lahko lotimo tudi s pomočjo slovarjev in metode Naivni Bayes, kot so to storili v članku [6]. V tem članku so analizirali sentiment komentarjev Danmaku[1] videov. Uporabili so slovar sentimenta, ki so ga razširili z emotikoni, saj so ti zelo pomembni pri komentarjih Danmaku videov. Pri analizi sentimenta pa niso samo klasificirali pozitivne in negativne komentarje, temveč so uporabili kar sedem razredov: gnus, žalost, všečnost, jeza, presenečenje, strah in veselje. Po tem, ko so generirali slovar, ki ustreza kontekstu (za komentarje Danmaku videov), so se lotili klasifikacije komentarjev s pomočjo metode Naivni Bayes. Pristop z uporabo slovarjev in metode Naivni Bayes je bil od vseh (prej omenjenih) najprimernejši, zato smo se odločili, da ga uporabimo pri implementaciji lastne ideje.

[1]Danmaku je Japonski izraz za sistem podnapisov, ki ga uporabljajo spletne video platforme. Omogočajo uporabniku objavljanje premikajočih se komentarjev na video, ki se predvaja.

3 Predlagan pristop

V tem poglavju smo predstavili našo analizo sentimenta za klasificiranje hotelskih komentarjev in pridobivanje točk zadovoljstva nastanitve posameznih hotelov, ki se skrivajo v besedilu komentarjev. Sledi kratka predstavitev načina pridobivanja podatkov, ter načrtovanja in izvedbe eksperimenta.

3.1 Pridobivanje podatkov

Poiskali smo podatkovno bazo, ki je že vsebovala klasificirane komentarje hotelov. To bazo podatkov smo našli na spletni strani Kaggle. Sledi kratka predstavitev spletne strani.

Kaggle, podružnica podjetja Google LLC, je spletna skupnost podatkovnih znanstvenikov in izvajalcev strojnega učenja. Kaggle uporabnikom omogoča, da najdejo in objavijo nabore podatkov, raziskujejo in gradijo modele v spletnem okolju za znanost o podatkih, sodelujejo z drugimi znanstveniki in inženirji strojnega učenja ter se udeležujejo tekmovanj za reševanje izzivov na področju podatkov.

Izbrana baza podatkov je bila narejena s pomočjo spletne strani Booking.com. Vsebuje 515.000 komentarjev o 1493 različnih Evropskih hotelih. Podatki imajo naslednjo obliko: naslov hotela, datum komentarja, povprečen rezultat hotela (ta je izračunan s pomočjo zadnjega komentarja, ki se je pojavil v tekočem letu), ime hotela, nacionalnost pisca komentarja, negativen komentar, število besed v negativnem komentarju, pozitiven komentar, število besed v pozitivnem komentarju, točke pisca komentarja (točke, ki jih je pisec dodelil hotelu), število komentarjev, ki jih pisec podal, število komentarjev, ki jih ima posamezen hotel, značke (ki jih je dodelil pisec komentarja), koliko dni je minilo od zadnjega komentarja, dodatne točke k hotelu (nekateri so napisali samo točke) ter geografska širina in dolžina lokacije hotela.

Komentarji v podatkovni bazi so že imeli izločena vsa ločila in znake, ki jih ni v angleški abecedi (ASCII znaki). Ker je bila baza dobro pripravljena, nismo imeli veliko dela – odstranili smo številke in pretvorili besedilo komentarjev v male črke. Podatkovno bazo smo tudi razdelili na testno in učno množico, pri čemer je učna množica zajemala približno 80 % vseh podatkov, testna pa preostalih 20 %. Učno množico smo uporabili za grajenje slovarjev, testno pa za klasifikacijo komentarjev. Delitev podatkov v učno in testno množico je potekala za vsak hotel posebej. Zbrali smo vse komentarje o določenemu hotelu, nato pa smo le-te razdelili na učno in testno množico tako, da je učna množica vsebovala približno 80 % komentarjev.

3.2 Grajenje slovarjev

Podatki so bili predhodno obdelani in lahko smo se lotili gradnje slovarjev. Odločili smo se, da bomo algoritem izvedli na dva načina – s pomočjo unigramov in bigramov. Algoritem, ki upravlja z unigrami, se od algoritma z bigrami ne razlikuje preveč, le da slovarja temeljita enkrat na unigramih, drugič pa na bigramih. Tudi pri uporabi slovarjev in računanju klasifikacij s pomočjo metode Naivni Bayes je prišlo do razlike le pri predprocesiranju posameznih komentarjev, kjer smo pri unigramih vzeli unigrame, torej posamezne besede, pri bigramih, pa bigrame, torej dvojice besed.

Grajenja slovarjev smo se lotili tako, da smo najprej prebrali učno množico podatkov in si shranili vse komentarje posebej – razdelili smo jih na pozitivne in negativne. Pri vsaki različici algoritma (unigrami in bigrami) smo naredili dva slovarja. Prvi slovar je imel izračunane verjetnosti, da posamezen n-gram (kadar koli se sklicujemo na n-gram imamo v mislih unigram ali bigram) pripada pozitivnemu komentarju, drugi pa verjetnosti, da posamezen n-gram pripada negativnemu komentarju. Da smo verjetnosti pripadnosti n-gramov v posamezen razred lahko izračunali, smo najprej morali narediti še nekaj korakov. Če smo gradili slovar pozitivnih komentarjev, smo potrebovali vse n-grame, ki jih najdemo v besednjaku pozitivnih komentarjev. Potrebovali smo tudi seznam vseh n-gramov, ki se najdejo tako v pozitivnih komentarjih kot tudi v negativnih, in njihove frekvence pojavitve. Izračunali smo tudi velikost slovarja vseh n-gramov (koliko različnih n-gramov se nahaja v pozitivnih in negativnih komentarjih). Sedaj smo se lahko lotili računanja posameznih verjetnosti n-gramov, da le-ti pripadajo določenemu razredu (pozitivnemu ali negativnemu). Verjetnost, da n-gram pripada razredu smo izračunali po enačbi:

$$P(\textit{n-gram}|\textit{razred}) = \frac{\textit{frekvenca}(\textit{n-gram}, \textit{razred}) + 1}{\textit{frekvenca}(\textit{razred}) + V} \tag{1}$$

Vsako verjetnost izračunamo tako, da vzamemo frekvenco pojavitve n-grama znotraj razreda, tej prištejemo ena (da se izognemo ničelnim vrednostim), nato pa dobljen rezultat delimo z vsoto frekvence razreda in velikostjo slovarja (V). Frekvenca razreda predstavlja število n-gramov, ki se pojavi znotraj posameznega razreda. Za vsak n-gram smo izračunali verjetnost za oba razreda (pozitivni in negativni razred) ter si na tak način ustvarili slovarja, ki smo ju shranili v datoteki. Ker se v testni množici, ki smo jo uporabili za klasifikacijo, lahko pojavi tudi takšen n-gram, ki ga v učni množici (s pomočjo te smo gradili slovarja) nismo imeli, smo v slovar na začetek dodali vrednost neznanega n-grama in ga označili kot "unknown_". Pri oznaki smo uporabili podčrtaj, saj bi se sam n-gram "unknown" lahko pojavil pri komentarjih, torej bi v tem primeru vzeli napačno verjetnost. Ker pa smo iz podatkovne zbirke že odstranili vsa ločila, smo tukaj lahko uporabili to lastnost in n-gramu dodali ločilo.

3.3 Klasifikacija komentarjev

Klasifikacija komentarjev iz testne množice je potekala tako, da smo najprej prebrali slovarja, ki smo si ju shanili v prejšnjem koraku. Ustvarili smo podatkovno strukturo Review, ki je vsebovala vsebino komentarjev, informacijo o tem ali je komentar pozitiven, informacijo o pravilnosti klasificije komentarjev in vrednost klasifikacije. Naredili smo tudi podatkovno strukturo HotelReviews. Ta je vsebovala ime hotela, seznam komentarjev (s pomočjo prej definirane strukture Review) in vrednost klasifikacije. Naredili smo seznam podatkovnih struktur HotelReviews, kamor smo si shranili podatke, ki smo jih kasneje pridobili iz testne množice podatkovne baze.

Prebrali smo testno množico podatkovne baze in si pri tem shranjevali za nas potrebne informacije – ime hotela in komentarje, ki smo jih že označili, ali so v podatkovni bazi bili prepoznani kot pozitivni ali negativni, saj nam je to kasneje pomagalo pri izračunu natančnosti klasifikacijskega modela. Podatke iz testne množice smo shranili v prej omenjene strukture. Ko smo imeli vse podatke shranjene v naših podatkovnih strukturah, smo se lotili klasifikacije komentarjev.

Za vsak komentar, ki je sestavljen iz sekvence n-gramov, smo vedno izračunali 2 vrednosti: klasifikacijsko vrednost, da komentar pripada pozitivnemu komentarju, in klasifikacijsko vrednost, da pripada negativnemu komentarju.

$$\textit{komentar} = \textit{n-gram1},\ \textit{n-gram2},\ \textit{...}\ \textit{n-gramN} \tag{2}$$

$$\begin{aligned} P(\textit{komentar}|\textit{pozitiven}) = \\ & \log(P(\textit{pozitiven})) \\ & + \log(P(\textit{n-gram1}|\textit{pozitiven})) \\ & + \log(P(\textit{n-gram2}|\textit{pozitiven})) \\ & ... \\ & + \log(P(\textit{n-gramN}|\textit{pozitiven})) \end{aligned} \tag{3}$$

$$\begin{aligned} P(\textit{komentar}|\textit{negativen}) = \\ & \log(P(\textit{negativen})) \\ & + \log(P(\textit{n-gram1}|\textit{negativen})) \\ & + \log(P(\textit{n-gram2}|\textit{negativen})) \\ & ... \\ & + \log(P(\textit{n-gramN}|\textit{negativen})) \end{aligned} \tag{4}$$

Če je bila pozitivna klasifikacijska vrednost večja od negativne, smo komentar klasificirali kot pozitiven, v nasprotnem primeru smo komentar klasificirali kot negativen.

Izračunano klasifikacijsko vrednost smo si shranili, pri čemer smo pri negativnem klasifikacijskemu rezultatu dodali negativen predznak – to nam je pomagalo v prihodnjih korakih pri določanju splošnega zadovoljstva hotelskih gostov, ki ga najdemo v komentarjih s pomočjo računanja točk posameznih hotelov. Več o tem sledi v naslednjem poglavju.

3.4 Računanje točk zadovoljstva hotelskega bivanja

Ko smo klasificirali vse komentarje posameznih hotelov, smo se posvetili ocenjevanju splošnega zadovoljstva, ki ga najdemo v komentarjih. Prvotna ideja je bila, da bi prešteli pozitivne in negativne komentarje, ter izračunali njihov delež v primerjavi z vsemi komentarji za posamezen hotel. Odločili smo se, da bomo naredili še en korak naprej in namesto štetja računali vrednosti klasifikacije, kar je točkam posameznih hotelov dodalo še dodatno težo. Ta teža se odraža pri komentarjih, ki smo jih klasificirali znotraj istega razreda – npr. 2 komentarja, ki sta klasificirana kot pozitivna se lahko razlikujeta v tem, da je eden bolj pozitiven kot drugi.

V prejšnjem koraku smo komentarjem, ki smo jih klasificirali za negativne, dodelili negativen predznak (ostali imajo pozitivnega, saj je verjetnost nečesa vedno pozitivna). Te klasifikacijske vrednosti smo nato sešteli za posamezen hotel in na tak način izračunali splošno zadovoljstvo glede hotela. Večja kot je bila vsota komentarjev posameznega hotela, bolj so bili gosti hotela, ki so komentarje napisali, zadovoljni. Za vsak hotel smo torej izračunali vrednost, ki predstavlja točke hotela, in v splošnem zajema zadovoljstvo. Nato smo hotele uredili glede na njihove izračunane točke ter na tak način dobili lestvico hotelov.

4 Eksperiment

V eksperimentu smo klasifikacijo izvedli na dva načina – z uporabo unigramov in bigramov. Oba klasifikacijska načina smo primerjali med sabo z izračunom natančnosti modela, preciznosti, priklicem in mero F1. Pri izračunu natančnosti, preciznosti in priklicu smo uporabili parameter tp (ang. true positive), ki predstavlja število pravilno klasificiranih pozitivnih komentarjev, tn (ang. true negative), ki predstavlja število pravilno klasificiranih negativnih komentarjev, fp (ang. false positive), predstavlja število napačno klasificiranih negativnih komentarjev in fn (ang. false negative), ki pove število napačno klasificiranih pozitivnih komentarjev.

$$Natancnost = \frac{tp + tn}{tp + tn + fp + fn} \tag{5}$$

$$Preciznost = \frac{tp}{tp + fp} \tag{6}$$

$$Priklic = \frac{tp}{tp + fn} \tag{7}$$

$$F1 = \frac{2 * Preciznost * Priklic}{Preciznost + Priklic} \tag{8}$$

Pri računanju metrik uspešnosti klasificikacij smo vrednosti izračunali tako za unigrame kot za bigrame, ter prišli do rezultatov, ki so prikazani v tabeli 1. Zraven naših rezultatov, so prikazani tudi rezultati iz literature [6], po kateri smo se zgledovali pri implementaciji algoritma.

Iz tabele 1 lahko razberemo, da se unigrami v vseh metrikah uspešnosti bolje obnesejo od bigramov. Unigrami imajo namreč boljši rezultat pri natančnosti, preciznosti, priklicu in pri uglašeni meri F1. Največjo razliko med unigrami in bigrami vidimo pri izračunu priklica. Na podlagi tega lahko sklepamo, da smo pri bigramih imeli veliko število napačno klasificiranih pozitivnih komentarjev.

Tabela 1: Metrike uspesnosti klasifikatorjev.

	Unigrami	Bigrami	Članek [6]
Natančnost	0,92	0,80	0,75
Preciznost	0,94	0,93	/
Priklic	0,91	0,67	/
Mera F1	0,92	0,78	/

Če pa primerjamo priklic in preciznost bigramov, vidimo, da so bigrami bili pri preciznosti bolj uspešni kot pri priklicu. Iz slednjega lahko sklepamo, da je število napačno klasificiranih nagativnih komentarjev bilo manjše kot število napačno klasificiranih pozitivnih komentarjev. Pri unigramih pa je razlika med priklicom in preciznostjo dokaj mala, zato lahko sklepamo, da je delež napačno klasificiranih komentarjev približno enako razporejen med napačno klasificiranimi pozitivnimi komentarji (fn) in napačno klasificiranimi negativnimi komentarji (fp).

V članku [6] so za klasifikacijo komentarjev videov Danmaku uporabili slovar sentimenta, ki ni temeljil zgolj na besedišču komentarjev. Pri klasifikaciji komentarjev so uporabili bigrame, torej lahko njihove rezultate primerjamo z našimi rezultati pri pristopu z bigrami. Dosegli smo boljše rezultate (tabela 1) zaradi uporabe slovarjev, ki so bolj domensko specifični.

Kot zanimivost smo ustvarili tudi vizualno predstavitev unigramov (besed), ki se najpogosteje pojavijo v pozitivnih in negativnih komentarjih. Vizualno predstavitev vidimo na sliki 1, pri čemer zelena slika predstavlja pogoste besede, ki jih najdemo v pozitivnih komentarjih, rdeča slika pa prikazuje pogoste besede znotraj negativnih komentarjev.

Comfort
Perfect
Ideal
Best
Right
Love
Super
Fabulous
Excellent
Good
Fantastic
Convenient
Nice
Clean
Wonderful
Friendly
Great
Cleanliness
Tired
Poor
Price
Slow
Small
Check
Broken
Busy
Need
Loud
Wait
Bad
Problem
Expensive
Terrible
Look
Dirty
Smell
Disappointed
Noisy
Money

Slika 1: Vizualna predstavitev najpogostejsih besed v komentarjih

5 Zaključek

V okviru naše naloge smo uporabili analizo sentimenta nad komentarji hotelskih gostov. Klasifikacijo komentarjev smo izvedli s pomočjo slovarjev in metode Naivni Bayes, pri čemer smo eksperiment izvedli s pomočjo unigramov in bigramov. Končni rezultati so pokazali, da je klasifikacija s pomočjo unigramov uspešnejša, kot pa z bigrami. Pri klasifikaciji s pomočjo unigramov smo dosegli 92 % natačnost, pri bigramih pa 80 %. V članku [6] so dosegli 75 % natančnost, torej so bili manj uspešni pri klasifikaciji. Boljše rezultate napram primerjalnega članka smo dosegli zaradi uporabe slovarjev sentimenta, ki temeljijo zgolj na besedišču, ki ga najdemo v komentarjih, med tem ko so v [6] uporabljali bolj splošen slovar.

Klasifikacijo komentarjev smo v našem članku uporabili za raziskavo splošnega zadovoljstva hotelskih gostov, ki so napisali komentarje za določen hotel. Eksperiment bi lahko v nadaljnem delu razširili z upoštevanjem nacionalnosti piscev komentarjev pri sami klasifikaciji. Na tak način bi lahko ugotovili korelacijo med nacionalnostjo piscev komentarjev in zadovoljstva glede določenega hotela, ki se skriva v komentarjih. Posameznim komentarjem bi v bodoče lahko dodali tudi številčno oceno (npr. od 1 do 5), ki bi jo pridobili s pomočjo razširitve trenutnega eksperimenta.

Literatura

[1] Bouazizi, M., and Otsuki Ohtsuki, T. A pattern-based approach for sarcasm detection on twitter. *IEEE Access 4* (2016), 5477–5488.

[2] Brandwatch, K. B. Understanding sentiment analysis: What it is & why it's used.

[3] Cai, R., Qin, B., Chen, Y., Zhang, L., Yang, R., Chen, S., and Wang, W. Sentiment analysis about investors and consumers in energy market based on bert-bilstm. *IEEE Access 8* (2020), 171408–171415.

[4] Fang, Y., Tan, H., and Zhang, J. Multi-strategy sentiment analysis of consumer reviews based on semantic fuzziness. *IEEE Access 6* (2018), 20625–20631.

[5] Jianqiang, Z., Xiaolin, G., and Xuejun, Z. Deep convolution neural networks for twitter sentiment analysis. *IEEE Access 6* (2018), 23253–23260.

[6] Li, Z., Li, R., and Jin, G. Sentiment analysis of danmaku videos based on naïve bayes and sentiment dictionary. *IEEE Access 8* (2020), 75073–75084.

[7] Phan, H. T., Tran, V. C., Nguyen, N. T., and Hwang, D. Improving the performance of sentiment analysis of tweets containing fuzzy sentiment using the feature ensemble model. *IEEE Access 8* (2020), 14630–14641.

[8] Saad, S. E., and Yang, J. Twitter sentiment analysis based on ordinal regression. *IEEE Access 7* (2019), 163677–163685.

[9] Son, L. H., Kumar, A., Sangwan, S. R., Arora, A., Nayyar, A., and Abdel-Basset, M. Sarcasm detection using soft attention-based bidirectional long short-term memory model with convolution network. *IEEE Access 7* (2019), 23319–23328.

[10] Watanabe, H., Bouazizi, M., and Ohtsuki, T. Hate speech on twitter: A pragmatic approach to collect hateful and offensive expressions and perform hate speech detection. *IEEE Access 6* (2018), 13825–13835.

[11] Xu, G., Meng, Y., Qiu, X., Yu, Z., and Wu, X. Sentiment analysis of comment texts based on bilstm. *IEEE Access 7* (2019), 51522–51532.

Časovni razporejevalniki in brezstrežniško okolje

Uroš Zagoranski
Univerza v Mariboru,
Fakulteta za elektrotehniko, računalništvo in informatiko,
Koroška cesta 46, 2000 Maribor, Slovenija
uros.zagoranski@student.um.si

Povzetek

V prispevku smo se osredotočili na časovne razporejevalnike (ang. cron job schedulers) v brezstrežniškem (ang. serverless) okolju in njihovo zanesljivo uporabo. Primerjali smo razporejevalnike, implementirane s pomočjo zabojnikov s tistimi, ki so gostovani v oblaku z uporabo pristopa funkcije kot storitve. Ugotavljali smo, katere so posebnosti časovnih razporejevalnikov v brezstrežniškem okolju in kdaj je le-te sploh smiselno uporabiti. Na praktičnem primeru smo predstavili, kako jih lahko vključimo v večji sistem in na kakšen način najlažje rešimo morebitne težave, ki jih ob izbiri brezstrežniškega okolja zavestno prevzamemo. Ugotovili smo, da so razporejevalniki v FaaS (ang. Function as a Service) okolju najprimernejši zaradi enostavnega in hitrega razvoja ter nizkih stroškov obratovanja.

Ključne besede brezstrežniška arhitektura · sodobni pristopi implementacije informacijskih rešitev · časovni razporejevalniki · funkcija kot storitev · zabojnik kot storitev

1 Uvod

Tempo življenja se v zadnjih letih veča in zaradi tega smo ljudje vse bolj časovno obremenjeni. S tem se posledično veča tudi potreba po časovnem razporejanju rutin, da se kakšna informacija v presežku le-teh ne izgubi. Že od nekdaj so se razvijali sistemi, ki so omogočali takšno in drugačno razporejanje, a to ni bilo še nikdar enostavnejše, kot je v današnji dobi računalništva v oblaku. Velik nabor ponudnikov oblačnih storitev omogoča izbiro med le-temi, ki pa ravno zaradi velike ponudbe včasih ni trivialna. Izbiro prave storitve otežuje predvsem pomanjkanje analitičnih pregledov in primerjav storitev, ki jih ponujajo različne korporacije. S tem namenom smo želeli predstaviti razlike, prednosti in dodatne izzive, ki nastanejo pri razvoju razporejevalnikov v oblaku z uporabo funkcij kot gonilne sile v primerjavi s tistimi, ki so implementirani v (bolj) klasičnih aplikacijah, s pomočjo orkestracije zabojnikov. Poleg tega smo našteli in predstavili glavne sisteme, ki ponujajo časovno razporejanje v brezstrežniškem okolju in na praktičnih primerih uporabe pojasnili, kateri izmed glavnih ponudnikov je za posamezen primer boljši in zakaj.

Raziskovalni vprašanji, ki smo ju obravnavali sta:

- **Katere so posebnosti časovnih razporejevalnikov v brezstrežniškem okolju?**
- **Kdaj je smiselno uporabiti časovne razporejevalnike brezstrežniškega okolja?**

V nadaljevanju članka so v drugem poglavju predstavljene posebnosti sodobnega pristopa razvoja programskih rešitev, to je z zabojniki in s pomočjo funkcij kot storitev, ter njuna primerjava. Tretje poglavje pokriva tematiko časovnih razporejevalnikov na splošno ter se podrobneje posveča razporejevalnikom v brezstrežniškem okolju, pregledno povzema glavne ponudnike takšnih storitev in navaja primere uporabe. Četrto poglavje predstavlja realen primer uporabe časovnih razporejevalnikov v namen razporejanja opravil v računovodskem servisu. V petem poglavju odgovarjamo na raziskovalna vprašanja in uvajamo diskusijo, v zadnjem poglavju pa povzemamo rezultate raziskave in glavna nova dognanja, pridobljena z njeno pomočjo.

2 Sodobni pristopi implementacije informacijskih rešitev v namen razporejanja opravil

Vse več razvijalcev informacijskih rešitev pomeni večji nabor možnosti in preferenc pri razvoju programske opreme. Zaradi tega raziskovalno podjetje Gartner vsako leto izdela poročilo trendov v informacijsko tehnološkem svetu za prihodnje koledarsko leto. V zadnjem času lahko na tej lestvici vse pogosteje zasledimo besedne zveze kot so umetna inteligenca, obogatena resničnost, analitika, avtomatizacija ipd. [12] Virtualizacija virov in programiranje v oblaku sta se hitro zlili s klasičnimi pristopi implementacije informacijskih rešitev, zato ni čudno, da je rast tega področja in inovativnost rešitev v tej kategoriji v pozitivnem trendu. Med napovedmi za bližnjo prihodnost se v Gartnerjevih raziskavah tako pojavljajo besedne zveze, ki omenjajo zabojnike, njihovo orkestracijo, funkcije kot storitev in brezstrežniški pristop razvoja programske opreme splošno. [6]

Brezstrežniški pristop razvoja programske opreme je model računalništva v oblaku, v katerem ponudnik oblačnih storitev v imenu svojih strank skrbi za strežnike in dodeljevanje strojnih virov. Glavna posebnost takšnega pristopa je, da aplikacija nima dodeljenih računalniških virov, če ta ni v uporabi. To se odraža v nižji ceni delovanja, saj le-ta temelji na dejanski količini sredstev, ki jih aplikacija porabi v določenem časovnem intervalu. [1]

Trenutno stanje brezstrežniških sistemov na trgu je v Gartnerjevih analizah primerjano s predpostavkami za bližnjo prihodnost, prav tako pa je podana finančna ocena

DOI https://doi.org/10.18690/978-961-286-516-0.16
ISBN 978-961-286-516-0

brezstrežniških sistemov v bližnji prihodnosti. Pri tem napovedujejo, da se bo iz lanske globalne kvote prihodkov v višini 465,8 milijonov dolarjev do leta 2024 višina prihodkov kar podvojila, in sicer naj bi znašala 944 milijonov dolarjev. Največjo rast naj bi po ocenah dosegli ravno orkestracija zabojnikov ter brezstrežniški pristop razvoja informacijskih rešitev, kateri tematiki bomo podrobneje opisali v nadaljevanju. [11]

Druga Gartnerjeva raziskava ocenjuje, da bo v prihodnosti drastično narasla tudi uporaba platform, ki kot sprožilce aktivnosti v sistemu uporabljajo najmanjše možne komponente, to so funkcije. Trenutno približno 20 odstotkov globalnih IT podjetij uporablja »funkcijski« (FaaS) način implementacije, do leta 2025 pa bi naj ta številka narasla na kar 50 odstotkov. [4]

Porast interesa uporabe oblačnih infrastruktur se odraža tudi na številu ponudnikov. Med vodilne uvrščamo naslednje:

- Amazon Web Services,
- Google Cloud Platform,
- Microsoft Azure,
- IBM Apache OpenWhisk.

V tekmovanju teh ponudnikov se kot glavni akter že vrsto let pojavlja platforma Amazon Web Services (v nadaljevanju AWS), a v zadnjem času pridobiva vse več tekmecev, med katerimi sta najbolj aktualna predvsem Microsoft Azure in Google Cloud Platform (v nadaljevanju GCP). A če povzamemo obe prej omenjeni področji (zabojnike in funkcije), v ospredje brez konkurence stopi Google, saj si lasti tako orkestracijsko rešitev Kubernetes kot tudi ogromno platformo za programiranje v oblaku, GCP. [7] Podjetja želijo, da se razvijalci osredotočajo predvsem na razvoj programske opreme, ne pa tudi na vzdrževanje, vzpostavitev in druge podporne aktivnosti. Pri zabojnikih zato pogosto zasledimo tudi osebo, ki je zadolžena za skrb in upravljanje s celotno arhitekturo (DevOps), katere pa pri funkcijskem pristopu načeloma ne potrebujemo.

2.1 Zabojniki: Container as a Service (CaaS)

Vse pogostejši trend uporabe zabojnikov za implementacijo informacijskih rešitev se odraža tudi pri nastanku dokaj nove paradigme računalništva. CaaS predstavlja nadgradnjo kategorije IaaS (Infrastructure as a Service), kjer korporacije v najem ponujajo strežnike, virtualne stroje, omrežja in shrambe, pri čemer so njihovi uporabniki odgovorni za upravljanje infrastrukture in nameščanje aplikacij nanjo. Zraven tega CaaS ponuja še zabojniške stroje in funkcionalnosti za orkestracijo le-teh. [19] Za nas je najzanimivejši podrazred CaaSa, KaaS (Kubernetes as a Service), saj je zaradi svoje robustnosti, zrelosti in bogatega nabora funkcionalnosti Kubernetes postal de-facto standard in najpogosteje uporabljena tehnologija, ko je govora o razvoju informacijskih rešitev s souporabo zabojnikov. [13]

Kubernetes je odprtokodni sistem za orkestracijo, ki omogoča avtomatizirano upravljanje z zabojniki, prav tako pa upravlja celoten življenjski cikel zabojnikov, pri čemer uporabniki upravljajo z izvedbo in interakcijo aplikacij. [8] Aplikacije so tipično sestavljene iz več zabojnikov, ki so porazdeljeni po različnih virtualnih in fizičnih gostiteljih. Kubernetes nam pri tem pomaga pri upravljanju s strukturiranjem zabojnikov v tako-imenovane stroke (ang. pods), ki okolje bogatijo z dodatnim nivojem abstrakcije. Na ta način je poenostavljeno razporejanje z viri, prav tako pa časovno razporejanje, ki je za nas najzanimivejše. [10]

Pristop z zabojniki je primeren predvsem za ekipe, ki prisegajo na podrobnejši nadzor nad storitvami kontejnerizacije. Prav tako omogoča avtomatizacijo uvajanja in vračanja v prejšnje stanje, fine nastavitve količine procesorske moči in pomnilniškega prostora, samodejno celjenje zabojnikov, ki so bili pri izvajanju neuspešni ter upravljanje s konfiguracijami. [8]

2.2 Funkcije: Function as a Service (FaaS)

FaaS je nov koncept v oblačnem računalništvu, podoben PaaSu (Platform as a Service) z vidika omogočanja enostavnejšega razvoja programske opreme in razbremenitve razvijalcev upravljanja s strežniki in operacijskimi sistemi. Prednost funkcij v oblaku v primerjavi s prej omenjenima konceptoma je predvsem v poslovnem vidiku. Medtem, ko se pri PaaSu plačuje čas izvajanja niti, se pri FaaSu plačuje izvajalni čas specifične funkcije. S tem se lahko konkretno zvišajo performančne karakteristike ogromnih sistemov, hkrati pa znižajo obratovalni stroški same informacijske rešitve. [1]

Z začetki v letu 2014 z Amazonovo Lambdo je do danes koncept FaaS postal priznan in dobro sprejet način implementacije informacijskih rešitev. Dokazano je veliko bolj skalabilen, elastičen, razvijalcem prijazen ter cenovno ugoden kot predhodne oblačne arhitekture. Poleg AWSove Lambde v to kategorijo uvrščamo tudi Microsoft Azure Functions, Google Cloud Functions, IBM Apache OpenWhisk. [2]

Pristop s funkcijami v oblaku je primeren predvsem za projekte, ki imajo neenakomerno razporeditev izvajanja, izjemno visoke ali neznane zahteve za skaliranje sistema, so vezani na zunanje dogodke, sestavljeni iz kratkoživečih diskretnih funkcij, ali pa lahko s klici operirajo brez stanja. Pogosto jih lahko zasledimo pri mobilnih aplikacijah, IoT senzorskih komponentah in časovno aktiviranih akcijah. [4]

2.2.1 Primerjava obeh skupin

V začetku porasta informacijske tehnologije je postala popularna gradnja monolitnih aplikacij, ki pa se danes vse manj pojavlja na seznamu trendov. V zadnjem času so jih pretežno zamenjale mikro-storitve in posledično se je vpeljala uporaba zabojnikov, s katerimi sta razporejanje in razvoj aplikacij veliko enostavnejša. Zabojniki so se v IT svetu prvič pojavili že pred letom 2000, zato ima ta koncept daljšo in bogatejšo zgodovino, kot funkcije v oblaku. Kljub temu t. i. eksplozija popularnosti obeh konceptov sega v podobno časovno obdobje. Zabojniki so namreč postali izjemno aktualni šele z Dockerjem v letu 2013, FaaS pa z AWSovo Lambdo v letu 2014. [16]

FaaS predstavlja višji nivo abstrakcije kot CaaS. Slednji namreč abstrahira nivoje strojne opreme, virtualizacij-

Tabela 1: Primerjava CaaS in FaaS [9]

	CaaS	**FaaS**
Prenosljivost	Zabojniki predstavljajo standardizirano enoto, zaradi česar je njihova prenosljivost med ponudniki enostavna	Slaba standardizacija, večinoma odvisno od ponudnika, razporejevalniki kljub temu delno standardizirani
Dvig in premik	Obstoječe aplikacije enostavno oviti v zabojnike, če to ni že izvedeno, zabojniki nimajo vpliva na delovanje razporejevalnikov	Ni mogoč, obstoječe aplikacije potrebujejo adaptacijo ali popolno re-implementacijo
Razdrobljenost	Enote ponavadi razdeljene na komponente, ponujena možnost razporejanja specifičnih funkcij, tudi tistih, ki skrbijo za razporejanje	Aplikacija razdrobljena na majhne enote, ki jih je enostavno razporejati (funkcije), idealno za implementacijo razporejevalnikov
Izvajanje	Upravljano s strani razvijalca	Upravljano s strani ponudnika v oblaku
Fleksibilnost	Visoka	Srednja
Primeri uporabe	Celotne spletne aplikacije, mikro-storitve, upravljanje s podatki	Specializirane funkcije, asinhrono procesiranje, dogodkovno vodena arhitektura
Izzivi	Zahteva razumevanje delovanja zabojnikov, težja implementacija razporejevalnikov	Upravljanje odvisnosti, sledenje in upravljanje več mikro-storitev hkrati
Primernost	Učinkovito za gostovanje osnovnih aplikacij, manj primerno za implementacijo razporejevalnikov	Najprimernejše za lansiranje novih aplikacij, izjemno primerno za implementacijo razporejevalnikov

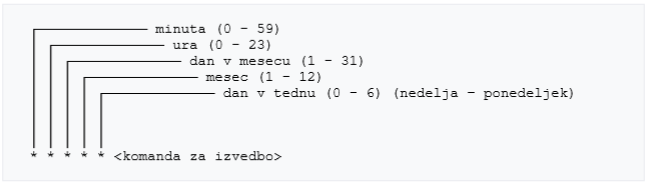
skih strojev ter operacijskega sistema, pri čemer pa prvi na vse prej omenjene nivoje dodaja še abstrakcijo izvedbe programske kode in aplikacije. Obe skupini prenašata odgovornost manipulacije z aplikacijo in funkcijami na uporabnika storitve.

Če na kratko povzamemo celotno tabelo 1, ki predstavlja primerjavo prej omenjenih pristopov lahko ugotovimo, da v splošnem pristop z zabojniki nudi več svobode pri implementaciji, nadzora nad izvajanjem ter enostavno migracijo v primeru, da neka obstoječa aplikacija še ne uporablja takšnega pristopa. Na drugi strani je pristop s funkcijami v oblaku primernejši in lažji za vzdrževanje, je pa manj fleksibilen in slabo standardiziran, zato se tehnične podrobnosti, ki jih je potrebno upoštevati ob implementaciji, razlikujejo med posameznimi ponudniki. Kateri pristop uporabiti je odvisno predvsem od primera uporabe, pri čemer je za spletne aplikacije z mikro-storitvami primernejša izbira CaaS modela, za dogodkovno vodene aplikacije z asinhronim procesiranjem pa se bo bolje izkazal FaaS model.

3 Časovni razporejevalniki

Časovno tempiranje aktivnosti je v današnjem svetu čisto običajna aktivnost, zato je vse pogostejša tudi uporaba sistemov, ki skrbijo za izvajanje določenih funkcij ob specifičnem času, na željeno frekvenco ali pa z določenimi zamiki. Razporejevalniki nas tako v informacijsko tehnološkem svetu spremljajo praktično od začetka. Ti so prisotni v operacijskih sistemih, omrežnih komponentah, konec koncev pa tudi v I/O operacijah (operacijah pisanja in branja).

Medtem, ko se pri vseh prej naštetih komponentah termin razporejanje pojavlja v obliki razporejanja virov, je za našo raziskavo veliko zanimivejše časovno razporejanje opravil. V tradicionalnih sistemih se takšni problemi večinoma rešujejo z aplikacijskimi strežniki in orkestracijo zabojnikov, v novi dobi oblačnega računalništva pa se vse pogosteje pojavlja prej omenjeni pristop FaaS. Za izvedbo specifičnih funkcij ob določenem času v informacijskih rešitvah takšnega tipa skrbijo funkcije, imenovane časovni razporejevalniki, ki v vseh sistemih, ne glede na izbiro tehnološkega sklada, delujejo po enakem principu. Poleg enakega principa imajo tudi enako obliko sprejema podatkov, sestavljeno iz petih spremenljivk in klica funkcije ali komande. Splošna oblika »cron« izraza je prikazana na sliki 1.


```
┌───────────── minuta (0 - 59)
│ ┌─────────── ura (0 - 23)
│ │ ┌───────── dan v mesecu (1 - 31)
│ │ │ ┌─────── mesec (1 - 12)
│ │ │ │ ┌───── dan v tednu (0 - 6) (nedelja - ponedeljek)
│ │ │ │ │
│ │ │ │ │
* * * * * <komanda za izvedbo>
```

Slika 1: Splošna oblika »cron« izraza

Med najpogostejšimi izrazi, ki jih zasledimo v imenih razporejevalnikov sta angleški besedi »cron« in »scheduler«. V sklopu prej opisanih razporejevalnikov najdemo glavne ponudnike takšnih storitev, to so Googlov Kubernetes, Amazonovo orodje Lambda v kombinaciji s storitvijo CloudWatch Events, Googlovi Cloud Scheduler in Scheduled Cloud PubSub ter Microsoftov Azure Time Trigger.

3.1 »Zabojniško« okolje

Zabojnike lahko uporabimo za različne namene, eden pomembnejših je procesiranje in obdelava nalog v ozadju. To je pogost primer uporabe, ki pa je vse prej kot trivialen in odpira vrsto vprašanj, med drugim je vprašljivo sledenje izvajanja, skaliranje in upravljanje s ponovnimi zagoni. Zraven razporejevalnikov z viri, ki so v razdelani arhitekturi z orodji kot sta Docker in Kubernetes neizbežni, se v teh okoljih srečujemo tudi s časovnimi razporejevalniki, ki znatno vplivajo na optimizacijo virov, da je uporaba teh kar se da učinkovita. [14]

Zabojniki sami po sebi omogočajo uporabo časovnih sprožilcev, ki pa z vpeljavo več instanc in posledično kompleksnosti zahtevajo uporabo posvečenega orodja, kot je na primer Kubernetes. Stanje posameznega zabojnika se tako ne more prenašati med preostalimi zabojniki zaradi varnostnih razlogov, zaradi česar je potrebno najti druge načine za izpostavitev skript. Kubernetes prav tako dinamično generira imena zabojnikov, zaradi česar je dostopanje do njih oteženo. [17]

3.2 »Funkcijsko« okolje

Časovni razporejevalniki sami po sebi ne veljajo za najbolj zanesljive. To pride do izraza v funkcijskem brezstrežniškem okolju, kjer jih pogosto najdemo kot popolnoma neodvisne komponente. Njihova edina naloga je, da se izvedejo ob določenem času, zaradi česar se lahko zgodi, da se določena funkcija, ki je bila klicana v sklopu nekega razporejevalnika, izvede neuspešno. Do tega lahko pripelje nekaj dejavnikov, med drugim začasna prekinitev delovanja infrastrukture (ki je v oblaku redka, a mogoča), neuspešna obdelava podatkov, ali pa prekinitev povezave s podatkovno bazo. Ker razporejevalniki v brezstrežniškem okolju kot parametre prejmejo le čas izvedbe, ni potrebna validacija vnesenih parametrov. Iz tega vidika so izjemno zanesljivi, saj (načeloma) ne more priti do napake pri sami izvedbi. Večjo težavo predstavlja dejstvo, da se predpisane funkcije po izvedbi preprosto »ugasnejo« in jih uspešnost zaključka klicane funkcije ne zanima. [15]

Med glavne primere uporabe lahko uvrstimo naslednje:

- vzdrževalna naloga re-indeksiranja podatkovne baze, ki se mora izvesti n-krat v določenem časovnem intervalu,
- deaktivacija poteklih uporabniških računov,
- nadzor nad prostorom pomnilnika,
- kreacija varnostnih kopij občutljivih podatkov,
- upravljanje s predpomnilnikom,
- razpošiljanje glasila ali posebne ponudbe prijavljenim uporabnikom,
- periodično preverjanje slepih povezav na spletni strani,
- kodiranje videov in večjih datotek v ozadju.

Iz navedenega lahko vidimo, da nam časovni razporejevalniki pomagajo pri marsikaterem rutinskem opravilu, ki bi bilo z izbiro drugačnega pristopa razvoja programske opreme zahtevnejše za implementacijo. Kot že omenjeno lahko za pristop k rešitvi problema izbiramo iz širokega spektra ponudnikov, zato sta v naslednjih podpoglavjih predstavljena dva najpomembnejša, GCP Cloud Scheduler in AWS CloudWatch Events.

3.2.1 Google Cloud Platform – Cloud Scheduler

Podjetje Google ponuja veliko storitev, ki so strnjene v ogromno platformo, imenovano Google Cloud Platform. Če se za implementacijo časovnega razporejanja rutinskih opravil odločimo za GCP, je potrebno za dosego cilja uporabiti kombinacijo dveh storitev, to je Cloud Functions in Cloud Scheduler. Slednja sama po sebi uporablja vse, kar prva ponuja, zato nam za povezavo med njima ni treba skrbeti.

Cloud Scheduler je v celoti upravljan razporejevalnik, ki omogoča razporejanje paketnih opravil, opravil z velikimi podatki ter operacij z oblačno infrastrukturo. Glavna prednost izbire tega razporejevalnika je zanesljivost, saj njegovo distribuirano infrastrukturo upravlja Google. S tem zagotavlja vsaj-enkratno dostavo sporočil na cilj, hkrati pa poenostavlja koncept »crontabov«, saj omogoča specifikacijo želenega urnika preko t. i. »cron« izraza. Zraven tega ponuja tudi močno orodje za beleženje izvedbe in uspešnosti ter enostavno konfiguracijo pravilnika za ponovni poskus v primeru napak. V osnovi so lahko na posameznem računu brezplačno gostovane tri instance razporejevalnika, vsaka naslednja instanca pa stane 0,10 dolarjev na mesec. [5]

3.2.2 Amazon Web Services – CloudWatch Events

Podobno kot Google tudi Amazon lasti platformo, ki ponuja velik nabor storitev, to je Amazon Web Services. V njenem sklopu lahko najdemo storitev AWS Lambda, ki sama po sebi ne ponuja časovnega razporejanja. Za dosego tega je Lambdo potrebno kombinirati z eno od treh storitev, ki to omogočajo: CloudFormation, CloudWatch Events in EventBridge, ki je posodobljena verzija storitve CloudWatch Events. Medtem ko AWS Lambda ponuja le storitve za oddaljeno izvajanje funkcij, se preostale tri storitve osredotočajo na sistemski tok dogodkov v realnem času. Storitev CloudWatch ostaja v tako imenovanem stanju pripravljenosti in ob določenih sprožilcih, kot so na primer časovni sprožilci, reagira. Slednja omogoča specifikacijo pravil za zagon funkcij, na primer na n-minut, dni, ali pa uporabi kompleksnejše razporejanje, za katero se prav tako uporablja t. i. »cron« izraz. [3]

Storitev CloudFormation v kombinaciji z AWS SAM (Serverless Application Model) omogoča tudi večstopenjsko izvajanje rutin, pri čemer je pisanje SAM predlog za brezstrežniško izvajanje povsem neboleče. Zaradi odlične podpore integracije različnih storitev znotraj AWS platforme je pri implementaciji razporejevalnikov meja le nebo. [18]

3.2.3 Primerjava predlaganih razporejevalnikov

Oba prej omenjena ponudnika sta razvila vrhunski rešitvi za časovno razporejanje, a vendar ima vsaka svoje prednosti.

Medtem ko GCP Cloud Schedule ponuja 3 brezplačne instance razporejevalnika AWS CloudWatch Events zagotavlja »zastonjsko« različico z maksimalno 100 zagoni na mesec. Opazna razlika je tudi v ceni - medtem, ko slednji za 10.000 zagonov ponuja ceno 0.30 dolarjev na mesec, GCP zagotavlja fiksno ceno 0.10 dolarjev na vsako dodatno instanco razporejevalnika. Dokumentacija Googlove storitve je napisana nekoliko bolj površinsko, Amazonova pa je podprta z ogromno primeri uporabe. Pri kompleksnosti v ospredje stopi Google, saj je povezovanje te storitve z drugimi iz nabora GCP trivialno, AWSove storitve pa so nekoliko zahtevnejše za integracijo. Hitrost izvedbe je pri obeh ponudnikih izjemno visoka, zato med njima v tem aspektu praktično ni razlike. Če na kratko povzamemo primerjavo je v primeru potrebe po več različnih časovnih razporejevalnikih z majhnimi frekvencami zagona ter za eksperimentalne namene bolje izbrati AWS, v primeru fiksnega (manjšega) števila razporejevalnikov in točno zadanih ciljev kaj z razporejevalniki želimo doseči pa GCP.

4 Primer uporabe

Za bolj plastično predstavo delovanja časovnih razporejevalnikov v funkcijskem okolju uporabimo primer predstavljenega računovodskega servisa na sliki 2 (na naslednji strani), ki skrbi za pošiljanje plačilnih listov vseh zaposlenih v podjetjih s katerimi sodeluje. Servis želi popolno avtomatizacijo pošiljanja plačilnih listov preko e-pošte. Ker e-poštna sporočila vsebujejo kritične podatke za zaposlene je potrebno zagotoviti, da so ta sporočila stoodstotno poslana. To lahko dosežemo s kombinacijo razporejevalnikov, ki omogočajo časovno tempiranje sporočil in sporočilnih sistemov, ki ponujajo zagotovljeno pošiljanje sporočil. BPMN diagram iz slike 2 je zasnovan splošno, zato ga je mogoče implementirati enotno, ne glede na izbiro ponudnika funkcijske arhitekture.

Z zahtevo po zanesljivosti naraste tudi kompleksnost zasnovane arhitekture. Kot že omenjeno, lahko pride med izvajanjem zgoraj opisanega procesa do motenj v delovanju. Z zankami je tako predstavljeno lovljenje napak znotraj objave sporočil na temo. V primeru, da instanca med izvajanjem kode na prvi ali drugi temi pade, se po ponovnem zagonu stanje zaradi shranjevanja vmesnih korakov vedno ohrani. S tem je doseženo, da se vsa sporočila zagotovo pošljejo, prav tako pa, da se vsa sporočila pošljejo natanko enkrat. V ta primer uporabe je mogoče vključiti tudi druge prej omenjene možnosti. Ker plačilni listi veljajo za občutljive podatke, bi računovodski servis lahko razporejevalnike uporabil za kreacijo njihovih varnostnih kopij. Podobno bi se lahko izvedla deaktivacija zaposlenih, ki v sistemu več niso aktivni, ali pa, v primeru dodatnih funkcionalnosti v sistemu, omogočilo kodiranje večjih datotek v ozadju.

5 Diskusija

Kot lahko razberemo iz drugega poglavja, je pristopov k implementaciji časovnih razporejevalnikov rutin v oblačnih arhitekturah veliko, zato pri izbiri le-teh prevzamemo tudi nekaj posebnosti, ki jih je potrebno upoštevati pri implementaciji, izvedbi in vzdrževanju takšnih sistemov. Prav to nas je zanimalo v sklopu prvega raziskovalnega vprašanja, ki se glasi:

- **Katere so posebnosti časovnih razporejevalnikov v brezstrežniškem okolju?**

Posebnosti časovnih razporejevalnikov v brezstrežniškem okolju se delijo na tiste, ki za uporabnika predstavljajo prednosti ter tiste, ki zanj predstavljajo slabosti, kar je razvidno iz tabele 1. Med posebnosti, ki jih je potrebno upoštevati pri implementaciji, spadata nekoliko slabša podpora personalizacije ter pomanjkanje standardizacije. Iz tega razloga se je potrebno pri izbiri ponudnika podučiti o specifikah razporejevalnika, ki ga ponujajo. Razlike, ki jih zasledimo med ponudniki so predvsem finančne ter performančne. Kljub temu izbira brezstrežniškega okolja s funkcijskim pristopom, predstavljena v poglavju 3.2, prinaša veliko prednosti, med drugim enostavne modifikacije in nastavitev razporejevalnikov preko »cron« izraza ter zagotovljena vsaj enkratna izvedba funkcije, ki je poklicana v sklopu razporejevalnika, kar je pri klasičnih sistemih nekoliko težje doseči.

Med posebnosti lahko uvrstimo tudi to, da so razporejevalniki v brezstrežniškem okolju samostojna komponenta. To pomeni, da so ločeni od preostalega dela kode, zaradi česar se lahko v primeru težav z osnovno aplikacijo na njih še vedno zanesemo. Posebnost je tudi ta, da je ozke časovne omejitve težko doseči, najnižji razpon dveh izvedb je namreč zaradi možnosti časovnega prekrivanja pri izvedbi dveh klicev ena minuta. V sklopu prednosti, ki jih pridobimo z izbiro brezstrežniške arhitekture pa je smiselno odgovoriti tudi na drugo raziskovalno vprašanje, ki je bilo rdeča nit celotne vsebine članka:

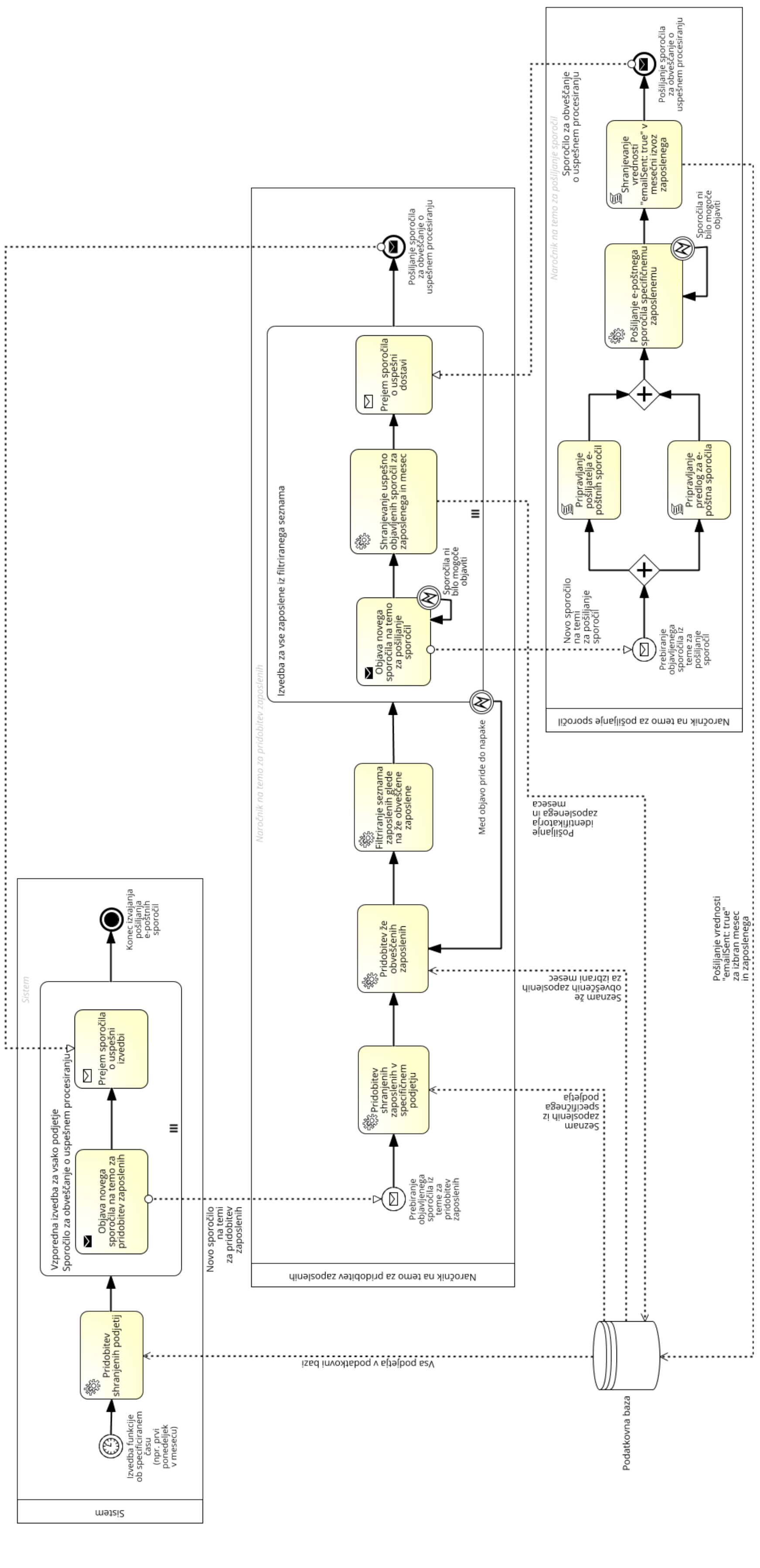
- **Kdaj je smiselno uporabiti časovne razporejevalnike brezstrežniškega okolja?**

Na prvi pogled razporejanje rutin ni nekaj, s čimer bi se srečevali pogosto. A realnost je precej drugačna, hitro lahko ugotovimo, da se časovne razporejevalnike uporablja za marsikatero opravilo, kar je razvidno iz nabora najpogostejših primerov uporabe iz poglavja 3.2. Smiselno jih je torej uporabiti predvsem za opravila, za katera želimo, da se izvedejo v ozadju in brez potrebe po poseganju upravljalca. Kot že omenjeno v poglavju 4 so to na primer vzdrževalne naloge re-indeksiranja podatkovne baze, deaktivacija poteklih uporabniških računov na nekem portalu, nadzor nad prostorom pomnilnika in upravljanje z njim, kreacija varnostnih kopij občutljivih podatkov, razpošiljanje e-poštnih sporočil določenim uporabnikom ter kodiranje večjih datotek v ozadju. Časovne razporejevalnike brezstrežniškega okolja z uporabo funkcij kot storitev je torej smiselno uporabiti vedno, ko je smiselno uporabiti vse druge časovne razporejevalnike, zraven tega pa imajo razporejevalniki tega tipa dodano vrednost hitre implementacije in malega vložka s strani uporabnikov, da storitev preverjeno deluje.

Izbira brezstrežniškega okolja pa ima še mnogo prednosti. Ena od teh je, da se razporejevalniki zaženejo le takrat, ko je nujno potrebno. Ravno nasprotno od tega lokalni razporejevalniki za nemoteno delovanje zahtevajo napravo, prižgano 24 ur na dan. Ločevanje komponent, v našem primeru razporejevalnikov, v našo kodo prinese svežino in olajša upravljanje s posameznimi deli projekta, saj so manjše enote veliko bolj obvladljive. Glavna prednost razporejevalnikov v brezstrežniškem okolju pa je vsekakor odgovor na vprašanje: zakaj bi ponovno implementirali nekaj, kar je že na voljo ter deluje zanesljivo in hitro.

6 Zaključek

V prispevku smo raziskovali pristope implementacije časovnih razporejevalnikov v brezstrežniškem okolju ter njihovo zanesljivo uporabo. Poglobili smo se v primerjavo


Sistem
Izvedba funkcije ob specificiranem času (npr. prvi ponedeljek v mesecu)
Pridobitev shranjenih podjetij
Vzporedna izvedba za vsako podjetje
Sporočilo za obveščanje o uspešnem procesiranju
Objava novega sporočila na temo za pridobitev zaposlenih
Prejem sporočila o uspešni izvedbi
Konec izvajanja pošiljanja e-poštnih sporočil
Novo sporočilo na temi za pridobitev zaposlenih
Vsa podjetja v podatkovni bazi
Podatkovna baza
Naročnik na temo za pridobitev zaposlenih
Prebiranje objavljenega sporočila iz teme za pridobitev zaposlenih
Pridobitev shranjenih zaposlenih v specifičnem podjetju
Seznam zaposlenih iz specifičnega podjetja
Pridobitev že obveščenih zaposlenih
Seznam že obveščenih zaposlenih za izbrani mesec
Filtriranje seznama zaposlenih glede na že obveščene zaposlene
Med objavo pride do napake
Izvedba za vse zaposlene iz filtriranega seznama
Objava novega sporočila na temo za pošiljanje sporočil
Sporočila ni bilo mogoče objaviti
Shranjevanje uspešno objavljenih sporočil za zaposlenega in mesec
Pošiljanje identifikatorja zaposlenega in meseca
Prejem sporočila o uspešni dostavi
Pošiljanje sporočila za obveščanje o uspešnem procesiranju
Naročnik na temo za pošiljanje sporočil
Prebiranje objavljenega sporočila iz teme za pošiljanje sporočil
Novo sporočilo na temi za pošiljanje sporočil
Pripravljanje pošiljatelja e-poštnih sporočil
Pripravljanje predlog za e-poštna sporočila
Pošiljanje e-poštnega sporočila specifičnemu zaposlenemu
Sporočila ni bilo mogoče objaviti
Shranjevanje vrednosti "emailSent: true" v mesečni izvoz zaposlenega
Sporočilo za obveščanje o uspešnem procesiranju
Pošiljanje sporočila za obveščanje o uspešnem procesiranju
Pošiljanje vrednosti "emailSent: true" za izbran mesec in zaposlenega

Slika 2: BPMN diagram zasnove razporejevalnika računovodskega sistema dveh predstavnikov brezstrežniškega okolja, zabojnikov in pristopa FaaS. Podrobneje smo spoznali glavna ponudnika funkcij kot storitev, AWS CloudWatch Events in GCP Cloud Scheduler, ki sama po sebi omogočata implementacijo in zanesljivo uporabo časovnih razporejevalnikov.

Z analizo literature in odgovorom na raziskovalni vprašanji, zastavljeni na začetku raziskave smo ugotovili, da je časovne razporejevalnike predvsem v okolju FaaS izjemno enostavno razviti, prav tako pa ne zahtevajo veliko vzdrževanja. Smiselno jih je uporabiti za rutinska opravila, s katerimi se nimamo časa ukvarjati, zato nas lahko razbremenijo, hkrati pa so cenovno izjemno dostopni. Če vse skupaj povzamemo je časovne razporejevalnike brezstrežniškega okolja torej priporočljivo uporabiti vedno, ko se v sklopu projekta pojavi potreba po reševanju ponavljajočih oziroma rutinskih opravil.

Literatura

[1] Alqaryouti, O., and Siyam, N. Serverless Computing and Scheduling Tasks on Cloud: A Review. *American Scientific Research Journal for Engineering, Technology, and Sciences (ASRJETS)* (mar 2018), 1–14.

[2] Aske, A. M. *SCHEDULING FUNCTIONS-AS-A-SERVICE AT THE EDGE*. PhD thesis, WASHINGTON STATE UNIVERSITY, Vancouver, may 2018.

[3] AWS. What Is Amazon CloudWatch Events? - Amazon CloudWatch Events, 2021.

[4] Costello, K. The CIO's Guide to Serverless Computing. *Gartner* (jun 2020).

[5] Google. Cloud Scheduler | Google Cloud, 2021.

[6] Iams, T. Gartner Blog Network. *Gartner* (may 2019).

[7] Kohgadai, A. 6 Container Adoption Trends of 2020 | StackRox. *StackRox* (mar 2020).

[8] Kubernetes. What is Kubernetes? | Kubernetes, feb 2021.

[9] Leger, Y., and Broshar, A. FaaS vs CaaS: Comparing Use Cases and Responsibilities, feb 2021.

[10] Marathe, N., Gandhi, A., and Shah, J. M. Docker swarm and kubernetes in cloud computing environment. In *Proceedings of the International Conference on Trends in Electronics and Informatics, ICOEI 2019* (apr 2019), vol. 2019-April, Institute of Electrical and Electronics Engineers Inc., pp. 179–184.

[11] Moore, S. Gartner Forecasts Strong Revenue Growth for Global Container Management Software and Services Through 2024. *Gartner* (jun 2020).

[12] Panetta, K. Gartner Top Strategic Technology Trends for 2021. *Gartner* (oct 2020).

[13] Pereira Ferreira, A., and Sinnott, R. A performance evaluation of containers running on managed kubernetes services. In *Proceedings of the International Conference on Cloud Computing Technology and Science, CloudCom* (dec 2019), vol. 2019-December, IEEE Computer Society, pp. 199–208.

[14] Rodriguez, M. A., and Buyya, R. Container-based cluster orchestration systems: A taxonomy and future directions. *Software - Practice and Experience 49*, 5 (may 2019), 698–719.

[15] Singhvi, A., Houck, K., Balasubramanian, A., Danish Shaikh, M., Venkataraman, S., and Akella, A. Archipelago: A Scalable Low-Latency Serverless Platform. Tech. rep., University of Wisconsin-Madison, Wisconsin, nov 2019.

[16] Tozzi, C. Why Is Docker So Popular? Explaining the Rise of Containers and Docker – Channel Futures, jul 2017.

[17] Walker, J. How to Use Cron With Your Docker Containers, jan 2021.

[18] Yilmaz, E. 3 Ways to Schedule AWS Lambda and Step Functions State Machine Executions , jan 2020.

[19] Zhang, R., Chen, Y., Zhang, F., Tian, F., and Dong, B. Be Good Neighbors: A Novel Application Isolation Metric Used to Optimize the Initial Container Placement in CaaS. *IEEE Access 8* (sep 2020), 178195–178207.

Proceedings of the 2021 7th Student Computer Science Research Conference (StuCoSReC)

IZTOK FISTER ET AL. (ED.)
University of Maribor, Faculty of Electrical Engineering and Computer Science, Maribor, Slovenia.
E-mail: iztok.fister@um.si

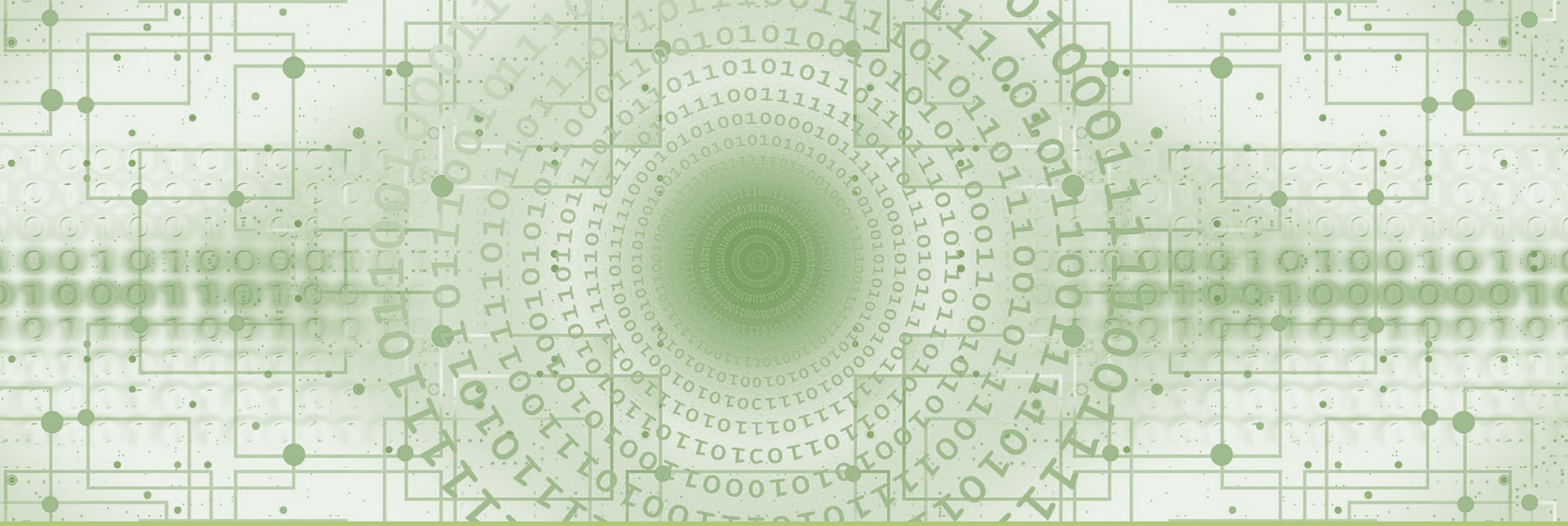
Abstract The 7th Student Computer Science Research Conference is an answer to the fact that modern PhD and already Master level Computer Science programs foster early research activity among the students. The prime goal of the conference is to become a place for students to present their research work and hence further encourage students for an early research. Besides the conference also wants to establish an environment where students from different institutions meet, let know each other, exchange the ideas, and nonetheless make friends and research colleagues. At last but not least, the conference is also meant to be meeting place for students with senior researchers from institutions others than their own.

Keywords: student conference, computer and information science, artificial intelligence, data science, data mining

University of Maribor Press

DOI https://doi.org/10.18690/978-961-286-516-0

ISBN 978-961-286-516-0



Student Computer Science Research Conference

StuCoSReC

University of Maribor

Faculty of Electrical Engineering and Computer Science

Univerza v Ljubljani

Università del Litorale

Univerza na Primorskem