StuCoSReC

Proceedings of the 2019 6th Student Computer Science Research Conference

University of Primorska Press

StuCoSReC
Proceedings of the 2019 6th Student Computer Science Research Conference

Edited by
Iztok Fister Jr., Andrej Brodnik, Matjaž Krnc and Iztok Fister

Reviewers and Programme Committee
Andrej Brodnik, Chair ▪ University of Primorska, Slovenia
Iztok Fister, Chair ▪ University of Maribor, Slovenia
Iztok Fister Jr., Chair ▪ University of Maribor, Slovenia
Matjaž Krnc, Chair ▪ University of Primorska, Slovenia
Nikolaj Zimic, Chair ▪ University of Ljubljana, Slovenia
Amina Alić ▪ University of Maribor, Slovenia
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
Patricio Bulić ▪ University of Ljubljana, Slovenia
Mojca Ciglarič ▪ University of Ljubljana, Slovenia
Jani Dugonik ▪ University of Maribor, Slovenia
Matjaž Gams ▪ Jozef Stefan Institute, Slovenia
Mario Gorenjak ▪ University of Maribor, Slovenia
Andres Iglesias ▪ Universidad de Cantabria, Spain
Sašo Karakatič ▪ University of Maribor, Slovenia
Branko Kavšek ▪ University of Primorska, Slovenia
Štefan Kohek ▪ University of Maribor, 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
Peter Rogelj ▪ University of Primorska, Slovenia
Damjan Vavpotič ▪ University of Ljubljana, Slovenia
Grega Vrbančič ▪ University of Maribor, Slovenia
Borut Žalik ▪ University of Maribor, Slovenia

Published by
University of Primorska Press
Titov trg 4, si-6000 Koper

Editor-in-Chief
Jonatan Vinkler
Managing Editor
Alen Ježovnik

Koper, 2019

ISBN 978-961-7055-82-5 (pdf)
www.hippocampus.si/ISBN/978-961-7055-82-5.pdf
ISBN 978-961-7055-83-2 (html)
www.hippocampus.si/ISBN/978-961-7055-83-2/index.html
DOI: https://doi.org/10.26493/978-961-7055-82-5

© University of Primorska Press

Kataložni zapis o publikaciji (CIP) pripravili v Narodni in univerzitetni knjižnici v Ljubljani
COBISS.SI-ID=302029568
ISBN 978-961-7055-82-5 (pdf)
ISBN 978-961-7055-83-2 (html)

Preface

Computer science is now among the most popular study programmes worldwide. We live in a digital age where most industries rely on data and software programmes. From transport infrastructure to public health systems, banking and communications, computer science is everywhere. Technology has made the world better, faster, and more connected. However, it is easy to miss an important component of this exciting success story.

Such development was made possible thanks to the brilliant minds of IT graduates, who took their passion for technology and used it to create ground breaking gadgets and computer programmes. Here in Slovenia, the three public universities share these values and invest heavily in their computer science students. These efforts facilitate collaboration among our departments, resulting in joint events such as this StuCoSRec student conference.

We are proud that, over the past five years, these Student Computer Science Research Conferences have grown in several ways. In this 6th installment, we received 24 full-paper submissions and one abstract submission. Among these, 21 papers were accepted to these proceedings, and 22 talks are scheduled to be presented during the conference, in three parallel sessions. The continued internationalization of our departments is also reflected with the authors of ten full-paper submissions originating from outside of Slovenia.

The conference is dedicated to graduate and undergraduate students of computer science and is therefore free of charge. We gratefully acknowledge the support of the Faculty of Mathematics, Natural Sciences and Information Technologies (University of Primorska).

Matjaž Krnc

Contents

Online Long Short-Term Memory Network for Stock Trading

Dušan Fister
Univerza v Mariboru, Ekonomsko-poslovna fakulteta,
Razlagova 14,
SI-2000 Maribor, Slovenia
dusan.fister1@um.si

Timotej Jagrič
Univerza v Mariboru, Ekonomsko-poslovna fakulteta,
Razlagova 14,
SI-2000 Maribor, Slovenia
timotej.jagric@um.si

ABSTRACT
Economic theory teaches that it is impossible to earn higher-than-normal returns on trading stocks, and that the only chance to earn higher profits is to take higher risk. Often, the practice reveals that higher-than-normal stock returns may indeed be earned, which is a hypothesis we would like to empirically test. In this way, we design so-called mechanical trading system, which focuses on the technical analysis of past stock data, and perform daily trading for the German stock from period 2010-2019. Long short-term memory network is taken as a basis of the mechanical trading system. Obtained results show that higher-than-normal returns are easily obtainable, which polemicizes the efficiency of the observed stock.

Keywords
LSTM networks, mechanical trading system, stock trading

1. INTRODUCTION
Efficient market hypothesis (EMH) states that asset prices fully reflect all information available [3] and that asset prices quickly incorporate any new information without delay. Consequently, higher-than-normal returns cannot be achieved and predictions of future prices cannot be valuable [10]. Stocks strictly follow a random walk (are unpredictable) and it is impossible to beat the market. Malkiel [10] reports opposite opinion, where he states that market inefficiencies and arbitrage opportunities to earn higher-than-normal returns indeed exist. A sample of these includes market bubbles, market crashes and other irrational pricing. Pedersen [12] even outlines common investment strategies to detect arbitrage opportunities. To investigate any market inefficiencies for a case study of Germany, we employ a single-stock automated mechanical trading system (MTS). As a benchmark, we choose the stock Daimler AG. We implement the MTS trading strategy using the long short-term memory network (LSTM) and compare its performance to the passive trading strategy. LSTMs are found usable in broad areas, such as sentence classification [2], trajectory prediction of autonomous vehicles [9], flood forecasting [8] and malware detection [15]. Furthermore, LSTM are used in engineering for estimating remaining useful life of systems [17] and in medicine for automated diagnosis of arrhythmia [11]. The structure of the paper is as follows: chapter two outlines the fundamentals of LSTM networks. Chapter three lists the information about the dataset and explains methodology. Chapter four shows the experiments and results, while chapter five concludes the paper.

2. LONG SHORT-TERM MEMORY
Long short-term memory networks (LSTMs) are a kind of artificial neural networks, specifically designed to deal with sequential and dynamic data [6]. LSTMs are recurrent neural networks (RNN) with an agile structure that attempt to remember long-term dependencies and prevent both the usual RNN problems: exploding and vanishing gradients.The benefit of the LSTM lies in a memory cell, i.e. central element of the network, and the three kinds of gates which control the flow of information. Memory cell accumulates the internal state by supplying the sequences of data. Each gates constitutes of weights, which are adapted during the learning process [5]. Figure 1 shows the structure of the usual LSTM. The common LSTM structure consists of the

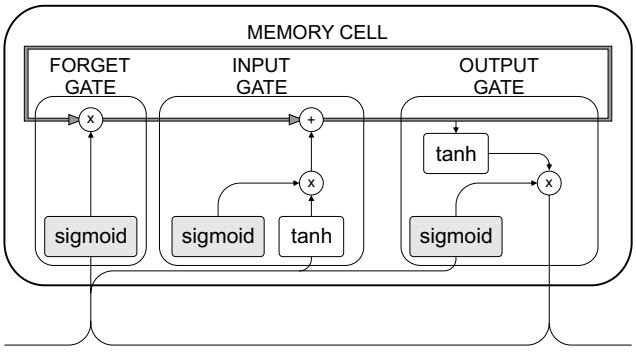


Figure 1: LSTM structure.

forget, input and output gate. Forget gate, which is a simple *sigmoid* function, is used to filtrate past information to be brought into following time step. Input gate, which is a combination of *sigmoid* and *tanh* functions, is used to enter new information into the memory cell. Output gate,

DOI: https://doi.org/10.26493/978-961-7055-82-5.5-8

which is also a combination of *sigmoid* and *tanh* function, is used to output relevant information from the memory cell and transfer the information into following time step. LSTM layer can be single, or stacked into multi-layered networks. Figure 1 shows the individual LSTM layer, which is sufficient for many cases. For more comprehensive (e.g. non-stationary) data, [1] suggests the use of stacked LSTM network, e.g. two individual LSTM layers composed into a single network.

Table 1: Dataset variables, used during trading.

Explanatory variables	Vars.
1. Stock and market data:	
open, close, high, low, adj. close, volume	12
2. Date data:	
month, day, day of week, days to next trading day, days from previous trading day	5
3. Technical indicators:	
RET: $n = \{1, 2, 3, ..., 10\}$-day period	10
DIFF: $n = \{1, 2\}$-day period	2
DIFF_RET: $n = \{1, 2\}$-day period	2
RSI: 14-day period RSI	1
MACD: 12-day short, 26-day long and 9-day signal period	1
INCL: $n = \{5, 10, 15, 20\}$-day period	4
INCL_CHG:	4
REL_DIFF:	2
Sum	43

3. MECHANICAL TRADING SYSTEM

LSTMs can be applied to diverse problems and domains of sequential data. In this paper, we apply the LSTM for classification problem, which drives the MTS. MTS follows the underlying daily stock prices and outputs the three common trading decisions respectively: buy, hold and sell. According to the trading decisions, stocks are bought, sold or held in the portfolio. Since the movement of stock prices affects the portfolio value, it is desired that the current and final value of portfolio (and thus profit), is maximized. For implementation of MTS, we assume (1) perfect liquidity of stocks and (2) close price trading. Perfect liquidity means that stocks can be sold or bought instantly. Close price trading means that stocks are traded at the close price, just after the market close. The MTS trades with stocks daily. Here, only the relevant historic daily trading data is supplied. No additional information, such as company's financial reports and other statements, are taken into account, since these are available for longer time periods only, e.g. a quarter of a year. The LSTM is thus able to extract daily trading patterns and identify trading opportunities that arise by behavioural acting of ordinary investors. These are often psychologically-affected and subjected to phenomena, such as panicking and following the herd instinct. Those two largely drive the price of a stock and forecasting them may be beneficial to earn higher-than-normal returns.

3.1 Dataset

Daimler AG is a German automotive company, with approximately 50 B Eur of market capitalization, 1.07 B of shares outstanding and almost 4.6 M of trades daily, at the time of writing. It is one of the blue-chip stocks, listed in the index of 30 most prominent German companies DAX30. The stock data is obtained from Yahoo Finance website[1], during the period from 4 Jan 2010 - 7 Jun 2019 on a daily basis. Following data is downloaded: the highest daily price ($High$), the lowest daily price (Low), open price ($Open$), close price ($Close$), daily volume ($Volume$) and adjusted close price ($AdjClose$). The data is downloaded for the Daimler AG stock and DAX30 industrial index. Composing those two into a dataset, several other variables are derived from. In this way, a more expanded, detailed and concrete dataset is built. Table 1 lists the dataset variables. There are 12 stock and market data variables, 5 variables which symbolize the date and 8 technical indicators. 10 stock returns (RET) are extracted, ranging from 1-day returns to 10-day returns. 2 stock differences (DIFF) are obtained as differences between today and yesterday's price and today's price and price 2 days ago. Difference of returns (DIFF_RET) is calculated by subtracting the stock returns. Relative strength indicator (RSI) and moving average convergence divergence (MACD) are common trading indicators. Inclination (INCL) is obtained by calculating the regression coefficient over n periods of stock prices. Inclination change (INCL_CHG) is calculated from the 5-day inclination and the 5-day inclination five days ago and current 5-day inclination to current 10-day, 15-day and 20-day inclinations. Relative differences (REL_DIFF) are obtained as ratio between close and open prices, and high and low prices. Response variable ($\mathbf{y}$) is generated by deriving the 1-day stock returns. If the underlying stock return overcomes the fixed threshold, set at 0.01 (1%) in our case, the response variable is marked as Buy. Furthermore, if the stock return exceeds the negative value of threshold, it is marked as $Sell$. If it does not exceed any of the two, it is marked as $Hold$. These signals are encoded in binary form to fulfill the requirements of the dense layer.

3.2 Methodology

The work with LSTM is divided into two samples, i.e. training and prediction, since LSTM is a modeling tool and thus needs some data to be trained from. Initially, the training sample of the LSTM is used for offline training and is intended to capture general long-term stock dependencies. It is performed only once. Additionally, an online learning and regular re-train of the LSTM is applicable to update the pre-built LSTM model with the latest information, i.e. latest stock price movements. The procedure of training and prediction samples is explained in Algorithm 1. First the stock data (dataset) is normalized in range(0,1) and batch size b is defined. Dataset is split into two samples. In-sample is used for the offline training and after, the memory cell is reset. The LSTM is used to predict the in-sample and establish a stable memory cell. Predictions on the in-sample are biased and are not used for real trading. After the internal state is established, the out-of-sample is used for prediction. Since stacked LSTMs require the batch size to be divisible from the number of samples, the first b-samples from out-of-sample are only used for prediction each time. Its solutions are appended into a vector, to be evaluated later. The iterative online training comes next, where each time, the b-samples are transferred from out-of-sample to in-sample. By expanding the in-sample for b-samples, out-

[1] `https://finance.yahoo.com`

Algorithm 1 Algorithm of online LSTM training.

```
1: procedure ONLINE LSTM TRAINING
2:     NORMALIZE(dataset);
3:     DEFINE b                                  ▷ define batch size
4:     in-sample,out-of-sample=SPLIT(dataset);
5:     for no. of epochs do                      ▷ number of epochs
6:         model ← TRAIN(in-sample);             ▷ offline training
7:     end for
8:     RESET_STATE(model)                        ▷ reset memory cell state
9:     PREDICT(in-sample)                        ▷ establish memory cell state
10:    decision ← PREDICT(out-of-sample[0:b])
11:    while out-of-sample do
12:        in-sample=APPEND(out-of-sample[0:b])
13:        out-of-sample=REMOVE(out-of-sample[0:b])
14:        RESET_STATE(model)
15:        for no. of epochs / 100 do
16:            model ← RETRAIN(in-sample);       ▷ online training
17:        end for
18:        RESET_STATE(model)
19:        PREDICT(in-sample)
20:        decision ← PREDICT(out-of-sample[0:b])   ▷ append first b predictions
21:    end while
22:    results ← BACKTEST(decision)
23:    INTERPRET_RESULTS
24: end procedure
```

of-sample is simultaneously reduced for the same amount. The LSTM pre-built model is then retrained using the expanded in-sample, which practically means that b trading days need to (physically) pass prior any LSTM online retrain occurs. For retraining, a lower number of the number of epochs is taken. The memory cell is next reset and the expanded in-sample predicted to establish internal state of the memory cell. Next, the first b-samples of the remaining out-of-sample are predicted and its solutions appended in the **decision** vector.

The quality of the MTS trading strategy, i.e. **decision** vector, is evaluated using the back-testing approach. By back-testing, the MTS is put into history and is given an initial amount of cash, which is used to buy stocks during the trading. MTS follows underlying movement of stock prices day-by-day and trades with stocks in a continual trading process. If the LSTM supposes that tomorrow's close price of a stock will increase over the preset threshold, it gives the signal to buy. Alternatively, if the price is expected to drop below the threshold, the MTS gives the signal to sell. Once the signal to buy (*buy*) is classified, maximum number of stocks, reduced for transaction costs, are bought for the current amount of money. Similarly, when the signal to sell (*sell*) is classified, all the stocks are sold and the amount is lowered for transaction costs. No information about any forthcoming stock prices is specified at any time.

4. EXPERIMENTS AND RESULTS

The goal of the experimental work is to show that automated MTS using the LSTM trading strategy can be used to catch excess profits (higher-than-normal returns) on trading stocks. Daimler AG stock is taken as a benchmark. Its dataset is split 70%-30%, where 70% goes to in-sample and the rest 30% to out-of-sample. Framework is implemented in the Python programming language, using the Keras Deep Learning Library[2]. A two layered stacked LSTM and Adam optimizer are used [7]. Figure 2 outlines the structure of the used network, while the table 2 the algorithm setup. Trading costs of 1% are adopted, according to literature [13, 14, 16], which come into play when buying or selling stocks.

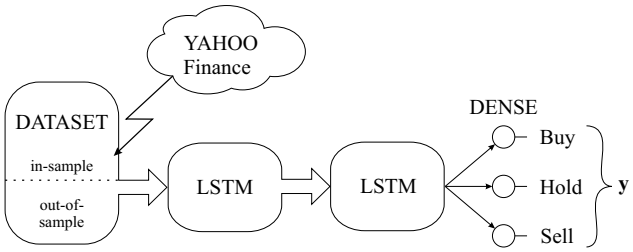


Figure 2: Structure of the network.

Experiments are conducted using the out-of-sample back-testing. The higher the final portfolio value, the better the trading strategy. Results are reported graphically. Figure 3 presents the flow of trading, where the x-axis shows the trading days and the y-axis shows the portfolio value. Dashed line equates to passive trading strategy, while the solid line to MTS using the LSTM trading strategy. The greater the difference of solid line, compared to dashed line, the higher the excess profit. Passive trading strategy implements a single transaction and single transaction costs. From the day 450, passive strategy starts losing portfolio value below the initial amount of cash. In the end it scores 25.28% of loss. Although the LSTM on the other hand implements 6 transactions and thus increases transaction costs significantly, it behaves much more beneficially. When the MTS detects that the stock price is about to rise, it buys stocks, and when detects the price to fall, it sells them. The LSTM scores the 16.02% of profit and thus overcomes the passive trading strategy for more than 55%.

Table 2: Algorithm setup.

Parameter	Value
Batch size b	15
No. of epochs	1000
Learning rate	0.001
Optimizer	Adam
No. of units $LSTM_1$	10
No. of units $LSTM_2$	10

5. CONCLUSIONS

Efficient market hypothesis states that technical analysis is worthless for predicting the stock market performance. To test the validity of this hypothesis, we have applied the stacked long short-term memory network for trading the German stock Daimler AG from year 2010 to 2019. Back-testing approach was used to evaluate the quality of trading decisions. Implemented trading strategy significantly outperformed the passive trading strategy and found many arbitrage opportunities and other inefficiencies. Obtained results coincide with the results from a more detailed study in [4]. Results show that it is possible to generate higher-than-normal returns by relying on the technical analysis

[2] `https://keras.io`

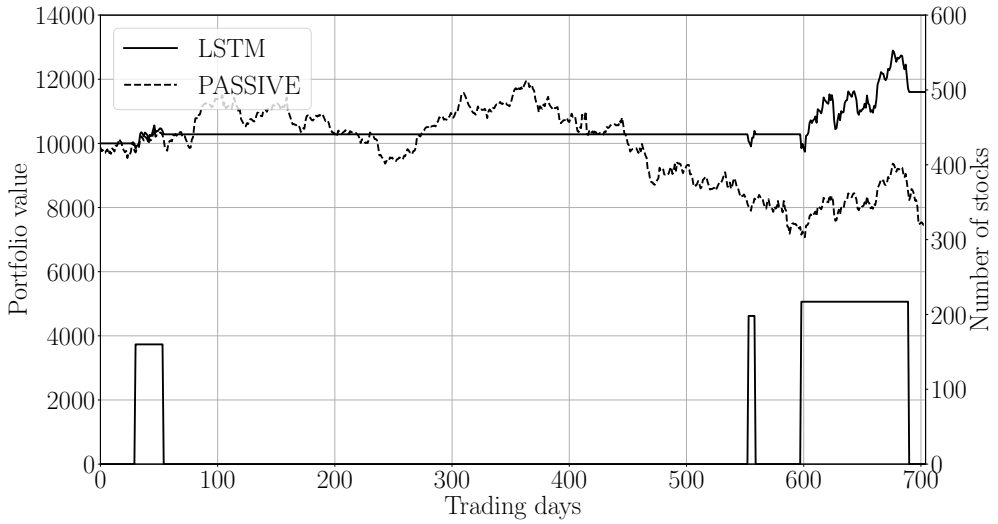


Figure 3: Comparison between the automated MTS with LSTM trading strategy and passive trading strategy.

only. We conclude that either stock is inefficient, or the technical analysis is indeed a very efficient tool. We agree that arbitrage opportunities are partly conditional to the level of volatility of stock prices, since the higher the volatility, the higher the arbitrage opportunities. Volatility to its nature comes with constant reception of new information – the EMH states that stock prices quickly adjust to new information. If new information is constantly received, volatility is necessarily implied as well. For future work, we would like to implement a multi-stock MTS with LSTMs. Instead of single-stock returns, we would like to examine the portfolio returns. For example, German stocks in stock index DAX30 would be suitable for such an analysis.

6. REFERENCES

[1] J. Brownlee. *Long Short-term Memory Networks with Python: Develop Sequence Prediction Models with Deep Learning*. Jason Brownlee, 2017.

[2] Z. Ding, R. Xia, J. Yu, X. Li, and J. Yang. Densely connected bidirectional lstm with applications to sentence classification. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pages 278–287. Springer, 2018.

[3] E. F. Fama. Efficient capital markets: A review of theory and empirical work. *The journal of Finance*, 25(2):383–417, 1970.

[4] D. Fister, J. Mun, V. Jagrič, and T. Jagrič. Deep learning for stock market trading: A superior trading strategy? *Neural Network World*, 29(3):151–171, 2019.

[5] F. A. Gers, J. Schmidhuber, and F. Cummins. Learning to forget: Continual prediction with lstm. 1999.

[6] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[7] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[8] X.-H. Le, H. V. Ho, G. Lee, and S. Jung. Application of long short-term memory (lstm) neural network for flood forecasting. *Water*, 11(7):1387, 2019.

[9] L. Lin, S. Gong, T. Li, and S. Peeta. Deep learning-based human-driven vehicle trajectory prediction and its application for platoon control of connected and autonomous vehicles. In *The Autonomous Vehicles Symposium*, volume 2018, 2018.

[10] B. G. Malkiel. The efficient market hypothesis and its critics. *Journal of economic perspectives*, 17(1):59–82, 2003.

[11] S. L. Oh, E. Y. Ng, R. San Tan, and U. R. Acharya. Automated diagnosis of arrhythmia using combination of cnn and lstm techniques with variable length heart beats. *Computers in biology and medicine*, 102:278–287, 2018.

[12] L. H. Pedersen. *Efficiently inefficient: how smart money invests and market prices are determined*. Princeton University Press, 2015.

[13] H. Shin and S. Y. Sohn. Segmentation of stock trading customers according to potential value. *Expert systems with applications*, 27(1):27–33, 2004.

[14] V. Šonje, D. Alajbeg, and Z. Bubaš. Efficient market hypothesis: is the croatian stock market as (in) efficient as the us market. *Financial theory and practice*, 35(3):301–326, 2011.

[15] R. Vinayakumar, K. Soman, P. Poornachandran, and S. Sachin Kumar. Detecting android malware using long short-term memory (lstm). *Journal of Intelligent & Fuzzy Systems*, 34(3):1277–1288, 2018.

[16] B. W. Weber. Screen-based trading in futures markets: recent developments and research propositions. In *Proceedings of the 32nd Annual Hawaii International Conference on Systems Sciences. 1999. HICSS-32. Abstracts and CD-ROM of Full Papers*, pages 10–pp. IEEE, 1999.

[17] Y. Wu, M. Yuan, S. Dong, L. Lin, and Y. Liu. Remaining useful life estimation of engineered systems using vanilla lstm neural networks. *Neurocomputing*, 275:167–179, 2018.

Defining computational thinking framework for introductory programming in higher education

Boštjan Bubnič
University of Maribor
Faculty of Electrical Engineering and Computer Science
bostjan.bubnic@student.um.si

ABSTRACT

Computational thinking (CT) is gaining recognition as an important skill set for the general problem solving. Although CT has its origin in computer science and programming, there is a great interest of researchers and educators to explore how to include CT in all scientific and engineering disciplines, as well as in kindergarten to 12^{th} grade education (K-12) curriculum. To determine the effective methods for teaching, learning and assessing CT, a definition and its scope is needed. To date there is no consensus in terms of formal CT definition as well as the definitive or necessary components of CT. However, our study builds upon the consensus that multiple skills are involved in CT. The result of this study is the CT framework proposal to be used in introductory programming courses in higher education. The framework is an intersection of previous research that has identified basic, domain independent components of CT and domain specific programming practices. We hope that the framework will encourage the future research on teaching and assessing CT in the higher education.

Keywords

Computetional thinking, Framework, Introductory programming, Higher education

1. INTRODUCTION

Computational thinking (CT) has a long history within computer science (CS). It was known under the terms "algorithmizing" and "algorithmic thinking" in the 1950s and 1960s, when it was considered as a mental practice for problem conceptualization, for inventing formalisms and concepts to solve problems [22]. In the educational context, the CT phrase was introduced by Seymour Papert in the 1980s, when he was teaching Logo, an educational programming language, to improve students's ability to think procedurally [12]. Recently, the term was reintroduced and popularized by Wing who described CT as a way of solving problems, designing systems, and understanding human behavior by drawing on the concepts fundamental to CS [25]. She also argued that computational thinking is a fundamental skill for everyone, not just for the computer scientists. The shift from the CS to the general problem solving domain increased attention among researchers, educators and practitioners in exploring how to include CT across the educational spectrum and in everyday life. However, the broad spectrum of perspectives on CT also presents challenges to teaching, learning and assessing CT. The diverse spectrum of concepts, skills and practices under the umbrella of CT resulted in the lack of clarity as to what computational thinking should be. In this regard, to date there is no consensus concerning the definition and the scope of CT, nor there is a consensus about the definitive or necessary components that constitute CT. Nevertheless, it is generally agreed that multiple skills are involved in CT [18]. Moreover, researchers have begun to characterize computational thinking by means of CT taxonomies and frameworks, where particular concepts and practices were observed.

This work expanded upon our previous work where general, domain independent components of CT were identified [4]. Moreover, the aforementioned study motivated this work. The main contribution of this paper is the definition of computational thinking framework that was sourced from the general, domain independent CT components. It is intended to be used in the higher education institutions. We envision this work to serve as a foundation for designing CT instruments and course materials to assess and foster computational thinking for CS majors and non-CS majors. The proposed framework is also aligned with the particular CS curriculum categories [17].

2. RELATED WORK

When computational thinking was reintroduced by Wing [25], abstraction and decomposition were the fundamental components. The initial set of components were later refined with automation. After initial component conceptualization, researchers have begun to characterize computational thinking by means of CT taxonomies and frameworks.

This section presents a brief description of related work with the focus on the frameworks and taxonomies that are applicable to higher education with the emphasis on CS domain. Gouws et al. [7] defined a framework of six distinct skills and practices that served as the foundation for the assessment design, where the intervention was applied to introductory computer science course (CS1) students. Billion-

DOI: https://doi.org/10.26493/978-961-7055-82-5.9-12

niere [1] assessed CS1 students' comprehension within abstraction, arrays of objects, and inheritance. The results of the assessment were categorized into five computational thinking concepts. Romero et al. [16] proposed a framework for evaluating CT in creative programming activities. The intervention was applied to undergraduate students of a bachelor's degree in elementary school education. Peteranetz et al. [14] report on fostering computational thinking through computational creativity based on an online course that was tailored for non-CS majors. Rambally [15] proposed a framework of CT skills and practices that were integrated into discrete structures course, which was a required course for all Information Technology majors. Finally, the two most cited papers in context of CT frameworks appear to be Brennan and Resnick [3] and Weintrop et al. [24]. Although none of the papers was aimed for the CT within higher education, there are studies that build on these particular papers in the context of higher education.

3. PROPOSED FRAMEWORK

Algorithm, abstraction and decomposition, previously identified as the uppermost general, domain independent CT components, served as a starting point for this study. Afterwards, each of the CT concepts were further examined within the computer programming domain. This was a two-step process. Fundamental programming concepts and practices were grouped into four categories within the first step. The initial corpus of computer programming concepts was sourced from the results of research conducted by Luxton-Reilly et al. [10]. Within the literature review they identified more than sixty different computer programming concepts that were grouped into twelve master categories. To extract only concepts relevant to algorithm, abstraction and decomposition, each particular concept from [10] was further analysed with the relevant sections of the Computer Science Curricula 2013 [17] and chapters concerning the software engineering [2]. The aims of the second step were the intersection points between particular fundamental computer programming concepts and the CT concepts. Computer Science Curricula 2013 [17] and the dissertation reporting on the process of developing validated CS assessment [23] were the primary sources for the aligning process.

The proposed framework of computational thinking for introductory programming in higher education (CTF) is presented in the Table 1. The CTF is envisioned to serve as the foundation for designing CT instruments and course materials to assess and foster computational thinking for CS majors and non-CS majors. The visual representation of the CTF is represented as a two dimensional grid. The axis of this framework describes the main components that make up CT, as well as programming practices that are part of computer programming. Each grid element incorporates computer programming concepts that are aligned with particular CS curriculum category [17]. We envision each of these programming concepts to be implemented as programming artefacts within the specific programming methodology or as a pseudocode.

Algorithm, abstraction and decomposition are the relevant, domain independent CT components that were identified in our previous study [4]. It should be noted that thirty-six different CT concepts, skills and practices were identified within our previous study, while only algorithm, abstraction and decomposition appeared to be relevant, domain independent CT components. The relevance was evaluated according to the frequency identified in a literature review. Within our CTF, they are represented on the vertical axis.
Algorithms can be generally defined as procedural building blocks of a computer programming, of a human thought and of a general problem solving [4]. In the theoretical computer science, an algorithm is defined as a procedure that satisfies the criteria of finiteness, input, output, effectiveness, and definiteness [8]. However, in our framework an algorithm is associated with constructing an algorithmic solution to a problem to be solved. In educational context, this practice is referred as algorithmic thinking.
Abstraction can generally be characterized as the conceptual process of eliminating specificity by ignoring certain features. Abstraction is closely related to the modeling concept and to the concept of generalization. The model is abstraction of a real or a conceptual complex system. Abstraction levels and the hierarchy of abstraction are important aspects in the models design practice. In software engineering, abstraction involves the extraction of properties of an object according to some focus: only those properties are selected which are relevant with respect to the focus [5]. In computer programming abstraction practices can be observed within two categories: by mechanisms or by programming constructs. Abstraction by specification and abstraction by parametrization are the two mechanisms that make abstractions explicit and tangible. Furthermore, these two powerful methods for constructing computer programs contribute to the definition of the following important programming constructs: procedural abstraction, data abstraction, iteration abstraction and type hierarchy.
Decomposition deals with breaking down a problem into smaller, more manageable components where each component can be managed independently. Levels of abstraction need to be utilized to successfully decompose a problem into smaller components. In computer science, distinct variants of decomposition can be observed. Parnas [13] investigated hierarchical decomposition in the context of modularity in order to decompose complex information system into a number of smaller, manageable modules. Moreover, decomposition is the crucial part of structured, object-oriented and functional programming paradigms. Generally, the aim is to decompose a computer program into modules, which are set of smaller programs to solve sub problems. Smaller programs interact with one another in a simple, well defined way.

As a discipline, computer programming incorporates several processes, skills and practices. However, our aim at this point was to identify computer programming skills and practices that correlate with particular CT concepts. The primary source for identifying programming skills and practices was the Computer Science Curricula's sections Fundamental Programming Concepts, Development Methods, Algorithms and Design [17]. Furthermore, to observe the broadest spectrum, software engineering skills and practices were also included within our study [2], [9]. Finally, problem conceptualization, implementation, debugging and evaluation were included in the framework. They are represented on horizontal axis in the Table 1. We define problem conceptualization as an algorithmic solution to the observed problem.

Table 1: Computational thinking framework for introductory programming in higher education

Components of CT	**Problem conceptualization**	**Implementation**	**Debugging**	**Evaluation**
Algorithm	• Algorithmic solution to a problem	• Diagram or flowchart • Algorithmic notions of flow control	• Debugging of algorithmic solution • Error detection strategies • Code comprehension	• Correctness • Efficiency • Simplicity
Abstraction	• Abstraction levels • Abstraction hierarchy	• Mechanisms of abstraction • Programming constructs	• Error detection strategies • Code comprehension	• Relevance • Efficiency • Simplicity
Decomposition	• Components identification • Components modeling	• Modularization	• Error detection strategies • Code comprehension	• Correctness • Simplicity

It is a part of requirements engineering process that assures the problems are properly defined, understood and framed in a way that allows algorithmic solution. Domain specific knowledge is required within the requirement engineering process [2]. Implementation is a practice of programming the algorithmic solution within specific programming language. The programming language constructs principally relay on the programming methodology that is applied, such as structured programming, modular programming, abstract data type programming or object-oriented programming. Moreover, the algorithmic solution can be implemented as a pseudocode. Debugging is the process of understanding, finding and correcting errors. While various error detection strategies exist, systematic debugging strategy has proven to be the most effective [26]. To effectively locate and fix an error within a computer program, code comprehension skills are required. Donaldson and Cutts [6] proposed several activities to develop the code comprehensions skills. The evaluation process normally involves some identification of relevant standards of worth, merit or value. Moreover, the process is also concerned with some investigation of the performance of the evaluands on these standards [19]. Smith and Cordova [20] propose several traits to be used for computer program evaluation, such as correctness, efficiency and completeness.

4. DISCUSSION

The motivation for this work arose whilst working on previous work, where we were identifying general, domain independent components of CT [4]. During the literature review, the paper from Tang et al. [21] reporting on content analysis of computational thinking research has been examined. Results within the paper has revealed that higher education level was the second most frequently investigated category within CT. Furthermore, computer science was the most frequent subject of research in context of CT [21]. On contrary, within our previous study we have found only a few studies investigating on the CT frameworks or taxonomies in higher education [4]. In this context, our study tried to fill the gap between CT concepts and computer programming concepts and practices by proposing the CTF.

The relevancy of the proposed CTF in context of CT is fundamentally different from previous studies, because only relevant, domain independent components of CT were used as a starting point of our study. It should be noted that thirty-six different CT concepts, skills and practices were identified within our previous study, while only algorithm, abstraction and decomposition appeared to be the prospects for achieving the relevancy consensus [4]. The primary aim of this study was the alignment process between these CT concepts and computer programming concepts. The mapping of the programming concepts to programming code artifacts is yet to be the subject of further research. On the contrary, previous studies that had proposed CT frameworks and taxonomies [3], [24] had mainly focused on computer programming artifacts and mapped them directly to various CT skills and practices.

Nevertheless, the major result of this work are the CTF grid elements that incorporate computer programming concepts, aligned with CS curriculum and their associated CT concepts. In this regard, the CTF is envisioned to serve as the foundation for teaching and evaluating core CT skills and practices within computer programming.

5. CONCLUSIONS

The result of this work is a framework proposal for CT in introductory programming in higher education. The framework provides a useful starting point for further research. The future work should be oriented toward implementing computer programming artefact based on proposed CTF. The envisioned outcome of further research might be the CT instrument suited for implicitly assessing CT based on computer programming tasks within the higher education level.

Furthermore, the proposed CTF should stimulate further research in the context of success or failure of novices in introductory programming in higher education, often referred as "CS1 dropout rate". The researchers observed that the dropout rate problem can be divided to the following two categories: language problem and design problem [11]. If these categories could be mapped into the computational thinking context, then the proposed CTF could serve as a foundation for CT assessment, a potential predictor for the CS1 dropout rate.

6. REFERENCES

[1] E. Billionniere. *Assessing Cognitive Learning of Analytical Problem Solving*. Doctoral dissertation, Arizona State University, December 2011.

[2] D. Bjørner. *Software Engineering 3 - Domains, Requirements, and Software Design*. Springer-Verlag, Heidelberg, 2006.

[3] K. Brennan and M. Resnick. New frameworks for studying and assessing the development of computational thinking. pages 1–25, Vancouver, BC, Canada, 2012.

[4] B. Bubnic and T. Kosar. Towards a consensus about computational thinking skills: Identifying agreed relevant dimensions. Newcastle, UK, 2019. Proceedings of the 30th Annual Workshop of the Psychology of Programming Interest Group - PPIG 2019 - submited for publication.

[5] K. Czarnecki. *Generative Programming: Methods, Tools, and Applications*. Addison-Wesley Professional; 1 edition, June 2000.

[6] P. Donaldson and Q. Cutts. Flexible low-cost activities to develop novice code comprehension skills in schools. pages 1–4, Potsdam, Germany, 2018. ACM New York.

[7] L. Gouws, K. Bradshaw, and P. Wentworth. First year student performance in a test for computational thinking. pages 271–277, East London, South Africa, 2013. ACM New York.

[8] D. Knuth. *The Art of Computer Programming: Volume 1: Fundamental Algorithms, Third Edition*. Addison-Wesley, USA, 1997.

[9] A. J. Ko, R. Abraham, L. Beckwith, A. Blackwell, and M. Burnett. The state of the art in end-user software engineering. volume 43, USA, April 2011. ACM New York.

[10] A. Luxton-Reilly, B. A. Becker, Y. Cao, and R. McDermott. Developing assessments to determine mastery of programming fundamentals. pages 47–69, July 2017.

[11] M. McCracken, V. Almstrum, D. Diaz, and M. Guzdial. A multi-national, multi-institutional study of assessment of programming skills of first-year cs students. volume 33, pages 125–180, USA, 2001. ACM New York.

[12] S. Papert. *Mindstorms: children, computers, and powerful ideas*. Basic Books, Inc, USA, 1980.

[13] D. Parnas. On the criteria to be used in decomposing systems into modules. volume 15, pages 1053–1058, USA, 1972. ACM.

[14] M. S. Peteranetz, L.-K. Soh, and E. Ingraham. Building computational creativity in an online course for non-majors. pages 442–448, Minneapolis, USA, 2019. ACM New York.

[15] G. Rambally. Integrating computational thinking in discrete structures. pages 99–119, Switzerland, 2017. Springer International Publishing AG.

[16] M. Romero, A. Lepage, and B. Lille. Computational thinking development through creative programming in higher education. volume 14, pages 1–15, 2017.

[17] M. Sahami, A. Danyluk, S. Fincher, and K. Fisher. *Computer Science Curricula 2013*. The Joint Task Force on Computing Curricula Association for Computing Machinery (ACM) IEEE Computer Society, USA, December 2013.

[18] C. Selby and J. Woollard. Refining an understanding of computational thinking. University of Southampton Institutional Repository, 2014.

[19] I. Shaw, J. Greene, and M. Mark. *The SAGE Handbook of Evaluation*. SAGE Publications Ltd, USA, 2013.

[20] L. Smith and J. Cordova. Weighted primary trait analysis for computer program evaluation. pages 14–19. Consortium for Computing Sciences in Colleges, 2005.

[21] K. Y. Tang, T. L. Chou, and C. C. Tsai. A content analysis of computational thinking research: An international publication trends and research typology. pages 1–11, USA, 2019. Springer Nature.

[22] M. Tedre. The long quest for computational thinking. pages 120–129. Koli, Finland, Koli Calling '16 Proceedings of the 16th Koli Calling International Conference on Computing Education Research, November 2016.

[23] A. E. Tew. *Assessing Fundamental Introductory Computing Concept Knowledge in a Language Independent Manner*. Doctoral dissertation, Georgia Institute of Technology, December 2010.

[24] D. Weintrop, E. Beheshti, and M. Horn. Defining computational thinking for mathematics and science classrooms. volume 25, pages 127–147, February 2016.

[25] J. Wing. Computational thinking. volume 49, pages 33–35, USA, March 2006. ACM New York.

[26] A. Zeller. *Why Programs Fail: A Guide to Systematic Debugging 2nd Edition*. Morgan Kaufmann, USA, June 2009.

Passive Floating Probe

Authors: *

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

Urban Knupleš
University of Maribor
Faculty of Electrical Engineering and Computer Science
Koroška cesta 46, Maribor
urban.knuples@student.um.si

Michele Perrone
University of Maribor
Faculty of Electrical Engineering and Computer Science
Koroška cesta 46, Maribor
michele.perrone@student.um.si

Tadej Šinko
University of Maribor
Faculty of Electrical Engineering and Computer Science
Koroška cesta 46, Maribor
tadej.sinko@student.um.si

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

ABSTRACT

This paper illustrates a student project design of an autonomous passive floating probe, which gathers data from its on-board sensors. The data are subsequently transmitted via the Iridium satellite network to a dedicated server which then displays the information on the web. Once assembled, the probe is planned to be deployed into the Atlantic Ocean. The main features of the probe are a solar panel with a rechargeable battery, a GPS module, an Iridium satellite modem, a vast set of sensors and an ARM-based microcontroller running a custom firmware based on FreeRTOS.

Keywords

Remote sensing, data gathering, real-time systems

1. INTRODUCTION

Remote sensing and data retrieval is a challenging task because physical access to the equipment is limited or impossible; more so, if executed in a harsh environment such as the oceans. Operating in such an enviroment brings additional challenges concerning protection from water, salt corrosion, plaque, or waste build-up. Furthermore, unpredictable weather can affect the energy autonomy of the device or the accuracy of sensor data. To overcome these problems, we propose a design for a passive floating probe, which is accessible due to its consumer-available components.

The proposed design follows a modular architecture, which allows faster prototyping and more advanced future iterations based on this outline.

The main difference between a passive and an active floating probe is in the way it interacts with its surrounding environment. A passive probe only listens and gather information through its sensors and move along with the ocean currents, while an active probe is equipped with a propulsion system that would allow it to change its course by itself based on sensor data and parameters or remotely by an operator.

The goal of this project is the acquisition of real-world measurements for analysis and distribution. A dedicated website [1] is planned to show project progress, visualize received data once deployed and enable distribution of collected data to interested parties.

2. RELATED WORK

Before discussing the proposed design, this paper outlines a handful of projects which had been the primary source of inspiration for the authors. The main inspiration comes from the Maker Buoy project [2]. Its features are a modular building design powered by solar energy, communication via the Iridium satellite network and the use of a floating exterior design, which is the basis of this paper's proposed design (discussed in Sec. 3). A project called OMNI (Ocean Monitoring Network Initiative) uses a similar hardware approach to the Maker Buoy [3].

The ZL1SIX Ocean Floater is a floating buoy powered by a battery pack which uses an Amateur radio transmitter to

*Listed in alphabetical order

DOI: https://doi.org/10.26493/978-961-7055-82-5.13-17

transmit data on a 10 MHz Amateur Radio band [4].

The international project Argo is an initiative that uses thousands of battery-powered floats around the world to measure the temperature, salinity, and velocity of the ocean currents. The floats collect data by staying submerged underwater and transmit data by coming up to the surface and sending it via a satellite antenna. As of this writing, the project has 3869 floats active across all the oceans [5].

The main difference between the outlined design and other similar projects is the presence of a Geiger counter, a gas sensor, a microphone [1], a camera [2], and the use of a more capable additional microcomputer system in the form of Raspberry Pi for processing extensive data and ease of development of such processing software.

3. PROPOSED DESIGN

This section outlines the design of the probe, the software, and the communication between them. Because the proposed design is not necessarily final, minor changes can happen to individual modules while components are sourced.

3.1 Hardware design

For its operation, the probe must accomplish different tasks. Modules that contribute to the same task are classified into a subsystem. All proposed subsystems are shown in Fig. 1. Modules inside these subsystems will be presented later.

The core of the probe will be a SensiBLE SIMBA-PRO development board. Besides various sensors, which are included in the inner sensors subsystem, it will feature an STM32L476xx microcontroller with an ARM Cortex M-4 m processor. The microcontroller will retrieve data from the sensors and send them to the data storage subsystem. The most critical data will be sent to our server daily using the communication subsystem. When the system has access to enough power, advanced operations, such as image capture, sound recording, and advanced data processing will be executed on an external Raspberry Pi module (which is part of the image capture and processing subsystem).

The data storage subsystem shown in Fig. 2 consists of a micro SD card that will store the raw collected data and an adapter for the storage device. Since satellite communication is expensive, not all data will be sent to the server (e.g., only average or extreme measurements during the day). Nevertheless, every measurement will be saved to the data storage device. This way, if the probe is ever recovered, much more data could be analyzed. Even with low chances for recovery event, it was decided to include this system due to its low cost.

The inner sensors subsystem shown in Fig. 3 consists of the sensors which operate from inside the probe. Some of them gather engineering information about the environment inside the casing and can, therefore, reveal some problems with the probe (e.g., increased temperature of the processors, change in inner pressure or humidity which could indicate that the watertight case is damaged). Other sensors can detect and measure outside sources (GPS signal, Beta and Gamma radioactive rays, sound, vibration, and light – since the case is transparent).

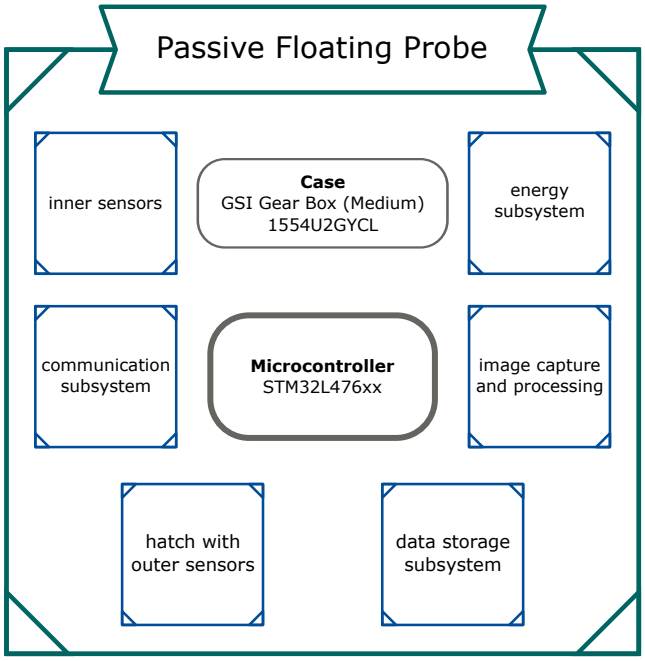


Figure 1: Subsystems are shown in rectangles. The microcontroller synchronizes the work of all the sensors. The case is not part of any subsystem, since it is a passive element and therefore not controlled by the microcontroller.

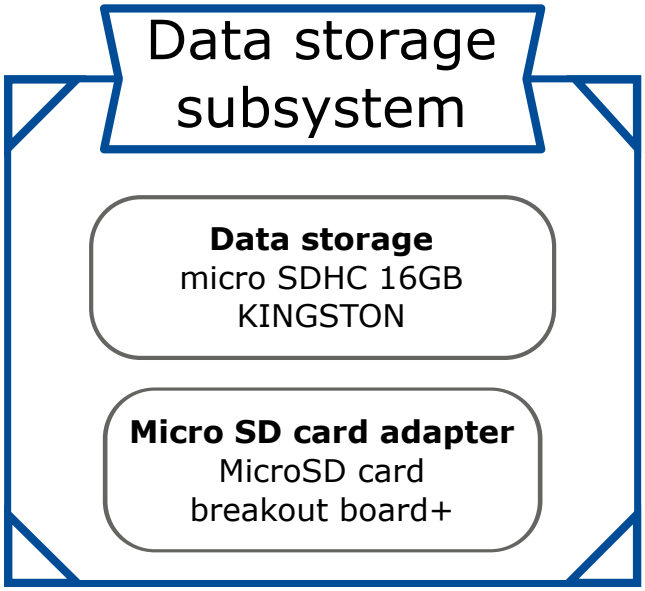


Figure 2: Modules in the data subsystem

The outer sensor subsystem shown in Fig. 4 consists of sensors that cannot perceive outside environmental changes from the interior of the airtight probe. As such they are places

[1] possible detection of nearby ships or wildlife for imaging purpuses

[2] in order to visually detect plastic debris

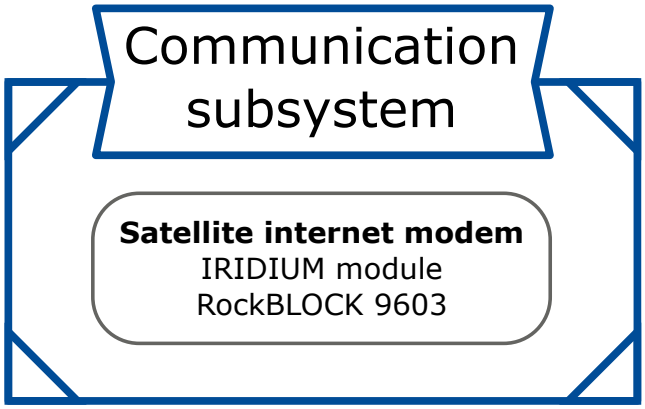


Figure 3: Modules in the communication subsystem

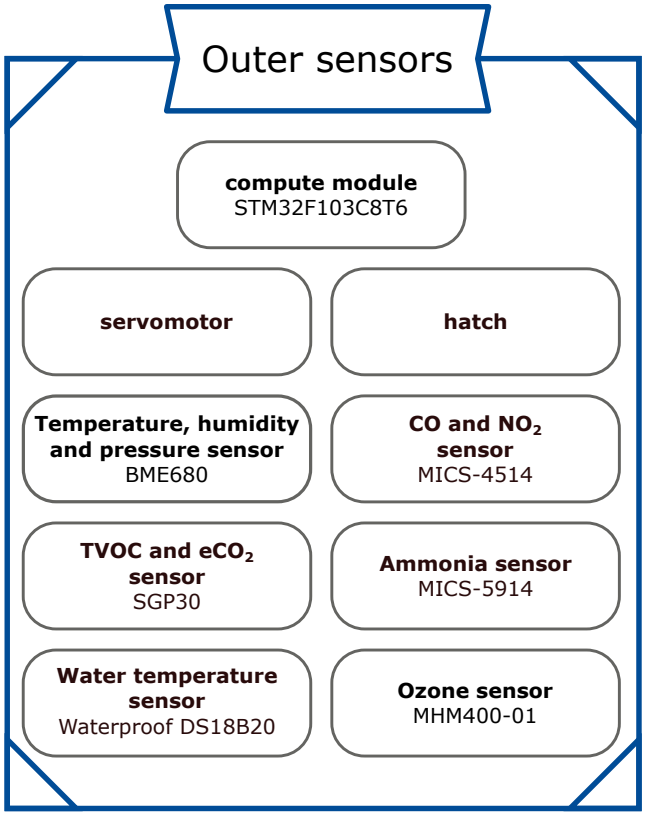


Figure 4: Modules in the outer sensors subsystem

outside of the probe casing. To protect the sensors, when they are not in use, an additional standalone module with a hatch will be used. Connection to the probe will be through waterproof connectors. The primary purpose of the outer sensors is the detection of air pollution.

The image capture and processing system shown in Fig. 5 wil contain a processing unit (a Raspberry Pi 4) separated from the microcontroller, which is capable of various image processing methods (e.g., pattern recognition, image compression). Since running the processing unit uses more power, it will be working only in short time intervals in order to process and transfer the needed data to the main microcontroller. For capturing an image, the module will also feature a camera and 360° lens assembly, which makes it possible to take panoramic photos.

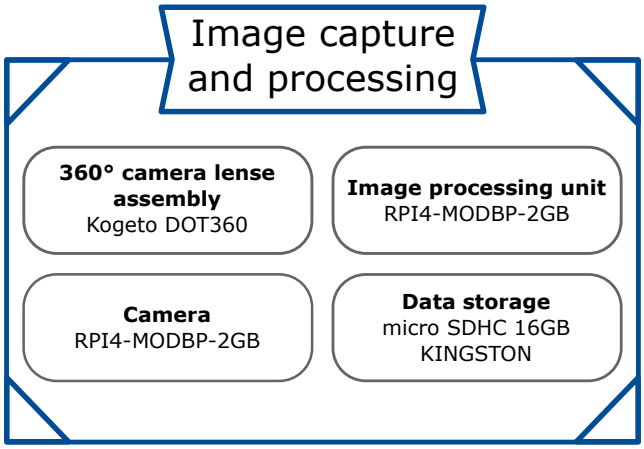


Figure 5: Modules in the image processing subsystem

3.2 Firmware design

The software on the probe is separated into three standalone modules. Their physical execution flow and physical location in microcontroller flash is shown in Fig. 6. This design will enable data gathering, over the air updates and possible recovery of essential functions in the event of an unforeseen error in programming or damage to the microcontroller flash.

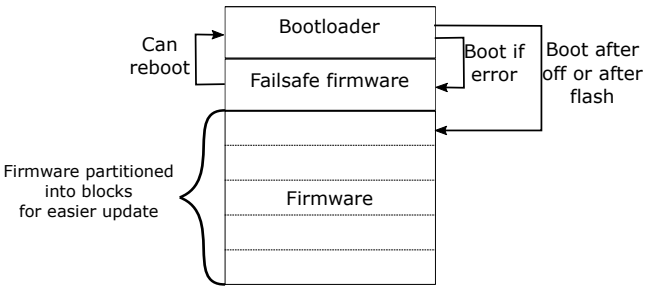


Figure 6: The outline of the firmware.

The bootloader module will be responsible for deciding and booting the appropriate firmware. This feature will enable the probe to turn itself off due to the lack of power. An additional feature will be the ability to flash prepared firmware from an external storage, enabling it to restore a previous firmware or to perform firmware updates prepared by the full firmware. In case of a fatal error during the booting of the full firmware, the failsafe version of firmware will be loaded.

The failsafe firmware will support only essential functions such as communicating with the GPS sensor, the Iridium modem, and the external storage. Its purpose is the continuation of the highest priority task and a last resort for recovering from critical errors. When booted, it will periodically send a package with GPS coordinates, requested diagnostic messages and possible direct flash commands. This

operation will allow it to try and recover the functionality of the probe or at least receive the most critical data.

The full firmware will feature FreeRTOS with tasks to read data from all of the sensors, process and save the data and communicate through the Iridium satellite network. It will be able to receive new settings (e.g., a new task schedule) or differential updates, which will be processed and a new version of the firmware prepared on the external storage device.

This modular firmware design allows for the flash memory to be partitioned into sections. The bootloader and failsafe sections will be read-only, so it will not be possible to change their content. Instead, they will be less complicated and extensively tested. There will be a section after that which can be changed by the first two modules. Also, the preparation of the full firmware allows the tasks to be compiled and linked into predetermined sized blocks, which will allow for more efficient and partial upgrades since a change in one task should not require to change or move compiled code from every other task.

3.3 Software design

The mediator of the communication between the users and the probe has the most significant role in supporting the proposed architecture. For this purpose, custom software is implemented. To accommodate the need for adaptability and accessibility of the software being developed, a dedicated server is necessary.

As of the writing of this paper, the server has an Intel® Core™ i7 930 CPU with a base frequency of 2.80 GHz and four physical cores, 12 GB random access memory (RAM), two 1 TB hard disk drives (HDDs) and a 128 GB solid-state drive (SSD). For the purpose of data redundancy and reliability, a Redundant Arrays of Inexpensive Disks (RAID) 1 is set up on the two used HDDs [6]. The server runs a Linux operating system Debian Gnu/Linux 9 (stretch).

The tasks, running on the server, are divided into three main groups: two-way communication with the probe, data handling, and hosting services and websites.

As already mentioned, the probe will send data using expensive satellite communication. The cost is calculated based on the number of sent messages [7], therefore the data are compressed to decrease the number of messages. The packages in the Iridium satellite network will be transmitted from the sending device to the predefined web service, which will run on the described server. The received data must be decompressed before it can be used. Communication can also occur in the other direction – from the server to the probe. In this case, only the critical updates are sent (e.g., firmware updates).

Data handling refers to storing, analyzing, and visualizing received data. The received data are planned to be stored in the InfluxDB database, which is designed for the time-series data. This way, data can be efficiently queried for analysis and visualization. Data analysis is necessary for controlling the probe and research. Based on the data received from the core of the probe, we can inspect its state and act upon unexpected failures with firmware updates. The data from the additional sensors will be used for the analysis of the surroundings of the probe (e.g., water pollution). For data visualization, Grafana will be used due to its supported integration with InfluxDB. Different visualization techniques will help the authors in monitoring the probe and improve the research. The dataflow is presented in Figure 7.

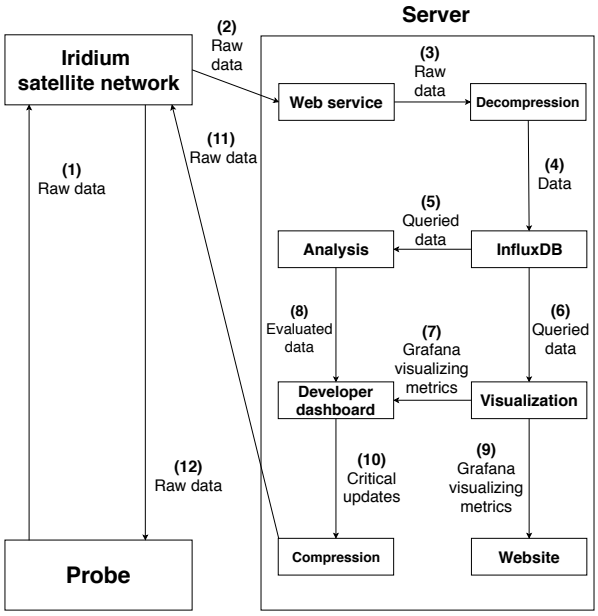


Figure 7: A high-level depiction of the data flow in the communication between the probe and the server.

The analyzed and visualized data will be available to the authors on the web-based developer dashboard, hosted on the server. The server also hosts a custom-made public website, which shows only data relevant to the general public [1]. For both websites, Nginx, Node.js and MongoDB are used. Because of the growing scale of the project and its codebase, the server also features GitLab for source code management and OpenProject for project management. All services run in separate Docker containers for easier management. Daily backups are implemented to prevent unexpected loss of data.

4. CONCLUSION

The release of the probe into the ocean is planned for late February 2020.

The main project's goal is to gather various types of information about the ocean, such as temperature and currents.

Based on other explored projects, the expected battery life of the probe is around 300 to 400 days, but the authors will try to extend that life by utilizing its smart battery management.

The collected data may prove useful to large scale initiatives such as the Ocean Cleanup [8] and Argo [5], therefore giving a further incentive to the support of the project.

Technical difficulties related to different electrical and mechanical aspects of the probe are to be expected, such as its water tightness, the lifespan of its various sensors and the battery, and electrical failures due to increased humidity and temperature. As far as the software side of the project is concerned, complications related to the maintenance and setup of the services running on the server may arise, such as data handling, data visualization, security and so forth.

5. ACKNOWLEDGMENTS

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.

6. REFERENCES

[1] ZZZ. Our homepage. http://zzz.feri.um.si/, July 2019. Accessed on 2019-5-9.

[2] M. Buoy. Homepage. https://www.makerbuoy.com/, May 2019. Accessed on 2019-5-8.

[3] designlab.ac. Omni - ocean monitoring network initiative. https://hackaday.io/project/165963-omni-ocean-monitoring-network-initiative, July 2019. Accessed on 2019-5-9.

[4] B. Sutton. Zl1six ocean floater build and voyage archive page. https://www.qsl.net/zl1rs/oceanfloater1.html, July 2016. Accessed on 2019-5-9.

[5] Argo. Homepage. http://www.argo.ucsd.edu/, July 2019. Accessed on 2019-5-9.

[6] P. M. Chen, E. K. Lee, G. A. Gibson, R. H. Katz, and D. A. Patterson. Raid: High-performance, reliable secondary storage. *ACM Comput. Surv.*, 26(2):145–185, June 1994.

[7] R. Seven. Rock seven | truly global gps tracking and messaging systems using iridium satellite | rockblock. https://www.rock7.com/products-rockblock, 2014. Accessed on 2019-7-28.

[8] T. O. Cleanup. Homepage. https://theoceancleanup.com/, July 2019. Accessed on 2019-5-9.

Efficient Collision Detection for Path Planning for Industrial Robots

László Zahorán
EPIC Center of Excellence in Production Informatics and Control,
Inst. Comp. Sci. & Control, Hun. Acad. Sci., and
Dept. Measurement and Information Systems, Budapest Univ. Technology and Economics
laszlo.zahoran@sztaki.mta.hu

András Kovács
EPIC Center of Excellence in Production Informatics and Control,
Inst. Comp. Sci. & Control, Hun. Acad. Sci.
andras.kovacs@sztaki.mta.hu

ABSTRACT

Efficient collision detection is crucial for the success of automated process planning and path planning for robotic manipulation and assembly. Yet, collision detection for articulated industrial robots holds various challenges. This paper gives an overview of these challenges, and presents an efficient implementation of collision detection techniques for such robots. The applicability of the developed techniques to support path planning in an industrial test case is also illustrated.

1. INTRODUCTION

A crucial requirement towards automated process planning and path planning methods for robotic operations is that they must guarantee the geometric feasibility of the computed plans, by ensuring that no collisions occur during the movement of the robot and the manipulated objects. At the same time, collision detection is typically the computationally most challenging sub-problem of path planning [1, 3]. Particular challenges in collision detection for industrial robots lie in the following:

- Collision detection methods must be able to find *all potential collisions* of every moving and static object in the work cell, including the robot, the gripper, the workpiece, the fixture, as well as all static elements of the cell.
- The above objects are all characterized by *complex free-form geometries*.
- While the *continuous motion* of the robot must be checked for collisions, nearly all approaches in computational geometry focus on checking static configurations. To overcome this discrepancy, continuous collision detection must be reduced to an appropriately defined series of queries on static configurations.
- The kinematic chain of typical articulated industrial robots consists of 6 or more robots links. The motion of these links can be characterized in the *joint configuration space* of the robot, i.e., by the vector of joint angles. At the same time, effective tasks must be planned and collisions must be detected in the *Cartesian space*. Hence, the *mapping* between the two representations must be maintained at all times. For this purpose, *forward kinematic transformation* calculates the position of the robot links and the grasped objects in the Cartesian space from the joint angles, whereas *inverse kinematics* search for the joint angles that realize a given position in the Cartesian space. Yet, for many kinematic structures, the inverse kinematic calculation is a challenging computational problem with non-unique solutions.
- Certain types of contact between objects are allowed, e.g., between neighboring robot links or between the gripper and the workpiece. Moreover, the allowed types of contact may vary over time, e.g., a workpiece can touch the fixture when the robot inserts the workpiece

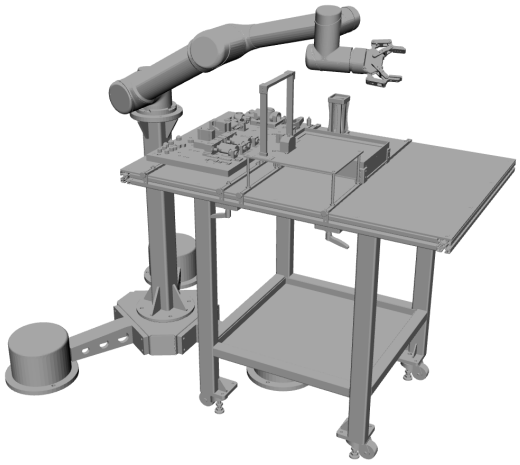


Figure 1: Work cell with a UR5 robot and a Robotiq gripper.

DOI: https://doi.org/10.26493/978-961-7055-82-5.19-22

into the fixture, but the same contact is forbidden during other motions. Hence, *collision rules* must be maintained dynamically.

- The configuration of the robot and the work cell may vary over time, e.g., when the robot grasps or releases the workpiece. These configuration changes must be managed, and pre-computation techniques must be handled with care.

- Since process planning and path planning methods rely on iteratively checking a vast number of candidate robot motions, the *computational efficiency* of collision detection is crucial.

Various generic-purpose libraries are available today for collision detection, such as the Proximity Query Package (PQP) [2] or the Flexible Collision Library (FCL) [4]. These libraries offer collision and distance queries for static configurations of free-form 3D solid objects (although FCL handles some restricted forms of continuous collision queries as well). Hence, they must be extended substantially to respond to the above challenges.

This paper presents a library for efficient collision detection for industrial robots. The library is built on the top of the generic-purpose PQP collision detection engine, and extends it with various kinematic and geometric calculation methods to serve the needs of robotic process planning and path planning. On the top of these collision detection techniques, the library contains an implementation of the Rapidly-exploring Random Trees (RRT) single-query probabilistic path planning algorithm [3], as well as the Probabilistic Roadmaps (PRM) multi-query path planner [1], which use the continuous collision queries as a so-called local planner (i.e., checking the direct movement of the industrial robot between two configurations). The paper gives an overview of the implemented collision detection techniques and demonstrates their computational efficiency in industrial case studies.

2. COLLISION DETECTION FOR ARTICULATED INDUSTRIAL ROBOTS

As pointed out above, collision detection for industrial robots requires extending general-purpose collision libraries in two main directions: (1) robot motions specified in the joint configuration space must be mapped into the Cartesian space using forward kinematic calculations; and (2) continuous collision detection for the robot motion must be reduced to checking an appropriate series of static robot configurations.

A static configuration c of an m-axis industrial robot can be characterized by a vector of m joint angles in the form of $c = (\alpha_0, ..., \alpha_m) \in C$, where C is the configuration space of the robot, defined by its joint limits. As usual, we focus on linear movements in the joint configuration space. Accordingly, the movement between start configuration c_0 and end configuration c_1 is interpolated by $c(t) = c_0(1-t)+c_1t$, $t \in [0, 1]$. This movement is considered free of collisions if every static configuration $c(t)$ is collision-free for $t \in [0, 1]$.

Collision detection must capture every so-called collision object in the work cell, including the robot (with a separate collision object for each robot link), the gripper (with interchangeable geometric models corresponding to different degrees of opening), the workpieces, as well as all other objects in the cell. While the geometry of each collision object is characterized by a triangle mesh representation, given in an STL file, a configuration is described by a homogeneous transformation matrix for each collision object. This matrix can be computed by forward kinematics from the robot configuration.

Collision rules between pairs of collision objects define whether the contact of the two objects is considered as a collision or not. By default, the contact of the neighboring robot links, as well as the contact between the robot base and the static work cell elements are allowed. These default rules can be overridden dynamically depending on the task executed by the robot.

2.1 Sampling-based Collision Detection

Sampling-based collision detection is the most common approach in robotics to check robot movements. The continuous movement $c(t)$ is sampled by looking at a finite set of static configurations $c(t_i)$, with $i = 1, ..., n$. Since the motion is given in the joint configuration space, the sampling rate is controlled by angle δ that specifies the maximum distance between neighboring samples $c(t_i)$ and $c(t_{i+1})$, using the maximum norm over different joints. The movement is classified as collision-free if and only if every static sample is collision-free.

A critical issue is the choice of parameter δ: using a low value is computationally demanding, whereas increasing δ also increases the risk of missing a collision. The proper value must be determined for each application individually.

Since typical path planning algorithms cannot exploit any information on the location of collisions, the checking of continuous movements can be interrupted upon finding the first collision. This implies that the performance of the algorithm on colliding motions can be improved significantly by the proper ordering of the collision queries.

As a heuristic, the probability of collision rises with the distance from known collision-free configurations. Accordingly, the implemented algorithm checks the start and end configurations first, whereas in the iterative step, it bisects the previous motion sections until the distance decreases below the given threshold δ. Furthermore, when checking a given static configuration, collision queries for different pairs of collision objects are ordered by the probability of collision, estimated based on historic records. In contrast, all collision queries must be executed on collision-free movements.

2.2 Conservative Advancement

Instead of the above heuristic method for checking continuous movements, another approach that provides a formal guarantee of collision-free continuous movements is strongly preferred. Such an approach is the so-called Conservative Advancement (CA) method [6], which achieves this by using distance queries on static configurations. The approach exploits that if, in a collision-free configuration c_1, the distance between two collision objects is d, and the relative displacement of these objects in configuration c_2 compared

to $c1$ is at most d, then these object do not collide in c_2 either.

The key difficulty with applying CA to industrial robots is that robot movements are defined in the joint configuration space, whereas the distance queries compute the allowed displacement in the Cartesian space. To overcome this discrepancy, an upper estimation of the Cartesian displacement caused by any given joint motion is required. This paper compares two implementations of this upper bound: the original bound from [6] and an improved bound. The bounds use the following information:

- $\varphi = (\varphi_0, \varphi_1, ..., \varphi_m)$: for an m axis industrial robot, the difference of joint angles between the start and end configurations of the motion.
- d_i, a_i: Denavit-Hartenberg parameters of the ith robot link (joint offsets in z and y).
- l_i: length of ith link from the lower joint node to the farthest point of the robot link geometry.
- r_i: distance of upper and lower joints of the ith robot link.

Based on these input data, an upper bound of $\delta_{i,j} = (\sum_{x=i}^{j}((r_x + l_x)\sum_{y=i}^{x}(\varphi_y)))$ can be given on the relative displacement of the ith and jth element of the kinematic chain.

Again, the appropriate choice and ordering of distance queries is crucial for computational efficiency. For this purpose, the proposed algorithm maintains a queue of so-called CA tasks, each CA task consisting of a pair of collision objects, as well as a start and end configuration of the motion. At every point in time, the queue is ordered by the length of the motion and the historic likelihood of collision between the two objects. Initially, the queue contains one CA task for each relevant object pair with the original start and end configurations of the motion.

In each iterative step, the first CA task is taken from the queue, and the distance of the two collision objects is computed in the mid-point of the motion, i.e., configuration $c(\frac{1}{2})$. If the query returns with a positive distance, then configurations $c(\frac{1}{2}^-)$ and $c(\frac{1}{2}^+)$, i.e., the first and last proven collision-free configurations before and after the mid-point are determined according to the above bound. If these are different from the start and end configurations of the original CA task, then two new CA tasks corresponding to $[c(0), c(\frac{1}{2}^-)]$ and $[c(\frac{1}{2}^+), c(1)]$ are created and inserted into the queue. The process is terminated when a collision is encountered or the queue is empty, where the latter means that the original movement is proven to be collision-free.

The accuracy of the upper bounds on the displacement greatly influences the number of distance queries executed. The original bound of [6] uses the bound for most distant objects in the kinematic chain for every pair of collision objects. The proposed minor improvement is to apply the bound $\delta_{i,j}$ corresponding to the specific objects.

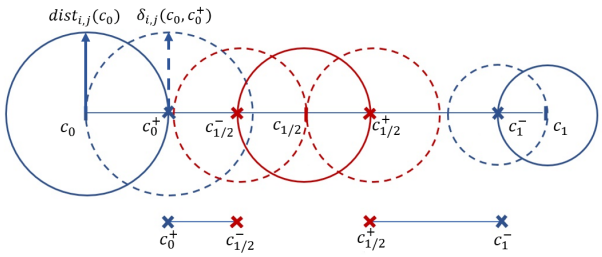


Figure 2: Conservative Advancement.

2.3 Comparison of the Two Approaches

The boolean collision queries used by the sampling-based approach are an order of magnitude faster than distance queries. CA can balance this difference by taking larger steps, and therefore executing less queries, especially in large open spaces. A crucial qualitative difference between the two approaches is that CA gives a formal guarantee of the geometrical feasibility of continuous robot movements. Moreover, CA can be naturally extended to maintain a specified safety distance between the objects in the work cell.

3. COMPUTATIONAL EXPERIMENTS

The presented algorithms were implemented in C# in MS Visual Studio. The solution contains separate projects for PQP (a native C++ project with C# wrapper), CollisionLibrary (a C# class library including UR5, UR10 robot models, RobotiQ and other grippers, implementations of collision detection, path planning, and path smoothing algorithms), CellVisualizer (a WPF application for the graphical animation of the work cell and the computed paths), as well as a console application for executing tests and measurements.

A reference solution was developed in the commercial robot simulation and off-line programming software called RoboDK [5] for verifying the correctness of the results and for comparing the computational performance to the state-of-the-art. RoboDK has a Python API to implement custom algorithms, and offers high-level functionality for simulating robot movements and collision detection. In the presented experiments, the `Move_JTest(configA, configB, samplFreq)` method of RoboDK was used, which implements a sampling-based approach for checking robot movements. It should be noted that this method finds every colliding object pair (although this information is not used later), whereas our implementation looks for the first collision only. Some difference in the performance of the two approaches may also stem from the difference of the publicly available UR5 robot geometry adopted in our implementation and the robot model applied in RoboDK. All experiments were performed on an Intel i5-4200M 2.5GHz dual-core CPU and 8GB RAM.

3.1 Experiment Design

The work cell used in the experiment contains a UR5 robot equipped with a Robotiq gripper, as well as a robot stand and a work table with fixtures for assembling a ball valve. The number of collision objects is 10, with 165 000 triangles altogether, resulting in 27 active collision rules. The experimental workspace has large collision-free spaces as ca. 80% of checked movements are collision-free. The rate of colliding and non-colliding movements greatly affects perfor-

mance, since the checking of colliding movements can be interrupted upon finding the first collision.

Computational experiments were performed on a set of 5000 continuous robot movements arising when building a PRM on the above work cell with 1000 random robot configurations and 5 neighbors per node. The average length of the robot movements was 45–50° in each robot joint.

Four different collision detection techniques were compared: sampling in RoboDK and in the proposed implementation, as well as CA in the proposed implementation with the original displacement bound of [6] and its improved version.

3.2 Experimental Results

The computational results are displayed in Table 1, which displays the key parameters, as well as the results achieved by the four algorithms. Both sampling-based approaches used a 1° sampling rate for the joint movements, without giving a formal guarantee of the geometrical feasibility of the checked motions or maintaining a safety distance. With this sampling rate, our implementation classified 2 out of 5000 colliding robot motions incorrectly as collision-free. A higher number of mistakes by RoboDK probably stems from the different geometrical models used.The efficient implementation resulted in a 23 times speedup compared to RoboDK.

In contrast, the two CA implementations both provided a guarantee of geometrical feasibility and could maintain a safety distance. At the same time, in order to facilitate a comparison between CA and sampling, a safety distance of 0 mm was used in the experiments. Moreover, allowing a relative tolerance of 3% in the PQP distance queries resulted in a considerable speedup of the algorithm, without any incorrect classifications on this test set. As a result, the two CA implementations returned correct and identical classifications. The improved displacement upper bound resulted in a 2.89 times speedup compared to the original upper bound, and computation times only 19% higher than for sampling. We regard this as a favorable tradeoff for the formal guarantee on the feasibility of the robot motions.

Table 1: Experimental Results

	Sampling (RoboDK)	Sampling (own)	CA (orig.)	CA (impr.)
Sampling	1°	1°	-	-
Safety dist.	-	-	0 mm	0 mm
Guarantee	-	-	✓	✓
Time [mm:ss]	38:08	01:41	05:47	02:00

4. CONCLUSIONS

The paper gave an overview of the computational challenges in continuous collision detection for articulated industrial robots, and presented alternative approaches to tackling this challenge. An efficient implementation of the sampling and the conservative advancement approaches was introduced, with various improvements compared to earlier algorithms in the literature. In computational experiments, the proposed sampling-based algorithm achieved a 23 times speedup compared to a similar algorithm of a commercial software, whereas an improved displacement bound for conservative advancement resulted in a nearly three times speedup w.r.t. using the earlier bound from the literature.

The presented collision detection library is a key component of a process planning and path planning toolbox for industrial robots under development. Future work will focus on the completion of the robotic path planning algorithms, especially PRM and RRT, on top of the presented collision detection library. We plan to apply this library to process planning in various industrial applications, including a camera-based robotic pick-and-place work cell and the assembly of electric components. A research challenge is the handling of constraints and performance measurements defined in the Cartesian task space, such as linear motions or Cartesian speed limits, while planning in the robot joint configuration space.

5. ACKNOWLEDGMENTS

This research has been supported by the ED_18-2-2018-0006 grant on “Research on prime exploitation of the potential provided by the industrial digitalisation” and the GINOP-2.3.2-15-2016-00002 grant on an “Industry 4.0 research and innovation center of excellence”. A. Kovács acknowledges the support of the János Bolyai Research Fellowship.

6. REFERENCES

[1] L. E. Kavraki, P. Svestka, J. C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.

[2] E. Larsen, S. Gottschalk, M. C. Lin, and D. Manocha. Fast proximity queries with swept sphere volumes. In *Proc. IEEE Int. Conf. Robot. Autom.*, pages 3719–3726, 2000.

[3] S. M. Lavalle and J. J. Kuffner. Rapidly-exploring random trees: Progress and prospects. In *Algorithmic and Computational Robotics: New Directions*, pages 293–308, 2000.

[4] J. Pan, S. Chitta, and D. Manocha. FCL: A general purpose library for collision and proximity queries. In *IEEE International Conference on Robotics and Automation*, pages 3859–3866, 2012.

[5] RoboDK. Simulation and OLP for robots, 2019. https://robodk.com/.

[6] F. Schwarzer, M. Saha, and J.-C. Latombe. *Exact Collision Checking of Robot Paths*, pages 25–41. Springer, 2004.

Sensitivity analysis for p-median problems

Ágnes Vida
Institute of Informatics
University of Szeged
Vida.Agnes.1@stud.u-szeged.hu

Boglárka G.-Tóth
Institute of Informatics
University of Szeged
boglarka@inf.u-szeged.hu

ABSTRACT

Network location problems are commonly associated with real world situations such as locating a new facility in a city or setting up a new server in a computer network. In these real world situations changes can come up quite often, such as a closed road because of an accident, or a broken connection in the network. These kind of problems give the motivation of this paper. Our aim was to inspect, how an existing network operates, when something has changed and how sensitive a p-median solution is to the same changes. During our research we concentrated on deleting edges and solving p-median problem. In the p-median problem we try to locate p facilities so the sum of the distances between the facilities and the demand points is minimal. During the sensitivity analysis we deleted different amounts of the edges and we also tested the problem by locating 1, 2 or 3 medians to get a complex picture about the computational results. During our work, we concentrated on how the solution and the location of the medians change. To get a complex picture about the results we used two different graphs. After the tests we saw, that according to the shape of the graph, the number of the changes can be quite different. On the one hand, when we worked with a graph which is easy to cut, the changes of the solution was unstable, and relatively big, while with a well-balanced graph, the changes were not that significant. On the other hand, the location of the facilities did not change too much with either graph.

Categories and Subject Descriptors

G.2.2 [**Mathematics of Computing**]: Graph TheoryNetwork problems; G.1.6 [**Mathematics of Computing**]: Optimization

General Terms

Theory, application

Keywords

p-median, sensitivity analysis, Mixed Integer Programming

1. INTRODUCTION

Locating facilities is a common problem when we want to work with networks such as communication networks or traffic flows. In order to work with these networks in a mathematical aspect, we use an abstract model, a finite (often weighted, non-oriented) graph G which has a set of vertices (V) and connecting edges ($E \subseteq \{V, V\}$). Weights are commonly attached to the edges, which represent the cost of the transportation or the length of the road. Network locating problems include several different questions, such as absolute center and median problems or p-center and p-median problems. For a good introduction in facility location problems on networks, see [2].

In this paper we will concentrate on the p-median problem. The aim of the p-median problem is to find the optimal placement of p facilities on the network. The optimal location of the medians means that the sum of the distances between the vertices and the closest facility is minimal. To find the optimal location of the facilities, we must complete the following constraints: one demand point can be assigned to only one facility and maximum one facility can be placed at one vertex. We also have to provide that exactly p medians are located. To understand the p-median problem we studied the article by Hakimi [3].

During our research we always worked with non-oriented n-vertex graphs with edge weights, which can also be represented with a square matrix, called weight matrix. Since we are working on a graph, distance is always understood as shortest path on the graph. From the weight matrix we need to determine the distance matrix with some methods computing shortest paths on a graph.

The aim of this paper is to find out how sensitive a p-median solution is when we delete edges from the graph. The motivation is the use of traffic networks, to inspect how the traffic and the optimal locations of the p-medians change if something happens in the network. In this paper we concentrated on deleting edges with different likelihood, to inspect how the cost and the optimal location of the p-medians change. During our research we tested every case on two different graphs to get a more complex picture about the computational results.

Similar works has been done investigating the effect of changing network density in [4], and also studying facility reliability issues in Berman et al. [1].

DOI: https://doi.org/10.26493/978-961-7055-82-5.23-26

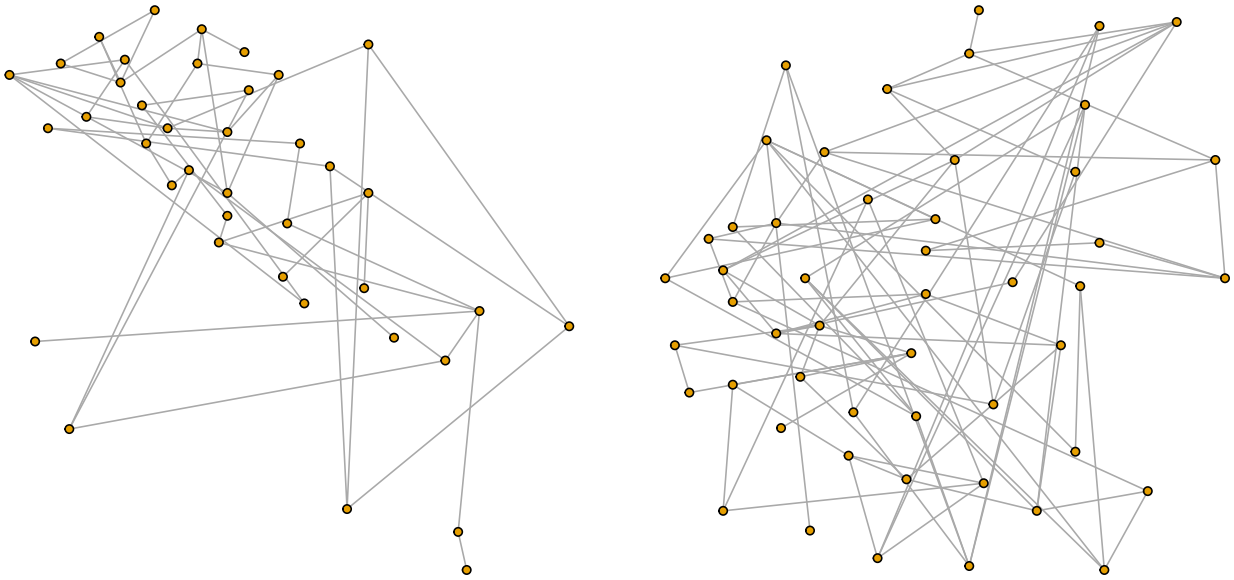


Figure 1: Graph of Italy with cities on the islands (left), graph of Germany (right)

2. THE P-MEDIAN PROBLEM

The aim of solving a p-median problem is to locate exactly p medians by minimizing the sum of the distances between the demand points and the located facilities and also completing a few constraints mentioned below. In order to solve the problem we used the following model which is based on a non-oriented weighted n-vertex graph. We worked with the graph as a set of vertices (V) connected by weighted edges ($E \subseteq \{V, V\}$).

$$\min \quad \sum_{i,j \in V} d_{ij} y_{ij} \tag{1}$$

$$\text{s.t.} \quad \sum_{j \in V} y_{ij} = 1 \quad \forall i \in V \tag{2}$$

$$\sum_{i \in V} z_i = p; \tag{3}$$

$$y_{ij} \leq z_j \quad \forall i, j \in V; \tag{4}$$

To represent the distances between the vertices we used a parameter d_{ij}, which is a squared matrix. The j-th column of the i-th row represents the length of the shortest path between the i-th and j-th vertex.

We also used two set of variables: y_{ij} and z_i. The binary variable y_{ij} is to sign that the i-th demand point is supplied by the j-th facility and z_i is a binary variable to sign if we locate a facility on vertex i. The aim of the model is to minimize the sum of the distances between the demand points and the facilities. The optimal solution must complete three constraints: (2) gives the condition, that exactly one facility has to be assigned to one vertex; the number of located facilities must be p which is required by constraint (3); and finally, that a demand point can only be assigned to a facility, that is located, see (4).

Our aim is to minimize the sum of the distances between the demand points and the closest facility, but we do not have any direct constraint for that. That is because the model minimizes the sum of $d_{ij}y_{ij}$, and so in this way the model automatically chooses the best solution, which is the minimal distance between a demand point and the assigned facility.

3. METHODS

Our choice to get the computational results was AMPL, which is A Mathematical Programming Language. First we implemented the model to solve the p-median problem. In order to obtain the optimal solution of this Mixed Integer Programming (MIP) problem we used a commercial solver called CPLEX.

Our goal was to inspect how the solution reacts to changes, especially deleting edges. At first, we always ran the algorithm with no changes to inspect the original solution, than we modified it with different amounts. We increased the number of deleted edges always by 10%. After deleting 40% of the edges we decided to stop, because we noticed that the graph fell apart. We also checked how the solution operates when we want to locate 1, 2 or 3 medians with the different number of deleted edges. In every variation of the number of deleted edges and the number of located medians we made 20 test runs.

In order to delete edges, first we made a mapping, so we could work with the edges like an ordered set. Then we randomly choose the right number of edges to delete from the graph. To implement this random choose, we used some of the built-in functions of AMPL. Deleting the desired amount of edges was done uniformly. In a cycle we have generated a random integer value k from 1 to the number of still existing edges and removed the kth edge, until we removed the required number of edges.

After deleting the chosen edges, we determined the distance matrix on the modified graph. In order to get the distances, we used the definition of shortest path, and the Ford-Fulkerson algorithm.

4. RESULTS AND ANALYSIS

The algorithm was tested on two data files, modeling the largest cities and roads of Italy and Germany. In order to have a better view on the results, we draw the graphs using the real coordinates of the cities in Figure 1. We segmented the tests according to how many medians are located (1, 2 or 3) and the number of the deleted edges (0, 10%, 20%, 30% and 40%). We made 20 test runs for every segment and reported the average, the minimum and maximum changes. The tests show that the shape of the graph influences the results a lot.

The results for the comparison of the solutions in changes of the cost, as well as in the changes in the locations for both the Italian and German networks can be seen in Table 1. Next we discuss the results that can be read from the table.

4.1 Changes of the cost

In the Italian graph, when we deleted 10% of the edges, the solution increased by 309% on average. This is quite huge change as in the German graph this increase was only 10%, and all changes was smaller than 134%. But let us concentrate first the results for the Italian graph. We have found that the solution was very unstable for this graph. When we located 1 median, by deleting 10% of the graph the minimal increase was by 22% while the maximal increase was around 1086% and the average increase was 309%. By deleting the double amount of the edges, the average increase of the objective function was also about double (615%), the minimal and maximal increase was 82% and 1314%, respectively. We also tested the results during deleting 30% and 40% of the edges. With a 30% loss, the average increase of the objective value was 1275%. After deleting 40% percent of the edges we stopped increasing the number of deleted edges, because we found that the increase of the objective value is too high (1387% on average), and we have seen that this happened because the graph fell apart. We can see a big jump in the minimum changes in costs from 20% to 30%, when from 82% it grew to 737%. Similar, but not so evident jumps can be seen in the minimum change when 2 or 3 medians are located.

When we located two medians, the results were quite similar. The main difference showed, when we deleted only 10% of the edges. In this case the cost increased by only 51%. By deleting 20% of the edges the average increase of the cost was around 345%, by deleting 30% of the edges, the cost increased by 925% on average, and next to a 40% loss of the edges, the objective value increased by 1421%.

The problem of locating 3 medians showed similar results as the 2-median problem. The average increase of the cost was 79% when we only deleted 10% of the edges and 1352% when we deleted 40% of the edges. Altogether, we can see that locating more medians makes less changes in cost in general for this graph (except 40% deleted edges), although not significantly.

When we used the graph of Germany, the results were different. The tendency was the same, but the amount of the rise of the solution was much lower. First we located only one facility. By deleting 10% of the edges and locating one facility the cost increased by 10% (minimum 1% and maximum 23%). By deleting more edges, the solution increased slightly, but not as much as with the Italian graph. When we deleted 40% of the edges, the cost increased with 84% on average (minimum 57%, maximum 124%). As with the previous graph, we repeated the test by locating two and then three facilities. The results were not that different, by deleting 10% of the edges the increase of the cost was 10% with 2 medians and 9% with 3 medians. Even by deleting 40% of the edges the increase of the cost was around 117%. In this case, the changes compared for the number of medians is quite balanced, as opposed to the Italian case. During the tests we could see, that there are vertices where facilities are more likely to be placed.

4.2 Changes of the locations

We also paid attention to the location of the medians. We have made a ranking on the number of times a vertex was selected as a median in the 20 runs. When we had to locate only one facility with the Italian graph, we saw that the original location was only the 11th most likely place to put the facility at. With the German graph the result was much better: the original placement of the median was the 4th in the ranking. When we solved a 2-median problem, the results were the same with the two graphs: the original locations of the medians were in the 5 most likely vertices to put a facility at. With 3-median problem the results were also similar. The original location of the medians were still in the top 5 vertices to locate a facility on. All in all we can say, that the location of the facilities can change according to the number of deleted edges, but the original locations are almost always in the top 5 vertices.

During our research we also made computations to inspect, how far the new locations from the old ones are. To get an objective picture about the results we always used the original graphs and the shortest distances between the old and the new locations of the medians. When we located more than 1 median, we choose the smallest change between the old and the new locations in pairs. Although the values were higher with the German graph, the tendency was the same. With the Italian graph when we located only one facility and deleted 10% of the edges the distance between the old and the new one was 16.6 on average (minimum 4, maximum 51). Even with deleting 40% of the edges the average distance was 61.1 (minimum 8, maximum 81). This small amount of change is interesting if we see, that the cost increased much more under these conditions. When we located 2 medians, the changes were even smaller. With a 10% loss, the distance between the old and the new facilities was 3.9. When we deleted 40% of the edges the distance was still around 9%. When we located 3 medians, the new locations were farther than with the 2- and 3-median problems. In this case the minimal change was around 10 (by deleting 10% of the edges) and the maximal change was about 44 (by deleting 30% of the edges).

With the German graph we did not see significant differences. During 1-median problem and a 10% loss the dis-

Table 1: Results for the comparison of the solutions in changes of the cost, as well as in the changes in the locations for both the Italian and German networks.

	Number of facilities	Deleted edges	Change of the cost			Change of the location		
			Min	Avg	Max	Min	Avg	Max
Italy	1	10%	22%	309%	1086%	4	16.6	51
		20%	82%	615%	1314%	0	18.4	51
		30%	737%	1275%	1524%	4	39.6	62
		40%	835%	1387%	1613%	8	61.1	98
	2	10%	1%	51%	170%	0	3.9	39
		20%	106%	345%	718%	0	8.3	66
		30%	325%	925%	1546%	0	8.2	55
		40%	878%	1421%	1831%	0	9.0	47
	3	10%	2%	79%	150%	0	10.4	132
		20%	34%	233%	613%	0	31.7	128
		30%	298%	616%	907%	0	44.3	153
		40%	904%	1352%	1854%	0	40.0	124
Germany	1	10%	1%	10%	23%	0	81.6	247
		20%	9%	27%	49%	0	131.9	265
		30%	23%	57%	86%	100	166.8	264
		40%	57%	84%	124%	178	188.4	264
	2	10%	2%	10%	19%	0	40.3	195
		20%	13%	26%	47%	0	75.7	276
		30%	21%	59%	98%	0	105.7	276
		40%	81%	104%	127%	0	105.2	276
	3	10%	2%	9%	19%	0	69.9	471
		20%	4%	24%	42%	0	117.5	614
		30%	34%	55%	79%	0	84.7	276
		40%	94%	117%	134%	0	132.3	440

tance between the old and the new location of the facilities was 81.6 on average (minimum 0, maximum 247). Even with a 40% loss of the edges the distance was 188.35 on average (minimum 178, maximum 264). When we located 2 medians, the average change was around 40 (minimum 0, maximum 195) with deleting 10% of the edges and 105,2 (minimum 0, maximum 276) with a 40% loss. With the 3-median problem, the minimal change was about 70 and the maximal change was about 132.

5. CONCLUSION AND FUTURE PLANS

By the results we can see that the change of the solution during deleting edges highly depends on the shape of the graph we work with. If the graph has just a few connections between two main parts, deleting of the edges can cause very high costs. If we work with a well-balanced graph, even by deleting 40% of the edges the solution changes just a little. We also could see that the location of the facilities do not change too much. In the future we are planning to inspect how a non-oriented weighted graph reacts to p-median problem when we change the weights of edges, like when traffic changes or when we add new edges. We also would like to check if we can make clusters according to how likely that a facility is located in a vertex. Furthermore we want to inspect how a non-oriented weighted graph reacts to other problems such as p-center problem during similar conditions.

6. ACKNOWLEDGMENTS

This research was supported by the European Union and co-financed by the European Social Fund (EFOP-3.6.3-VEKOP-16-2017-00002).

7. REFERENCES

[1] O. Berman, D. Krass, and M. B. C. Menezes. Facility reliability issues in network p-median problems: Strategic centralization and co-location effects. *Operations Research*, 55(2):332–350, 2007.

[2] M. S. Daskin. *Network and Discrete Location: Models, Algorithms and Applications*. John Wiley and Sons, New York, 1995.

[3] S. L. Hakimi. Optimum locations of switching centers and the absolute centers and medians of a graph. *Operations Research*, 12(3):450–459, 1964.

[4] X. Zhao, K. Carling, Z. Dan, and J. HÃěkansson. Network density and the p-median solution. Technical report, Högskolan Dalarna, Borlänge, 2013.

A two-stage heuristic for the university course timetabling problem

Máté Pintér
University of Szeged, Institute of Informatics
Árpád tér 2
Szeged, Hungary, 6720
pmate955@gmail.com

Balázs Dávid*
InnoRenew CoE
Livade 6
Izola, Slovenia, 6310
balazs.david@innorenew.eu
University of Primorska, FAMNIT
Glagoljaška ulica 8
Koper, Slovenia, 6000
balazs.david@famnit.upr.si

ABSTRACT

This paper presents a two-stage solution method for the curriculum-based university course timetabling problem. First, an initial solution is created using a recursive search algorithm, then its quality is improved with a local search heuristic. This method utilizes a tabu list of forbidden transformations, as well as a destructive step at the end of each iteration. The algorithm is tested on datasets of the 2007 International Timetabling Competition.

Keywords

University course timetabling; Local search; Heuristic

1. INTRODUCTION

Creating the timetable of employees is a crucial task for any organization, and educational institutions are no exceptions. While timetabling problems can sometimes be extremely hard, they are still solved manually in many cases. This process can take a long time, and the efficiency of the resulting timetables is not guaranteed.

There are several different types of timetabling problems that have to be solved in academia, mainly connected to the students or the employees. This paper will cover such a specific problem, namely university course timetabling.

The first section of this paper will introduce the field of course timetabling, and present the curriculum-based university course timetabling problem in detail. After this, we present a two-stage heuristic for the solution of the problem, which is then tested on datasets of the 2007 International Timetabling Competition. These instances are based on real world timetables of the University of Udine, Italy.

*Supervisor

2. COURSE TIMETABLING

Course timetabling is an optimization problem, where resources (courses, rooms, teachers) are allocated to create a timetable that satisfies given constraints. The problem schedules courses and teachers into available time-slots, while trying to avoid the different types of conflicts that can arise. Many variations exist for the course timetabling problem, but the most important ones are the following:

- *Curriculum-based Course Timetabling*: courses belong to one of several curricula, and courses of the same curriculum cannot be scheduled in overlapping time-slots.
- *Post Enrollment based Course Timetabling*: the problem also considers the students enrolled to the courses, and introduces several constraints connected to their attendance.
- *Examination Timetabling*: similar to the general timetabling problem, but considers exam committees instead of teachers, and also deals with the availability of the students for their required exams.

This paper will give a solution algorithm for the curriculum-based course timetabling problem. The following sections will define the problem, present its necessary resources and constraints, and give a short literature overview of the field.

2.1 Problem definition

The curriculum-based university course timetabling problem schedules a set C of courses over a horizon of d days to create a timetable. Each day is divided into s time-slots, and a course has to be assigned to one or more consecutive slots (depending on its length). A room has to be selected where the course will take place, and a teacher is also assigned as the instructor. Courses can belong to different curricula, introducing additional constraints to the assignment. Courses of the same curriculum usually cannot overlap, or should be

DOI: https://doi.org/10.26493/978-961-7055-82-5.27-30

scheduled close to each other in time. The constraints of the problem belong into two different groups.

Hard constraints should always be respected, and a timetable is not allowed to violate any of them. The most common hard constraints are the following:

- *Course assignment:* The timetable must contain all courses, and every course has to be assigned to exactly one room, time-slot and teacher.
- *Room availability:* A given room cannot have more than one assigned course at the same time-slot.
- *Teacher availability:* A given teacher cannot be assigned to more than one course at the same time-slot.
- *Room capacity:* A course cannot be scheduled to a room with less capacity than the number of students on the course. Some papers consider this as a soft constraint.
- *Teacher availability:* Teachers can have time-slots when they are not available. Naturally, no course can be assigned to a teacher at a time-slot where they are not available. Some papers consider this as a soft constraint.

Soft constraints can be violated, but they come with a penalty. The typical soft constraints are the following:

- *Compactness:* Isolated courses in a curriculum should be avoided. A course is isolated, if no other course of the same curriculum is scheduled in its previous or next time-slot.
- *Room stability:* Courses of the same curriculum should be assigned to the same room, if possible.
- *Minimal number of days:* Courses of the same curriculum should be spread out over the week: they should be scheduled to a given d amount of days.
- As it was mentioned above, some of the hard constraint (room capacity, unavailability of teachers) can also be considered as a soft constraint.

The objective of the problem is to create a timetable that does not violate any hard constraint, and has a minimum penalty associated to the violations of its soft constraints.

2.2 Literature overview

University course timetabling has been researched intensively in the past decades. The problem itself is NP-complete, which was proven by Even et al. [8], and as a result, many different solution methods were considered for its solution. A good overview of these is given by Bettinelli et al. [4] and Babaei et al. [2]. In the following, we will present some the most important of the approaches. An early mathematical model was given by [1], who consider it as a 3-dimensional assignment problem between courses, time-slots and rooms. A more efficient, two-phase mathematical model is presented by Lach et al. [9], where only courses and time-slots are assigned using a binary variable, and the possible rooms for each time-slot are described by an undirected graph. This approach reduces the model size, and is able to provide solutions for significantly bigger instances.

As mathematical approaches can result in a large model even for relatively small instances, various heuristic methods were also developed for the problem. Different variations of the local search were presented, such as the tabu search of Lü and Hao [10] or the simulated annealing of Bellio et al. [3]. Improved versions of Dueck's Great Deluge algorithm [7] - a genetic algorithm with a philosophy close to local search - were also been published [5]. Hybrid algorithms with multiple stages are also available, like Müller [11] or Shaker et al. [12], both of which are modifications of the Great Deluge.

3. A TWO-STAGE HEURISTIC

In this section, we propose a two-stage heuristic for solving the curriculum-based university course timetabling problem. First, an initial feasible solution is created using a greedy recursive algorithm, then a local search heuristic is utilized to improve its quality.

We apply the following soft constraints from Section 2: Compactness, Room capacity, Room stability, Minimum number of days. The reason for this is that we use the datasets of the Curriculum-based Course Timetabling track of the International Timetabling Competition 2007 (ITC) [6] as evaluation for the method, and this competition also considers the same constraints.

The initial solution is created using a recursive search, which only considers the hard constraints of the problem. The pseudo-code of this can be seen in Algorithm 1.

Algorithm 1 Recursive algorithm for initial solution.

Funct recSol($course, node, list$)

```
1: if All courses are assigned then
2:     return TRUE
3: end if
4: if No more assignment possibilities then
5:     return FALSE
6: end if
7: if (course, node) assignment is invalid then
8:     node := next (timeslot,room) pair
9:     return recSol(course, node, list)
10: end if
11: if course.teacher is available az node.timeslot then
12:     list ← (course,node) assignment
13:     node := next (timeslot,room) pair
14:     course := next free course
15:     return recSol(course, node, list)
16: else
17:     node := next (timeslot,room) pair
18:     return recSol(course, node, list)
19: end if
```

The function requires 3 input data: the course to be considered ($course$), the proposed time-slot and room pair ($node$), and a list of nodes corresponding to the partial solution that is already built ($list$). Initially, $list$ is empty, and $course$ and

node are randomly chosen. If the assignment of the current course is not possible to this node, then another one is chosen and a recursive call is made. If the course is compatible with the node, and the teacher of the course is also available at the time-slot in question, then the assignment is saved, and the next recursive call will feature a new course. The algorithm terminates if all the courses are assigned, or if there are no more possible nodes to choose from.

As the initial solution is built without considering any soft constraints, it will have a high penalty. A local search method is then used to decrease this penalty by also taking the soft constraints of the problem into consideration. The outline of the algorithm can be seen in Algorithm 2.

Algorithm 2 Local search algorithm.

```
Funct localSearch(tt, tabu, n)
 1: i := 0
 2: bestSol := tt
 3: while n > 0 do
 4:     while foundBetter = TRUE do
 5:         bestCost = cost(tt)
 6:         for Each a := (course,timeslot,room) in tt do
 7:             for Each b := (timeslot,room) in tt do
 8:                 if (a,b) ∈ tabu then
 9:                     CONTINUE
10:                 end if
11:                 neighbor := tt.swap(a,b)
12:                 if neighbor violates hard constraints then
13:                     CONTINUE
14:                 end if
15:                 if cost(neighbor) < bestNeighCost then
16:                     bestNeighCost := cost(neighbor)
17:                     bestNeigh := neighbor
18:                     tabutCand := (b,a)
19:                 end if
20:             end for
21:         end for
22:         if bestNeighCost < bestCost then
23:             bestCost := cost(bestNeigh)
24:             bestSol := bestNeigh
25:             tabu ← tabuCand
26:             foundBetter := TRUE
27:         else
28:             foundBetter := FALSE
29:         end if
30:         if i > x then
31:             tabu(0).erase()
32:             i := 0
33:         end if
34:         i := i+1
35:     end while
36:     destructSearch(bestSol, tabu)
37:     n := n-1
38: end while
39: return bestSol
```

The input of the algorithm is the *tt* timetable of assignments from the first stage, the empty list *tabu* for forbidden neighborhood transformations, and a parameter n that gives the number of destruction steps. The algorithm considers two different neighborhood transformations:

- *Swap*: The timeslot of a (*course*, *timeslot*,*room*) assignment is swapped with the slot of another assignment.
- *Move*: A (*course*,*timeslot*,*room*) assignment is moved to another (*timeslot*,*room*) position.

The process examines all possible neighborhood moves for a given timetable, and applies the one with the smallest penalty. If the resulting timetable is better than the currently stored best one, then it becomes the new best solution, and the neighborhood move that was used to produce this solution is saved to the *tabu* list of forbidden moves.

If a local optimum is found, the algorithm switches to the *destructSearch* phase. In this phase, a set number of neighborhood moves are applied to the solution, strictly decreasing the quality with each step. After this destruction phase, the local search part of the algorithm is executed again, searching for a new local optimum, while also using the existing *tabu* list to avoid certain transformations. The number of these destruction steps is given by the parameter n in the input.

4. TEST RESULTS

As it was mentioned in the previous Section, the algorithm was tested on the instances of the 2007 International Timetabling Competition. These instances are based on real-life input from the University of Udine. The competition provided three sets of input: Early, Late and Hidden datasets. Due to space limitations, we will only present the first two of these datasets. The most important information about these can be seen in Table 1.

Table 1: Input characteristics

Inst.	Day	Slot	Room	Course	Curr.	Unav.
Early Datasets						
Comp01	5	6	6	30	14	53
Comp02	5	5	16	82	70	513
Comp03	5	5	16	72	68	382
Comp04	5	5	18	79	57	396
Comp05	6	6	9	54	139	771
Comp06	5	5	18	108	70	632
Comp07	5	5	20	131	77	667
Late Datasets						
Comp08	5	5	18	86	61	478
Comp09	5	5	18	76	75	405
Comp10	5	5	18	115	67	694
Comp11	5	9	5	30	13	94
Comp12	6	6	11	88	150	1368
Comp13	5	5	19	82	66	468
Comp14	5	5	17	85	60	486

For each instance, the table gives the number of days, timeslots per day, rooms, courses, curricula and unavailability constraints for the teachers. The following penalty values were used for the soft constraints:

- *Compactness*: Every isolated course is worth 2 ponts.

- *Roomy capacity*: The capacity of a room can be violated, but every student above the capacity is worth 1 point.
- *Room stability*: If lectures of a course are scheduled into more than one room, then the penalty for each room beyond the first is 1 point.
- *Minimum number of days*: If a course is scheduled on less days than the minimum required number, 5 penalty points are given for missing each day.

Test results of the algorithm are presented in Table 2. The algorithm was executed ten times for each instance, and the rounded average of these solutions is give by the table. The destructive search ran for 30 steps in each iteration.

Table 2: Test results

Instance	Runningtime	Penalty
Early Datasets		
Comp01	32s	51
Comp02	16M26s	328
Comp03	9M16s	314
Comp04	8M25s	348
Comp05	7M32s	423
Comp06	14M54s	503
Comp07	15M46s	592
Late Datasets		
Comp08	10M31s	361
Comp09	12M3s	285
Comp10	16M17s	536
Comp11	43s	37
Comp12	11M4s	525
Comp13	9M42s	331
Comp14	7M32s	434

The table presents the total running time of both stages for each instance, as well as the total penalty of the best achieved solution. It can be seen from the results that while the running times are acceptable, the penalties in some cases are a bit high. This means that there is still room for the improvement of the algorithm.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we examined the curriculum-based university course timetabling problem, and developed a two-stage heuristic algorithm for its solution. This algorithm uses a greedy recursive approach to construct an initial solution, then increases its quality with the help of a local search method. While the local search itself is not fully a tabu search algorithm, it utilizes a tabu list to store forbidden transformations. A destructive step is also executed to escape local optima at the end of each iteration.

The algorithm was tested on the datasets of the 2007 International Timetabling Competition. While feasible solutions were found for all instances, both their running time and quality can be improved. We would like to implement faster approaches for creating the initial solution, as well as implementing a proper tabu search algorithm instead of the current local search.

6. ACKNOWLEDGMENTS

Máté Pintér was supported by "Integrated program for training new generation of scientists in the fields of computer science", no EFOP-3.6.3- VEKOP-16-2017-0002 (supported by the European Union and co-funded by the European Social Fund). Balázs Dávid acknowledges the European Commission for funding the InnoRenew CoE project (Grant Agreement #739574) under the Horizon2020 Widespread-Teaming program and the Republic of Slovenia (Investment funding of the Republic of Slovenia and the European Union of the European regional Development Fund), and is thankful for the support of the National Research, Development and Innovation Office - NKFIH Fund No. SNN-117879.

7. REFERENCES

[1] E. A. Akkoyunlu. A linear algorithm for computing the optimum university timetable*. *The Computer Journal*, 16(4):347–350, 1973.

[2] H. Babaei, J. Karimpour, and A. Hadidi. A survey of approaches for university course timetabling problem. *Computers & Industrial Engineering*, 86:43–59, 2015.

[3] R. Bellio, S. Ceschia, L. D. Gaspero, A. Schaerf, and T. Urli. Feature-based tuning of simulated annealing applied to the curriculum-based course timetabling problem. *ArXiv*, abs/1409.7186, 2014.

[4] A. Bettinelli, V. Cacchiani, R. Roberti, and P. Toth. An overview of curriculum-based course timetabling. *TOP: An Official Journal of the Spanish Society of Statistics and Operations Research*, 23(2):313–349, 2015.

[5] E. Burke, Y. Bykov, J. Newall, and S. Petrović. A time-predefined approach to course timetabling. *Yugoslav Journal of Operations Research*, 13(2):139–151, 2003.

[6] I. T. Competition. International timetabling competition 2007. http://www.cs.qub.ac.uk/itc2007/index.htm, 2007. Accessed: 2019-07-30.

[7] G. Dueck. New optimization heuristics: The great deluge algorithm and the record-to-record travel. *Journal of Computational Physics*, 104(1):86 – 92, 1993.

[8] S. Even, A. Itai, and A. Shamir. On the complexity of time table and multi-commodity flow problems. In *16th Annual Symposium on Foundations of Computer Science*, pages 184–193, 1975.

[9] G. Lach and M. E. Lübbecke. Curriculum based course timetabling: new solutions to udine benchmark instances. *Annals of Operations Research*, 194(1):255–272, 2012.

[10] Z. Lü and J.-K. Hao. Adaptive tabu search for course timetabling. *European Journal of Operational Research*, 200(1):235–244, 2010.

[11] T. Müller. Itc2007 solver description: a hybrid approach. *Annals of Operations Research*, 172(1):429, 2009.

[12] K. Shaker, S. Abdullah, A. Alqudsi, and H. Jalab. Hybridizing meta-heuristics approaches for solving university course timetabling problems. In P. Lingras, M. Wolski, C. Cornelis, S. Mitra, and P. Wasilewski, editors, *Rough Sets and Knowledge Technology*, pages 374–384. Springer Berlin Heidelberg, 2013.

Detection of different shapes and materials by glasses for blind and visually impaired

Urban Košale
Faculty of Electrical Engineering and Computer Science
University of Maribor
Maribor, Slovenia
urban.kosale@student.um.si

Pia Žnidaršič
Faculty of Electrical Engineering and Computer Science
University of Maribor
Maribor, Slovenia
pia.znidarsic@student.um.si

Kristjan Stopar
Faculty of Electrical Engineering and Computer Science
University of Maribor
Maribor, Slovenia
kristjan.stopar@student.um.si

ABSTRACT
Glasses for blind and visually Impaired were built to help blind and visually impaired to navigate around diverse range of obstacles.

In this study, we tested how well the glasses detect different everyday shapes and materials. The results are crucial for further development, because this device has to able to detect wide variety of shapes and materials in order to be safe and reliable for everyday usage.

For this purpose, we set glasses on a stationary stand and pointed them directly into the obstacle centre. Obstacles made of different materials were attached on a moving stand. The results showed that the sensors discriminate the shapes at the distances between 30 and 90 cm from the glasses. At the distance of 60 cm the triangle was successfully discriminated from circle and rectangle, whereas the latter two were not easy to discriminate. The second experiment showed that plexi glass and glass present a substantial detection challenge. On the other hand, aluminium foil, white paper and micro polyester are easily detected.

Keywords
glasses for the blind and visually imapred, detection, sensor, obstacle, material, shape

1. INTRODUCTION
Glasses for Blind and Visually Impaired is a device built to help blind and visually impaired to navigate around diverse range of obstacles. This device consists of two parts. First part is a head mounted sensor assembly whereas the second is a haptic feedback device, worn as a belt. Detection of obstacles is performed by 10 VL53L1X Time of Flight (ToF) sensors [1], whose ranging data is then processed with an on-board ESP-WROOM-32 microcontroller [2] and send via Bluetooth connection to the belt. The belt is equipped with a second ESP-WROOM-32 on-board microcontroller, which interprets the data and presents it to the wearer with 15 vibration motors arranged in a square grid. The glasses are worn on the head, whereas the belt is attached around the stomach area.

The motivation behind this work is the desire for this device to work in as many circumstances as possible. Rapid technological development brings new materials and shapes in our everyday living space. If we add fast lifestyle to the mix, we get the need for a device that can reliably detect different shapes and materials in close to real time.

Performance of the device was already tested on specifically designed polygon with 12 test subjects. The results are written in the next section. Expanding on these results, in this study, we focus on testing and quantifying device's ability to detect different shapes and materials.

2. POLYGON TEST RESULTS
Test took place in the main lobby of the Faculty of Electrical Engineering and Computer Science, Maribor. This area was used because it is big and open so the glasses detected the obstacles only. It also provided room lighting conditions and flat surface. Polygon seen on Figure 1 consisted out of 4 obstacles made out of polystyrene foam. They simulated everyday objects such as tables, poles and doors. Fifth obstacle which simulated hanging obstacle was tested individually. Every participant had two attempts. Test focused on the number of detected obstacles, number of steps and time necessary to finish the walk through the polygon [3].

All participants detected the fifth, hanging obstacle, but only one out of 12 participants detected an obstacle which simulated the table. On average 2.8 ± 0.39 out of 4 obstacles were detected in the first run and 2.92 ± 0.29 out of 4 in

DOI: https://doi.org/10.26493/978-961-7055-82-5.31-34

the second run. Average time required for participants to finish the walk through the polygon was 80 ± 31.5 seconds in the first and 56 ± 18.3 seconds in the second run. On average, participants made 56.2 ± 15.7 steps in the first and 48.3 ± 10.7 in the second run [3].

From increased walking speed on the second attempt we can conclude that participant certainty with the device increased over time. This was also reflected in improved obstacle detection. [3]

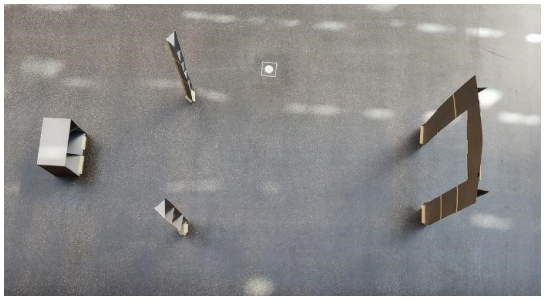


Figure 1: test polygon, set up in the main lobby of the Faculty of Electrical Engineering and Computer Science, Maribor.

3. DETECTION ZONE

Detection of obstacles is implemented using ToF sensors VL53L1X. They provide a maximum range of 4 m and a typical field of view of 27°. These sensors were selected because they are relatively affordable, offer long range detection, are power efficient, support I2C interface and are small enough to enable slim design [1].

Detection zone of the device is constructed by an array of 10 sensors, providing 150° wide and 50° tall coverage as seen on Figure 3. Sensors are divided into three groups. First group consists of two sensors which are oriented straight in the direction of view. One sensor is oriented horizontally and the other is oriented 30 degrees below the horizon. Second group consists of 6 sensors, 3 on the left side and 3 on the right side. Left group is vertically tilted for 22° to the left, whereas right group is vertically tilted for 22° to the right. The upper two and the lower two sensors in this group are horizontally tilted for 10° away from the central sensor which is oriented straight into direction of view. Third group consists of 2 sensors which are oriented straight into direction of view and vertically tilted for 44° [4].

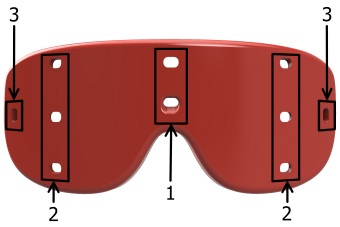


Figure 2: Groups of TOF sensors.

The device was designed in the shape of wearable glasses and 3D printed from Polylactic Acid plastic [4]. This technique was used to achieve special shape which allows sensors to be mounted in a way seen on Figure 3

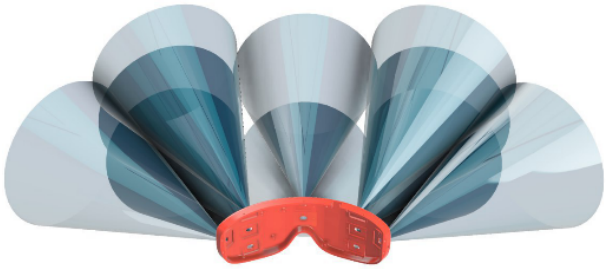


Figure 3: Fields of view for TOF sensors, mounted on the 3D printed glasses.

4. OBSTACLE DETECTION

We designed a special experiment to test how well the glasses detect obstacles of various shapes and materials. Shapes used were circle, triangle and square. Their surface measured 2116 cm^2. Materials used were grey polystyrene foam, white paper, aluminium foil, glass, polyester, plexiglass, ABS, wood, micro polyester and cotton.

Experiment took place in closed environment under dim lighting conditions. Controlled light conditions for this test are important as they effect sensors performance. Device was mounted on a 176 cm high stationary wooden stand which pointed directly into the obstacle centre. Obstacles were placed 30, 60 and 90 cm away from the glasses. Quality of detection was determined with data output consisting of 10 integers, ranging from 0 to 4096. Here, value 0 denotes the minimal and value 4096 the maximal distance. In order to increase the accuracy, every distance was calculated by averaging ten measurements.

Shape discrimination ability was assessed by counting the number of sensors that detected the obstacle. Here, we also considered the distance of obstacle from the glasses.

The initial testing session consisted of recognizing different shapes at the distance of 30 cm (Figure 4). The sensors could not recognize the shape at all. This suggests that the distance is too small for sensors to reach their full potential.

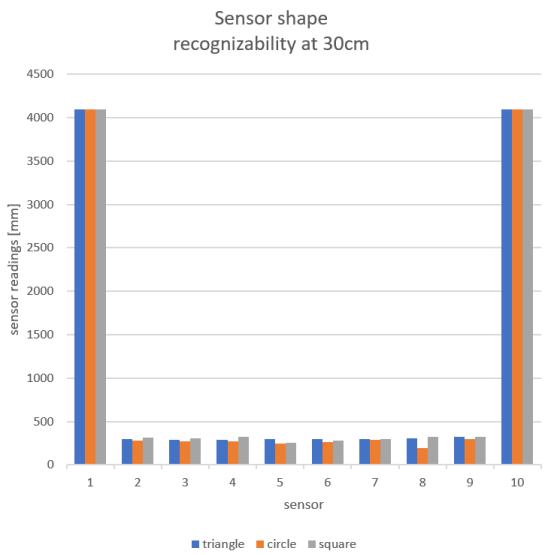


Figure 4: Results of obstacle detection at the distance of 30 cm.

In the next testing session, the obstacles were positioned at the distance of 60 cm from the sensors (Figure 5). This time the circle and the square were not differentiated, but the triangle was. Sensors 6 and 7 have pointed out a difference.

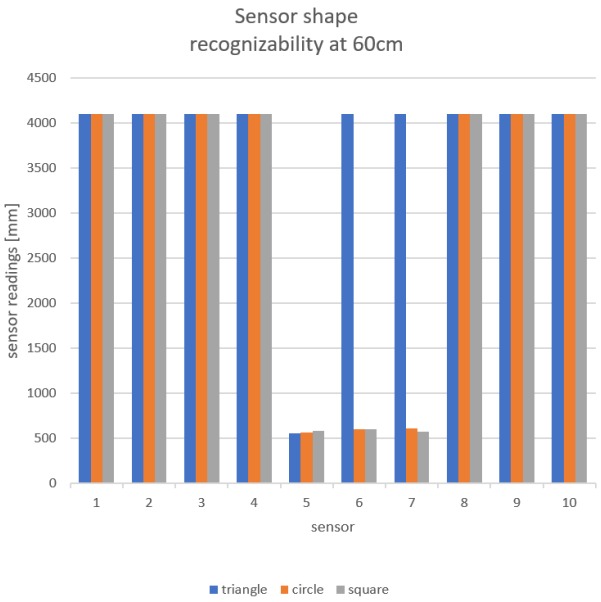


Figure 5: Results of obstacle detection at the distance of 60 cm.

In the third testing session, the obstacles were positioned 90 cm away from the sensors (Figure 6). In this case, the recognizability was again poor. The distance turned out to be too great, since only one (the middle) sensor detected any of the shapes.

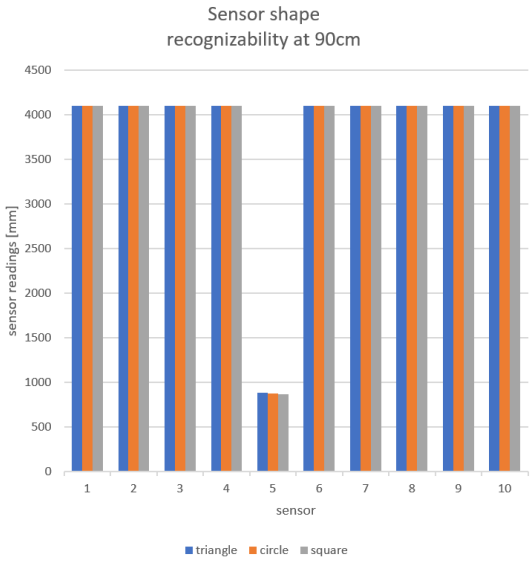


Figure 6: Results of obstacle detection at the distance of 90 cm.

In the last experimental session we tested the detectability of different materials. For this purpose, we put the square obstacle with surface area of 2116 cm^2 encased with the chosen material at the maximum detection distance, which was determined by moving the obstacle away from the glasses until it was no longer detected. The results demonstrated (Figure 7) that the aluminium foil has the best detection potential while ABS has the worst. Glass was also tested, but was excluded from the results in Figure 7, because its detection depends on the sensor angle. Under the specific angle glasses could detect it up to the distance of 1.5 m, but in most cases glass wasn't detected. Plexi glass was also interesting because it was consistently detected, but only by one or two sensors, even on a 30 cm distance.

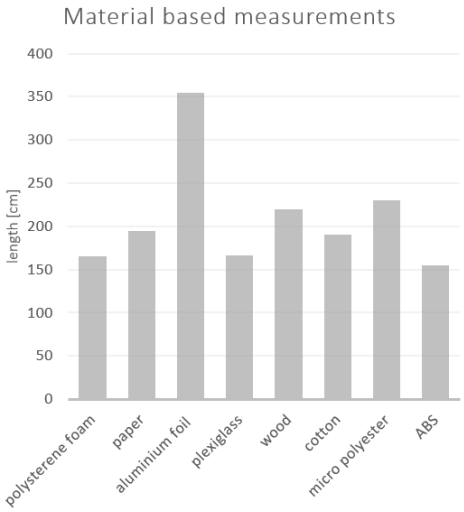


Figure 7: Maximum detection distances for tested materials.

5. CONCLUSIONS

The glasses detected all the obstacles, but the number of sensors detecting the obstacle decreased with the distance. Circle and square were detected better than triangle. This suggests that different shapes trigger different responses of sensors on glasses.

We have also demonstrated that the optimal distance for the sensors to recognize the shapes is somewhere between 30 and 90 cm. At the distance of 60 cm the triangle was successfully discriminated from circle and rectangle, whereas the latter two were not easy to discriminate. The biggest discriminative power of glasses likely lies at the distances between 30 and 60 cm. However, additional tests are required to analyse the performance at different points in this interval (for example at 40 and 50 cm).

The problem with misdetection of more distant obstacles is a consequence of detection principle used in our design. Emitted light cone is 27° wide and it expands with the distance. As a result, the obstacle takes up smaller part of the cone and that affects its detection. There are two solutions of this problem. The first one is to narrow the emitted light cone width by software or hardware. The second one includes adding the video camera for better object recognition.

Our experiments further showed that some of the materials are poorly detected. For example plexi glass and glass present a substantial detection challenge. On the other hand, aluminium foil, white paper and micro polyester are easily detected. In conclusion, the more reflective the material, the more sensors detect it. Further tests are required to analyse whether or not the problem of glass detection could be addressed by the use of video camera. As an alternative, an ultrasonic sensor could also be used.

At the moment our glasses perform best in open environments while detecting materials which are better at reflecting infrared light. The test justifies that the glasses could be used in everyday environments, because materials tested make up most of potential obstacles, glass and other similar materials being the exception.

Plasticity of the brain allows blind and visually impaired people to have significantly increased perception of touch [4]. Because of that, we would like to additionaly test if they can feel the difference between various shapes.

6. ACKNOWLEDGMENTS

We thank prof. dr. Aleš Holobar (University of Maribor) and M.S. Jernej Kranjec (University of Maribor) for their assistance in writing the article and all the professional advises they provided. We also thank Tomaž Ferbežar (Secondary school of electronics and technical gymnasium) and Boris Plut (Secondary school of electronics and technical gymnasium) for their help in the early stages of development.

7. REFERENCES

[1] STMicroelectronics https://www.st.com/resource/en/datasheet/vl53l1x.pdf.

[2] Adafruit HUZZAH32 - ESP32 Feather. https://cdn-learn.adafruit.com/downloads/pdf/adafruit-huzzah32-esp32-feather.pdf. May 2019.

[3] K. Stopar. Naprava za vizualno-kinestetično navigacijoslepih in slabovidnih. September 2019 (submitted for review).

[4] C. M. Bauer, G. V. Hirsch, L. Zajac, B. Koo, O. Collignon, L. B. Merabet Multimodal MR-imaging reveals large-scale structural and functional connectivity changes in profound early blindness PLOSjONE. March 22, 2017.

Comparison of clustering optimization for classification with PSO algorithms

Klemen Berkovič
University of Maribor, Faculty of Electrical Engineering and Computer Science
Koroška cesta 46
Maribor, Slovenia
klemen.berkovic@student.um.si

Uroš Mlakar
University of Maribor, Faculty of Electrical Engineering and Computer Science
Koroška cesta 46
Maribor, Slovenia
uros.mlakar@um.si

Borko Bošković
University of Maribor, Faculty of Electrical Engineering and Computer Science
Koroška cesta 46
Maribor, Slovenia
borko.boskovic@um.si

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

Janez Brest
University of Maribor, Faculty of Electrical Engineering and Computer Science
Koroška cesta 46
Maribor, Slovenia
janez.brest@um.si

ABSTRACT

In this paper, we compare Particle Swarm Optimization (PSO) algorithms for classification based on clustering. Clustering is presented within the proposed PSO algorithms as an optimization problem, and captured in a so called fitness function. Clusters are presented with centers that should represent the classes hidden in data. With the help of PSO algorithms and proper fitness function, we optimize centers of the clusters. Because clustering belongs to a field of unsupervised learning methods, we redefine the clustering problem to supervised learning with a new fitness function that helps PSO algorithms in finding clusters centers that can classify instances of data to the right class. Two experiments are performed in our study. In the former, various fitness functions for classification based on clustering are tested. The latter measures the performance of PSO algorithms using the best fitness function from the first experiment. For testing the PSO algorithms, we use three different datasets. Friedman non-parametric statistical test, and Nemenyi and Wilcoxon post-hoc tests are conducted to properly compare the PSO algorithms.

Categories and Subject Descriptors

I.5.3 [**Pattern recognition**]: Clustering; G.1.6 [**Numerical analysis**]: Optimization

Keywords

clustering optimization, classification, particle swarm optimization, statistical comparison

1. INTRODUCTION

In many fields, including physics, bioinformatics, engineering and economics, we can encounter problems, where from a plethora of solutions we are interested in finding the most suitable. These so called optimization problems (OPs) can be sub-categorized into: discrete, continuous, and/or mixed-variable problems [18]. The difference between three mentioned groups of OPs are the type of variables in their solution space. Discrete problems work with discrete variables that are elements of $\mathbb{N}^+$ set, continuous problems have variables with elements of $\mathbb{R}$ set, while the mixed-variable problems can capture variables of both aforementioned sets.

Since we are usually limited by various computational resources (e.g., computational power and space, problem's constraints), the complexity of OPs are increased and therefore we are forced to be satisfied with a "good enough" solutions for the real-world application. Definition of a "good enough" depends exclusively on the type of a problem. Majority of the OPs have a huge, yet finite solution spaces, which can be solved approximately using meta-heuristics. In our work we are focused on meta-heuristics algorithms that are inspired by nature and are a part of Swarm Intelligence (SI) algorithms [4]. From the field of SI algorithms, we use be using PSO algorithms.

In this work, we are focused on minimization of single-objective OPs that can be defined mathematically, as follows:

$$f\left(\mathbf{x}^*\right) \leq \min_{\mathbf{x}\in\Omega} f\left(\mathbf{x}\right), \tag{1}$$

where $f : \mathbb{R}^D \mapsto \mathbb{R}$ represents the fitness function of the problem to be solved, $\mathbf{x}$ is D dimensional vector consisting of problem variables and $\mathbf{x}^*$ is a global minimum or the best solution of the problem. The fitness function f is realized as a mathematical formula or a set of empirical data that refer to the problem of interest. In order to make the problem simpler, we demand that the solutions should be in some

DOI: https://doi.org/10.26493/978-961-7055-82-5.35-42

feasible space, written mathematically as $\mathbf{x} \in \Omega$. In our case, the feasible space Ω is determined by the upper and lower bounds of their corresponding problem variables.

Classification is a process of labeling data based on some attributes to the right type of information or to the right class. Clustering is a process of searching for centers of groups so called clusters that have very similar data instances. With clustering, we can find cluster centers that can be used for classification. The K-means algorithm is one of the most well known algorithms for clustering [9]. Unfortunately, it has two major defects: (1) the number of clusters must be known in advance, and (2) bad initial centers of clusters can degrades the performance. Because of those defects, many meta-heuristic algorithms have been used for clustering, like Big Bang–Big Crunch Algorithm (BBBCA) [8], Black Hole (BH) [7], Gravitational Search Algorithm (GSA) [6], and many others.

The rest of the paper is organized as follows: In Section 2 PSO algorithms used in our work are presented. Section 3 defines the clustering problem and presents some fitness function suitable for clustering with PSO algorithms. Section 4 shows the results of two experiments: The former based on usage of various fitness functions, while the latter compares different algorithms for classification based on clustering. Finally, in Section 5, we conclude our paper with summarizing the performed work and provide some directions for feature work.

2. PARTICLE SWARM OPTIMIZATION

PSO is an optimization algorithm for solving problems which solutions variables are elements of $\mathbb{R}$. The basic PSO algorithm was presented by Kennedy and Eberhart in 1995 [10]. Individual in PSO algorithms is presented as a particle that has current position, personal best position and velocity. In each generation the PSO algorithm goes over the whole population and for each particle first updates it's velocity based on the following expression:

$$\begin{aligned}\mathbf{v}_i^{(t+1)} = \omega^{(t+1)} \times \mathbf{v}_i^{(t)} \; &+ \; c_1 \times \mathbf{r}_1 \otimes \left(\mathbf{x}_{pb,i}^{(t)} - \mathbf{x}_i^{(t)}\right) \\ &+ \; c_2 \times \mathbf{r}_2 \otimes \left(\mathbf{x}_{gb}^{(t)} - \mathbf{x}_i^{(t)}\right),\end{aligned} \tag{2}$$

where $\otimes$ represents element wise multiplication, $\mathbf{r}_1$ and $\mathbf{r}_2$ have uniform random distributed values with D components $\in [0, 1]$, $\omega^{(t+1)}$ is an inertia weight, $\mathbf{v}_i^{(t)}$ is a vector representing i-th particle velocity at time t and $\mathbf{v}_i^{(t+1)}$ represents new velocity for particle i. Factors c_1 and c_2 represent social and cognitive learning rates respectively. After the velocity updating is done, the updating of particle position takes place as follows:

$$\mathbf{x}_i^{(t+1)} = \mathbf{x}_i^{(t)} + \mathbf{v}_i^{(t+1)} \; . \tag{3}$$

After particle updates its position, the personal best position, and the global best positions are update based on fitness value.

Opposition-base learning and velocity clamping are two fundamental phases of Opposition-Based Particle Swarm Optimization with Velocity Clamping algorithm (OVCPSO) defined in [16]. In the OVCPSO algorithm, opposition-based learning phase [19] is used for calculating the opposite particle $\mathbf{x'}_i$ as follows:

$$\mathbf{x'}_i = \mathbf{x}_{max} + \mathbf{x}_{min} - \mathbf{x}_i. \tag{4}$$

Opposition-based learning phase in OVCPSO consists of calculating opposite particles swarm $\mathbf{X}'$ then creating an union between original and opposite swarm of particle as $\mathbf{X}'' = \mathbf{X} \cup \mathbf{X}'$. And the last step of opposition-based learning phase is to select N best particles based on the fitness values of particles in $\mathbf{X}''$ that will present our new swarm of particles. The OVCPSO algorithm has velocity clamping operator that repairs the velocity of particles if the velocity exceeds some maximum velocity.

One of the the variations of PSO algorithm that we used in our work is Mutated Centered Unified Particle Swam Optimization (MCUPSO) algorithm described in [20]. The MCUPSO algorithm uses two additional operators that are centering and mutation. Centering operator was described in the CPSO algorithm [12] and mutation operator was defined in Mutated Particle Swarm Optimization (MPSO) algorithm [17]. Both operators are executed at the end of the algorithm generation. The MCUPSO algorithm has a new equation for velocity updating defined as:

$$\begin{aligned}\mathbf{v}_i^{(t+1)} = \omega^{(t+1)} \times \mathbf{v}_i^{(t)} \\ + \; c_1 \times \mathbf{r}_1 \otimes \left(\mathbf{x}_{pb,i}^{(t)} - \mathbf{x}_i^{(t)}\right) \otimes \mathbf{r}_3 \\ + \; c_2 \times \mathbf{r}_2 \otimes \left(\mathbf{x}_{gb}^{(t)} - \mathbf{x}_i^{(t)}\right) \otimes (1 - \mathbf{r}_3),\end{aligned} \tag{5}$$

where $\mathbf{r}_3$ is a D component vector with uniform random numbers $\in [0, 1]$.

Because of the bad performance of the PSO algorithm on multi-modal OPs, authors in [11] developed a PSO algorithm called Comprehensive Learning Particle Swarm Optimizer (CLPSO) algorithm which overcomes this problem. The CLPSO algorithm uses the basic PSO algorithm with additional comprehensive learning phase, where comprehensive learning phase uses all personal best positions for updating particles velocity. Particle velocity is update based on:

$$\mathbf{v}_i^{(t+1)} = \omega^{(t+1)} \times \mathbf{v}_i^{(t)} + c \times \mathbf{r} \otimes \left(\mathbf{z} - \mathbf{x}_i^{(t)}\right), \tag{6}$$

where c is user defined value, $\mathbf{r}$ is randomly uniform distributed vector with number of components equal to number of components in the solution and $\mathbf{z}$ is vector that is composed from different personal best positions.

In our implementation, we added solution repairing operator that is executed after the position of particle in updated. Solution repairing is done based on mirroring back the components of the solution that are out of the allowed search space back into the search space. Mirroring of bad components is based on modulo operation.

3. CLUSTERING OPTIMIZATION

Clustering is one of the most important data analysis techniques that involves analysis of multivariate data. It is a method of unsupervised machine learning for pattern recognition, data mining, supervised machine learning, image analysis, bioinformatics, prediction, etc. Result of clustering optimization is centers of clusters, where instances of one cluster have maximum similarity between each other and

minimum similarity from instances of the other groups [13]. In Figure 1, a simple clustering problem is presented, where we can simply find the centers of the clusters. In Figure 2,

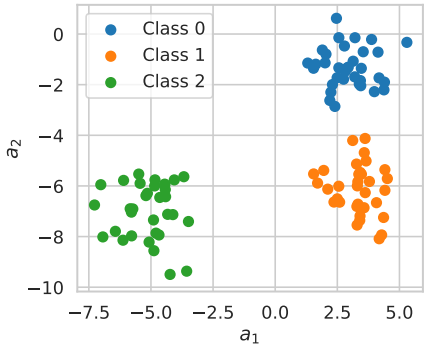


Figure 1: Example of simple clustering dataset

the harder clustering problem is shown, where the centers for clusters representing Class 0 and Class 2 are not so trivial to find. Clustering problem defined in Figure 2 is quite

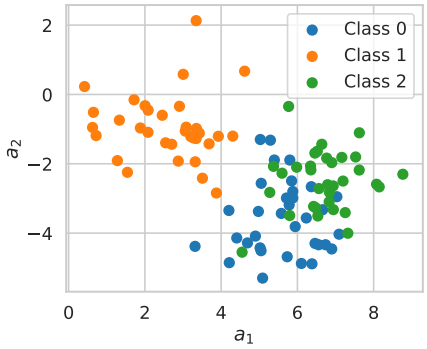


Figure 2: Example of hard clustering dataset

hard to solve with K-means algorithm because mean values of attributes a_1 and a_2 are very close for both classes. Even instances of class 0 and class 2 overlap for both attributes. Therefore we use optimization to solve the problem of clustering for all three classes represented in Figure 2. For clustering optimization we will need a fitness function that will guide our PSO algorithms toward optimal solutions in the search space. In this study, we propose the following fitness functions for clustering:

- basic clustering function,
- K-means clusters evaluation function,
- K-means clusters evaluation function with penalty function 1, and
- K-means clusters evaluation function with penalty function 2.

In the remainder of this section, the proposed fitness functions for clustering optimization are described in more detail.

If we have a dataset as presented in Figure 1, we can use the basic clustering function, expressed as:

$$f(\mathbf{O}, \mathbf{Z}) = \sum_{j=0}^{K-1} \sum_{i=0}^{N-1} w_{i,j} \times \|\mathbf{o}_i - \mathbf{z}_j\|_2 , \tag{7}$$

where $\mathbf{O}$ is a dataset without data labels, $\mathbf{Z}$ represents centers of clusters, $\mathbf{W}$ is a matrix with weights for instances for each cluster, N is a number of instances in data set $\mathbf{O}$, K represents a number of clusters in $\mathbf{Z}$ and $\|\mathbf{o}_i - \mathbf{z}_j\|_2$ is the Euclidean distance between instance $\mathbf{o}_i$ and cluster center $\mathbf{z}_j$. In paper [7] authors added weights $\mathbf{W}$ which are in basic clustering function set to one. Through weights, we can provide additional information to the PSO algorithms. One example of weights usage can be with setting the weight $w_{i,j}$ to one if the instance i belong to cluster j else the weight is set to zero. Second example of weights usage can be for fuzzy clustering where weight $w_{i,j}$ has the value $\in [0, 1]$.

The second fitness function for clustering optimization is defined, as follows:

$$f(\mathbf{z}, \mathbf{O}) = p(\mathbf{z}) + \sum_{i=0}^{N-1} \min_{j=0}^{K-1} \left(w_{i,j} \times \|\mathbf{o}_i - \mathbf{z}_j\|_2 \right), \tag{8}$$

where p denotes a penalty function [1]. Optimization function from Eq. (8) is used for evaluating clusters in the K-means algorithm [9], where penalty function is not used.

The third fitness function for clustering optimization takes into account intra-cluster distances, in other words:

$$p(\mathbf{z}) = \sum_{\forall \mathbf{e} \in \mathfrak{I}} \sum_{j=0}^{A-1} \min \left(\frac{r_j}{K}, \max \left(0, \frac{r_j}{K} - |z_{e_0,j} - z_{e_1,j}| \right) \right), \tag{9}$$

where r_j is calculated with $|u_j - l_j|$ and presents the intra-cluster distance between j attributes upper u_j and lower l_j limit, A is the number of attributes in one instance and $\mathfrak{I}$ is a set containing vectors with two components, that represents indexes of clusters. Set $\mathfrak{I}$ is generated with the help of Alg. 1 which gives pairs of cluster indexes that need to be checked for intra-cluster distances. The penalty function 1

Algorithm 1: Creating set $\mathfrak{I}$ for calculating penalty of individual $\mathbf{Z}$ based on cluster distances

Input: K
Output: $\mathfrak{I}$

```
ℑ ← {};
for i ← 0 to K − 1 do
    for j ← 1 to K − i − 1 do
        if i ≠ K − j − 1 then
            ℑ ← {i, K − j − 1};
        end
    end
end
return ℑ;
```

adds penalty based on intra-clusters distances, which means if the intra-clusters distance is to small then based on that distance the solution gets its penalty. The maximum penalty

that a solution can get is equal to:

$$\sum_{i=0}^{A-1} \frac{u_j - l_j}{K}, \tag{10}$$

where the search space is divided into K partitions and every cluster should be in partition occupied only by one cluster.

In our paper, we introduce the penalty function 2 for clustering, defined as:

$$p\left(\mathbf{Z}, \mathbf{O}, \mathbf{c}\right) = \left(1 - a\left(\mathbf{Z}, \mathbf{O}, \mathbf{c}\right)\right) \times \|\mathbf{u} - \mathbf{l}\|_2 \times 2, \tag{11}$$

where $\mathbf{u}$ represents upper limits of the search space, $\mathbf{l}$ represents lower limits of the search space, $\mathbf{c}$ represents a vector of class or class labels of an instances in dataset $\mathbf{O}$, and a is a function for calculating the accuracy of a given centers $\mathbf{Z}$. In Eq. (11) we added multiplication factor of two, because we want to give this penalty a higher weight. Eq. (11) redefines the problem from unsupervised to supervised learning method, because we added the information of how the instance is classified to the learning method.

4. EXPERIMENTS AND RESULTS

In our work, we performed two separate experiments. The first experiment, compares the fitness functions for classification based on clustering. The second experiment, presents statistical comparison of used PSO algorithms for classification based on clustering with the best fitness function from our first experiment. Python code of implemented algorithms can be accessed at GitHub [1] and Python code of experiments can be accessed at GitHub [2].

In our experiments we had to adjust the representation of the individual so that all operators of PSO algorithms could work properly. Figure 3 shows an example of an individual in a population, where $\mathbf{z}_1, \cdots, \mathbf{z}_k$ represents clusters. Centers have K components which represent center points for each attribute of the dataset. Figure 3 shows attributes centers in form of symbols $z_{1,1}, \cdots, z_{1,n}$ for cluster one and $z_{k,1}, \cdots, z_{k,n}$ for cluster k. For all PSO algorithms used in

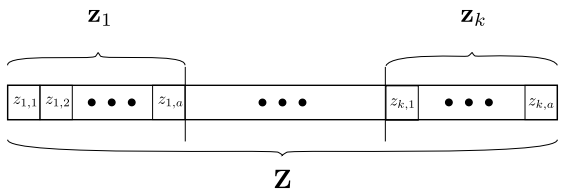


Figure 3: Example of solution coding

our work, we used $1,000$ solution evaluations as an stopping criteria.

In our experiments, we used the Friedman non-parametric statistical test [5], where the hypothesis H_0 asserts that there is no significant statistical differences between algorithms. If the Friedman test rejected H_0, then we used the Nemenyi post-hoc statistical test [14] for calculating critical distances determining, that the results of two algorithms are significantly different, when their critical distance intervals do not overlap. To be really confident that the difference between the results is significant, we ran a Wilcoxon 2-paired signed-rank test [22]. We present the result of Wilcoxon test in tables, where character "+" denotes that the results of two algorithms are statistically significantly different, while the character "∼" shows that the difference between the results is not statistically significant. For better accuracy of statistical testing, we used 51 runs. To get a better accuracy of Friedman tests, we used more then five classifiers in our second experiment as a rule described in [2].

For comparing the results of our two experiments we used error rate calculated as:

$$e\left(\mathbf{Z}, \mathbf{O}, \mathbf{c}\right) = 1 - \frac{a\left(\mathbf{Z}, \mathbf{O}, \mathbf{c}\right)}{N}, \tag{12}$$

where $\mathbf{O}$ is a dataset, $\mathbf{Z}$ represents centers of clusters, N represents number of instances in a dataset $\mathbf{O}$, $\mathbf{c}$ hold the labels that represent classes for each instance in a dataset $\mathbf{O}$ and a is the function that returns the number of correctly classified instances of the dataset $\mathbf{O}$ based on centers $\mathbf{Z}$. Before the experiment runs we divided the dataset into training and validation sets, where the validation set contained 30% of samples of the original dataset. The error rates that we compared where taken from the validation set after the optimization process finished. Because of using the error rate for statistical analysis with a combination of the Friedman test ranks, we can say that the best algorithm has the smallest mean rank.

4.1 Fitness function comparison

The first experiment that we conducted was for testing performance of the fitness functions used for classification. In this test, we used four different fitness functions described in Section 3 with the standard PSO algorithm [10] and the K-means algorithm. The K-means algorithm was initialized with random centers of clusters. For the experiment we generated a random dataset with four different classes and 500 samples that had nine components or so called attributes. For dataset generation we used scikt-learn's [15] function `make_blobs`. In the generated dataset each class had 125 instances. Figure 4 is showing significant differences between

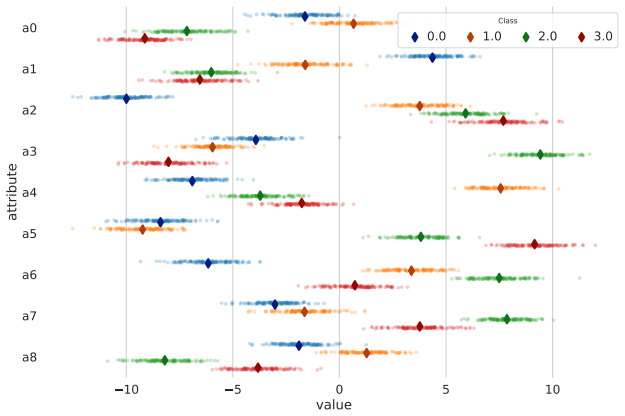


Figure 4: Samples grouped by class with mean values for each attribute in generated dataset

[1] `https://github.com/kb2623/NiaPy`
[2] `https://github.com/kb2623/NiaPy-examples/tree/master/clustering_datasets`

intervals of data values for each class. In some cases mean values are quite far away form each other. So based on this information, the generated dataset should not be a hard problem for clustering.

Table 1: Time of execution for one function evaluation based on selected optimization function

Label	Clustering fitness function	Mean execution time	Time complexity (worst)
C	Eq. (7)	11.8 µs ± 73.3 ns	$O(NKA)$
CM	Eq. (8) with penalty set to 0	12 µs ± 56.3 ns	$O\left(KA\left(N+1\right)\right)$
CMP	Eq. (8) with penalty based on Eq. (9)	46.3 µs ± 270 ns	$O\left(KA\left(N+\frac{K-1}{2}\right)\right)$
CC	Eq. (8) with penalty based on Eq. (11)	76.9 ms ± 510 µs	$O\left(NA\left(2K+1\right)\right)$

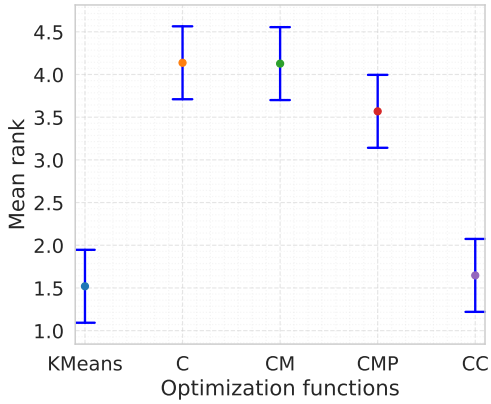


Figure 5: The Friedman test mean ranks with the Nemenyi post-hoc test with $q_\alpha = 0.05$ for optimization functions and K-means algorithm

Table 1 shows a time complexity for each optimization function used in our first experiment. The experiment was ran on computer with Intel's i5-4570 processor on one thread with 16 GB of main memory. It can be seen from the results, that the last fitness functions labeled CC is the best function for classification based on clustering. This fitness function has the highest time complexity and consequently the longest execution time, but gives the best results compared to other fitness functions used as seen on Figure 5. Friedman test gave us the statistic value of 151.82682 and p-value of 8.264 69e−32, so for q_α of 0.05 we can say that K-means and PSO algorithm with different fitness functions work significantly different on generated dataset. In Figure 5, we can observe that the Nemenyi post-hoc test detected some statistical insignificant differences between two groups of used methods. First group with C, CM and CMP and second group with K-means and CC. From Table 2, we can observe that second group has statistically significant difference, but for the first group only methods C and CM do not have statistically significant differences.

4.2 Comparison of PSO algorithms

In our second experiment we measured the performance of six different PSO algorithms and the basic K-means algorithm labeled as KM. For the fitness function of PSO algorithms we used Eq. (8) with penalty function described in

Table 2: The Wilcoxon test showing statistical differences for $q_\alpha = 0.05$ for all clustering methods used

	KMeans	C	CM	CMP	CC
KMeans	∞	+	+	+	+
C	/	∞	∼	+	+
CM	/	/	∞	+	+
CMP	/	/	/	∞	+
CC	/	/	/	/	∞

Eq. (11), since it showed the best performance in our first experiment.

We measured the performance of algorithms used on three different datasets, which are:

- **Iris**: Dataset has 150 instances with four attributes where instances are labeled with three different classes. All attributes in the dataset are elements of $\mathbb{R}^+$. Each class in the dataset has 50 instances.
- **Breast Cancer Wisconsin (BCW)**: Dataset has 569 instances distributed into two classes and each instance has 30 attributes. All attributes in the dataset are elements of $\mathbb{R}^+$ numbers set. The first class contains 212 instances, while the second class contains 357 instances.
- **Wine**: One of the hardest dataset used in our second experiment is the Wine dataset. The dataset has 178 instances with 13 attributes. All attributes except one are elements of $\mathbb{R}^+$ set. Only attribute Proline is an element of $\mathbb{N}^+$ set. The dataset has three classes. The dataset contains 59 instances which belong to the first class, 71 instances of the second and 48 instances of the third class.

All datasets where obtained form the Machine learning repository [3]. The performance was measured based on the error rates from 51 runs for each dataset.

Table 3: PSO algorithms parameters

	PSO	CLPSO	OVCPSO	MPSO	CPSO	MCUPSO
NP	25	25	25	25	25	25
c_1	2.0	2.0	2.0	2.0	2.0	2.0
c_2	2.0	2.0	2.0	2.0	2.0	2.0
ω	/	0.7	/	0.7	0.7	0.7
$\mathbf{v}_{min}$	/	-1.5	/	/	/	/
$\mathbf{v}_{max}$	/	1.5	/	/	/	/
m	/	10	/	/	/	/
ω_0	/	0.9	/	/	/	/
ω_1	/	0.4	/	/	/	/
c	/	1.49445	/	/	/	/
p_0	/	/	0.3	/	/	/
w_{min}	/	/	0.4	/	/	/
w_{max}	/	/	0.9	/	/	/
δ	/	/	0.1	/	/	/
μ	/	/	/	10	/	10

Table 3 shows all the parameter values used by PSO algorithms. Some parameters are used for all PSO algorithms, while some are algorithm specific. If a parameter is not used for a specific algorithm the symbol "/" is used.

From Table 4, we can see that the best classifiers are K-means and original PSO algorithm, because they got the best minimum error rates. The CLPSO algorithm has the best mean value and the MPSO algorithm has the smallest standard deviation. One major setback of the OVCPSO algorithm was its worst found centers for clusters, that's classification accuracy was only 29%. Friedman test gave the

Table 4: Basic statistics of used algorithms for 51 runs on Iris dataset

	KM	PSO	CLPSO	OVCPSO	MPSO	CPSO	MCUPSO
mean	0.0588	0.0553	**0.0544**	0.0644	0.0505	0.0579	0.0562
std	0.0376	0.0305	0.0356	0.0954	**0.0274**	0.0424	0.0380
min	**0.0000**	**0.0000**	0.0222	0.0222	0.0222	0.0222	0.0222
median	0.0444	0.0444	0.0444	0.0444	0.0444	0.0444	0.0444
max	0.1777	**0.1333**	0.2000	0.7111	0.1555	0.2444	0.1777

statistic value of 104.81 with p-value of 2.48215e−20. For q_α value of 0.05 we can say based on the Friedman test that used algorithms work significantly different. Mean ranks of algorithms from Figure 6 fit the results in the Table 4. Figure 6 shows that the MCUPSO algorithm holds the smallest rank, despite not having the best accuracy. We can see that

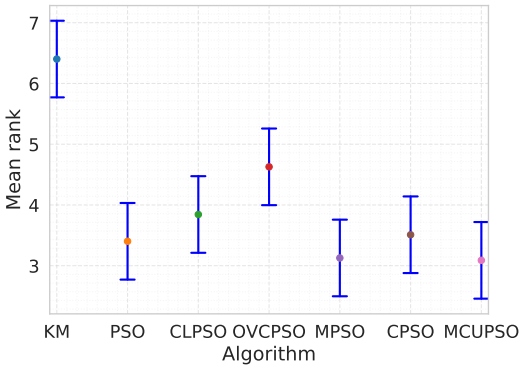


Figure 6: Friedman mean ranks and Nemenyi post-hoc test with $q_\alpha = 0.05$ for PSO algorithms and the K-means algorithm on Iris dataset

Nemenyi test implies that MCUPSO is not significantly different from CPSO, MPSO, CLPSO and PSO algorithms. Based on results in Table 5 we can reject the the hypothesis of Nemenyi test. The Wilcoxon test detected insignificant difference only between MCUPSO, CPSO, MPSO and PSO algorithms. Form the Wilcoxon test we can observe that the MCUPSO algorithm is significantly different compared to OVCPSO, CLPSO and K-means algorithms. MCUPSO, CPSO and MPSO algorithms performed the best based on Friedman mean rank and the Wilcoxon post-hoc test.

From the Table 6, we can observe that the best accuracy obtained the CLPSO algorithm, the best mean value obtained the OVCPSO algorithm, while the PSO algorithm recorded the smallest standard deviation. The MCUPSO algorithm

Table 5: Detected significant statistical differences with the Wilcoxon test with $q_\alpha = 0.05$ on Iris dataset

	KM	PSO	CLPSO	OVCPSO	MPSO	CPSO	MCUPSO
KM	∞	+	+	+	+	+	+
PSO	/	∞	~	+	~	~	~
CLPSO	/	/	∞	~	+	~	+
OVCPSO	/	/	/	∞	+	+	+
MPSO	/	/	/	/	∞	~	~
CPSO	/	/	/	/	/	∞	~
MCUPSO	/	/	/	/	/	/	∞

obtained good results, it had the best median value and the best accuracy in the worst run in all of the 51 runs. Friedman test gave the statistics value of 174.63646 with p-value of 4.66982e−35. For q_α value of 0.05 we can say that the Friedman test rejected the H_0 hypothesis. We can see from the Figure 7, that the MCUPSO algorithm obtained the smallest rank, because of the average standard deviation and smallest error rate on the worst run of the algorithm. On the Figure 7, we can see that MCUPSO,

Table 6: Basic statistics of used algorithms for 51 runs on BCW dataset

	KM	PSO	CLPSO	OVCPSO	MPSO	CPSO	MCUPSO
mean	0.1606	0.1552	0.1643	**0.1479**	0.1598	0.1672	0.1536
std	0.0768	**0.0686**	0.0779	0.0697	0.0728	0.0892	0.0723
min	0.0818	0.0818	**0.0760**	0.0818	0.0818	0.0877	0.0877
median	0.1345	0.1345	0.1403	0.1345	0.1345	0.1345	**0.1169**
max	0.3567	0.3684	0.3567	0.3567	0.3567	0.3684	**0.3333**

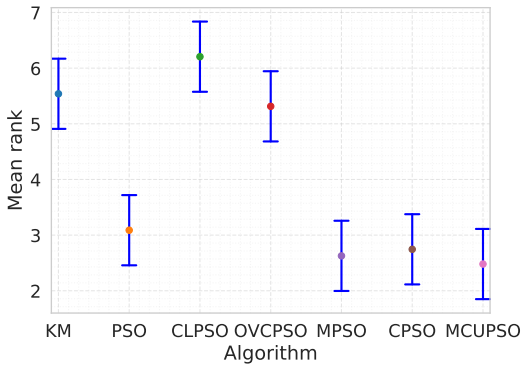


Figure 7: Friedman mean ranks and the Nemenyi post-hoc test with $q_\alpha = 0.05$ for PSO algorithms and the K-means algorithm on BCW dataset

MPSO, CPSO and PSO algorithms do not show significant differences based on the Nemenyi test, but the Wilcoxon test found insignificant differences only between MCUPSO, CPSO and MPSO. If we check the PSO algorithm compared to MCUPSO, MPSO and CPSO algorithms, then we can observe that the Wilcoxon test detected significant differences only between PSO and MCUPSO algorithms. MPSO, CPSO, MCUPSO algorithms based on Friedman mean rank and the Wilcoxon post-hoc test work the best for this dataset.

Table 7: Detected significant statistical differences with the Wilcoxon test with $q_\alpha = 0.05$ on BCW dataset

	KM	PSO	CLPSO	OVCPSO	MPSO	CPSO	MCUPSO
KM	∞	+	+	+	+	+	+
PSO	/	∞	+	+	∼	∼	+
CLPSO	/	/	∞	+	+	+	+
OVCPSO	/	/	/	∞	+	+	+
MPSO	/	/	/	/	∞	∼	∼
CPSO	/	/	/	/	/	∞	∼
MCUPSO	/	/	/	/	/	/	∞

In Table 8, we can see that two algorithms, namely MPSO and CPSO algorithms, obtained the best cluster centers for classification on the Wine dataset. We can see that the MPSO algorithm got the best and the worst clusters centers, and the best median and standard deviation values on this dataset. Table 8 is showing that not only the MPSO algorithm but CPSO and CLPSO algorithms got the best clusters centers for classification. Best median values where obtained with MPSO, OVCPSO and CLPSO algorithms. The Friedman test gave us the statistic value of 58.06224

Table 8: Basic statistics of used algorithms for 51 runs on Wine dataset

	KM	PSO	CLPSO	OVCPSO	MPSO	CPSO	MCUPSO
mean	0.3151	0.3111	0.2977	**0.2970**	0.2930	0.3122	0.3162
std	0.0490	0.0515	0.0519	0.0495	**0.0457**	0.0594	0.0510
min	0.2222	0.2037	**0.1851**	0.2037	**0.1851**	**0.1851**	0.2407
median	0.3148	0.3148	**0.2962**	**0.2962**	**0.2962**	0.3148	0.3148
max	0.4814	0.4259	0.4259	0.4074	**0.3703**	0.4444	0.4629

with p-value of 1.1131e−10. For q_α of 0.05 we can say that the Friedman test detected significant differences between used algorithms. From the results seen in Table 8 we would suspect that the MPSO algorithm would have the smallest mean rank, but the results from Figure 8 show that the CPSO algorithm has the smallest mean rank. As we can see from Figure 8 there is no significant differences between PSO, MPSO, CPSO and MCUPSO algorithms, which can be confirmed with results collated in Table 9. For the basic PSO algorithm the Wilcoxon test detected only two significant differences. As seen from Figure 8 the CPSO algorithm has the smallest mean rank, but we can not say that

Table 9: Detected significant statistical differences with the Wilcoxon test with $q_\alpha = 0.05$ on Wine dataset

	KM	PSO	CLPSO	OVCPSO	MPSO	CPSO	MCUPSO
KM	∞	+	∼	+	+	+	+
PSO	/	∞	+	∼	∼	∼	∼
CLPSO	/	/	∞	∼	+	+	+
OVCPSO	/	/	/	∞	+	+	+
MPSO	/	/	/	/	∞	∼	∼
CPSO	/	/	/	/	/	∞	∼
MCUPSO	/	/	/	/	/	/	∞

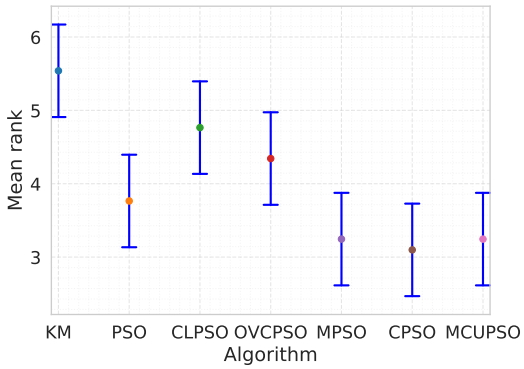


Figure 8: Friedman mean ranks and the Nemenyi post-hoc test with $q_\alpha = 0.05$ for PSO algorithms an the K-means algorithm on Wine dataset

this algorithm is the winner for this dataset. Because of insignificant differences between CPSO, MPSO and MCUPSO algorithms we can say that this three algorithms work best for for this dataset.

5. CONCLUSIONS

In our work, we used clustering optimization for classification. We proposed a new fitness function that has two components. The first component is a clustering function that is used in the K-means algorithm for clusters evaluation and the second component is a penalty function, which is the basis for supervised learning. Our proposed fitness function is a weighted sum of this two components. First component has weight equal to 0.25 and second component has weight equal to 0.75. As it turns out, on used datasets, this fitness function works well. In our work we tried to eliminate initial clusters defect of the K-means algorithm, which makes the K-means algorithm converge fast to some local optimum.

One of the options for feature work is to use more function evaluations with an additional archive, because as we have seen in our experiments, good solutions can be found only after 1,000 function evaluations. Because evolutionary computation is a large research area, we would look for other optimization algorithm. In our work, we did not found the solution for detecting the number of clusters in dataset. This is a challenging task for currently known algorithms and would be a good option for feature work. A multi-objective optimization algorithm would be a good stating point for detecting a number of optimal clusters hidden in data.

6. ACKNOWLEDGMENTS

Authors would like to thank the contributors of nature inspired framework for Python, called NiaPy [21], for they efforts in the development of the framework. The authors acknowledge the financial support from the Slovenian Research Agency (research core funding No. P2-0041).

7. REFERENCES

[1] A. Chehouri, R. Younes, J. Perron, and A. Ilinca. A constraint-handling technique for genetic algorithms

using a violation factor. *arXiv preprint arXiv:1610.00976*, 2016.

[2] J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, 7(Jan):1–30, 2006.

[3] D. Dua and C. Graff. UCI machine learning repository, 2017.

[4] R. C. Eberhart, Y. Shi, and J. Kennedy. *Swarm intelligence*. Elsevier, 2001.

[5] M. Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32(200):675–701, 1937.

[6] X. Han, L. Quan, X. Xiong, M. Almeter, J. Xiang, and Y. Lan. A novel data clustering algorithm based on modified gravitational search algorithm. *Engineering Applications of Artificial Intelligence*, 61:1 – 7, 2017.

[7] A. Hatamlou. Black hole: A new heuristic optimization approach for data clustering. *Information Sciences*, 222:175 – 184, 2013. Including Special Section on New Trends in Ambient Intelligence and Bio-inspired Systems.

[8] A. Hatamlou, S. Abdullah, and M. Hatamlou. Data clustering using big bang–big crunch algorithm. In *International Conference on Innovative Computing Technology*, pages 383–388. Springer, 2011.

[9] A. K. Jain. Data clustering: 50 years beyond k-means. *"Pattern Recognition Letters"*, 31(8):651 – 666, 2010. Award winning papers from the 19th International Conference on Pattern Recognition (ICPR).

[10] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of IEEE International Conference on Neural Networks*, volume 4, pages 1942 – 1948 vol.4, 12 1995.

[11] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. *IEEE Transactions on Evolutionary Computation*, 10(3):281–295, June 2006.

[12] Y. Liu, Z. Qin, Z. Shi, and J. Lu. Center particle swarm optimization. *Neurocomputing*, 70(4-6):672–679, 2007.

[13] S. J. Nanda and G. Panda. A survey on nature inspired metaheuristic algorithms for partitional clustering. *Swarm and Evolutionary Computation*, 16:1 – 18, 2014.

[14] P. Nemenyi. Distribution-free multiple comparisons (doctoral dissertation, princeton university, 1963). *Dissertation Abstracts International*, 25(2):1233, 1963.

[15] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[16] F. Shahzad, A. R. Baig, S. Masood, M. Kamran, and N. Naveed. Opposition-based particle swarm optimization with velocity clamping (ovcpso). In *Advances in Computational Intelligence*, pages 339–348. Springer, 2009.

[17] Y. Shi and R. Eberhart. A modified particle swarm optimizer. In *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360)*, pages 69–73, May 1998.

[18] K. Socha. *Ant Colony Optimization for Continuous and Mixed-Variable Domains*. PhD dissertation, Universite Libre de Bruxelles, 2014.

[19] H. R. Tizhoosh. Opposition-based learning: a new scheme for machine intelligence. In *International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC'06)*, volume 1, pages 695–701. IEEE, 2005.

[20] H.-C. Tsai. Unified particle swarm delivers high efficiency to particle swarm optimization. *Applied Soft Computing*, 55:371 – 383, 2017.

[21] G. Vrbančič, L. Brezočnik, U. Mlakar, D. Fister, and I. Fister Jr. NiaPy: Python microframework for building nature-inspired algorithms. *Journal of Open Source Software*, 3, 2018.

[22] F. Wilcoxon. Individual comparisons by ranking methods. In *Breakthroughs in statistics*, pages 196–202. Springer, 1992.

Hierarchical Routing Algorithm for Industrial Mobile Robots by Signal Temporal Logic Specifications

[Extended Abstract]

Balázs Csutak
Faculty of Information Technology and Bionics, Pázmány Péter Catholic University
H-1083 Práter u. 50/a, Budapest, Hungary
csutak.balazs@hallgato.ppke.hu

Tamás Péni
Systems and Control Laboratory, Institute for Computer Science and Control
H-1111 Kende u. 13-17., Budapest, Hungary
peni.tamas@sztaki.mta.hu

Gábor Szederkényi
Faculty of Information Technology and Bionics, Pázmány Péter Catholic University
H-1083 Práter u. 50/a, Budapest, Hungary
szederkenyi@itk.ppke.hu

ABSTRACT

A two-level route planning algorithm based on model predictive control (MPC) is proposed in this paper for industrial mobile robots, executing tasks in an environment specified using the methodology of signal temporal logic (STL). STL is applied to describe various conditions like collision-free and deadlock-free operation, followed by the transformation of the formulas into a mixed integer linear programming (MILP) problem, solved using dedicated software. To achieve real-time operation, the route planning is divided into two distinct phases using different underlying vehicle models. The correctness of the approach is guaranteed by the applied formal design method.

Categories and Subject Descriptors

[**Embedded and cyber-physical systems**]: Robotics—*Robotic control*; [**Applied computing**]: Physical sciences and engineering—*Engineering*

Keywords

mobile robots, route planning, signal temporal logic, optimization

1. INTRODUCTION

Optimal route planning based on transport demands is an intensively investigated topic in engineering fields. Depending on the applied model and assumptions, the computational complexity of such tasks and the effectiveness of the solution moves on a wide scale.

The problem itself generally consists of numerous autonomous guided vehicles (AGV) moving along given routes in a closed space (e.g. in an industrial plant), assuming a microscopic routing environment (i.e., the size of the vehicles is not negligible compared to the available space). This environment can be modeled as a directed graph, with only one agent allowed at a time in a given node or edge [1], which is suitable for a physically large setting, but might prove to be ineffective in a more crowded location. As another possible approach, the plant can be modeled as a coordinates system, in which agents can move freely with the exception of a number of obstacles or restricted zones.

Some of the solutions concentrate on giving suboptimal but real-time solution for the problem, using greedy iterative algorithms or heuristics. In the simplest case, the route planning of the agents is carried out in a completely independent manner: knowing the location of the obstacles, each agent computes a path locally, and avoids collision with other vehicles in real-time. This approach however is neither optimal regarding the completion time of the movements, nor has any guarantee to prevent situations like a deadlock formation. More complex solutions feature distributed calculation, but with a centrally accessible resource (like already planned paths of the other vehicles) [1, 2].

It is also possible to model route planning tasks as optimization problems [3]. These algorithms are indeed capable of giving the truly optimal solution regarding completion time, guarantee the collision-free and deadlock-free operation on the price of high computational complexity which might hamper real-time application.

Linear Temporal Logic (LTL) is a formalism originally developed for the formal design and verification of computer programs. In essence, LTL extends the set of operators familiar from Boolean logic with tools to describe time dependencies between the statements (such as 'always', 'never', 'eventually', 'next step', etc.). Signal Temporal Logic (STL) further extends the possibilities offered by LTL by introducing quantitative operators regarding time in LTL formulas. STL has been successfully used for describing traffic-related systems, due to its ability to express complex rule sets or complicated time dependencies [4, 7, 8].

DOI: https://doi.org/10.26493/978-961-7055-82-5.43-47

One of the first results on the successful application of LTL in vehicle routing is [9], where the computation task is written as a network flow problem with constraints. In [10], an incremental algorithm is given for the complex path planning of a single robot, where the specifications are given using LTL. A motion planning method for multiple agents is presented in [11], where STL is used for describing specifications, taking into consideration imperfect communication channels, too.

The above results motivated us to use STL in our work for route planning. The main novelty of our approach is the division of the computation into two levels which allows us to handle a relatively long prediction horizon with a singificantly reduced computation time.

2. PROBLEM FORMULATION

The problem is to plan and track routes for multiple AGVs, each of which is assumed to move in a two dimensional closed space, independently in the x and y directions. All agents have upper bounds for speed and velocity, respectively. Each agent has a target point assigned before running the planning algorithm. The objective for each agent is reaching its target point in minimal possible time, without colliding with obstacles or with each other.

Let N be the number of agents. On the low level, we model the agents as simple mass-points in a two dimensional cartesian coordinates system. Therefore, the motion of each agent can be described in continuous time as two double integrators with acceleration command inputs, resulting in $4N$ state variables (coordinates and velocities), and $2N$ inputs. This model is discretized in time using a uniform sampling time t_s. Let $x_i(k)$ and $y_i(k)$ denote the x and y coordinates of the ith robot at time step k, respectively. The inputs for agent i at time step k are denoted by $u_{xi}(k)$ and $u_{yi}(k)$ along the x and y coordinates, respectively.

The borders of the environment, as well as the rectangle-shaped obstacles can be introduced as simple constraints for the x_i and y_i state variables. Let the number of obstacles be M, each obstacle being defined by its lower-left and upper-right corners $(a_1^{(l)}, b_1^{(l)})$ and $(a_2^{(l)}, b_2^{(l)})$. This means that avoiding obstacles can be written as a set of linear constraints:

$$\{x_i < a_1^{(l)} \text{ or } x_i > a_2^{(l)} \text{ or } y_i < b_1^{(l)} \text{ or } y_i > b_2^{(l)}, \quad \forall\, i, l\}$$

Collision-aviodance between agents is realized using a threshold value δ and the constraints $|x_i(k) - x_j(k)| > \delta \;\forall\; i \neq j$. Note that due to the discrete-time model, this threshold must be carefully chosen considering maximum vehicle speed.

The planning itself is run on a finite time horizon, consisting of T steps. It is assumed, that all vehicles can reach their goal state within this time (given a reasonably large T). However, the computation should work even if some of them is not able to fulfil this constraint.

The optimization problem is minimizing the following objective function:

$J(x,u) = \sum_{k=1}^{T}\sum_{i=1}^{N}(|x_i(k) - x_{ti}| + |y_i(k) - y_{ti}|)$, subject to the collision, obstacle-avoidance and possible additional constraints, where x_{ti} and y_{ti} denote the prescribed target coordinates corresponding to agent i.

3. ONLINE ROUTE PLANNING BY TEMPORAL LOGIC CONSTRAINED OPTIMIZATION

Based on the problem formulation above, it can be clearly seen, that solving the routing problem requires at least $4NT$ variables, with $\mathcal{O}(2N^2) + \mathcal{O}(4NM)$ constraints, resulting from vehicle-vehicle collisions and obstacle avoidance respectively. Feasibility of obtaining solution in real-time is highly dependent on the applied solver, and on the constraints themselves.

Our experiments showed that the problem scales badly when the number of agents or the number of obstacles is increased. To overcome the issue, a two phase planning is proposed.

The double-integrator model is used only in the second phase, when a $\mathcal{O}(2 * N^2)$ part of the constraints can be omitted. Moreover, the second phase can be run individually for all agents, each having only $4T$ state variables and only $4M$ constraints resulting from obstacle avoidance (although some constraints for following the first-phase must be added, as described below).

3.1 First phase planning

In the first phase, a coarse route is designed with a relatively large sampling time $t_s^{(1)}$ to have as short prediction horizon as possible. Moreover, only the coordinates of the agents are taken into consideration, and the computed input ($\bar{u}_{xi}$, $\bar{u}_{yi}$) is the required coordinate change in each direction for agent i in each time step. This results in simpler dependency between the input and the desired output, and considerably reduces the required computation time.

$$\begin{cases} x_i(k+1) = x_i(k) + \bar{u}_{xi}(k) \\ y_i(k+1) = y_i(k) + \bar{u}_{yi}(k) \\ |\bar{u}_{xi}(k)| < K \\ |\bar{u}_{yi}(k)| < K \end{cases} \quad i = 1...N$$

where K is an upper bound for coordinate changes which is determined using the maximum speed of the vehicle and the sampling time t_s.

The description of the rules for such a simple system is quite straightforward using temporal logic. Given a rectangle shaped obstacle as defined above, the STL formula for avoidance looks like:

$$\bigcirc_{[0,T]} \left(x_i < a_1^{(l)} \text{ or } x_i > a_2^{(l)} \text{ or } y_i < b_1^{(l)} \text{ or } y_i > b_2^{(l)} \right)$$

. For avoiding collisions between the vehicles, we use:

$$\bigcirc_{[0T]} \left(|y_i(t) - y_j(t)| > dist \text{ or } |x_i(t) - x_j(t)| > dist \right)$$

. (Here, the notation $\bigcirc$ is the so-called *always* operator, which means, that the formula must be true in all time instants in the interval $[1, T]$ - which is the whole time of the planning in our case.)

3.2 Second phase planning

The double-integrator model for the agents is used in the second phase, where we assume, that agent i has a source and a target point, as well as a set of intermediary route points $P_i = \{(x_{pi}, y_{pi}) | p = 1...T-1\}$ known from the previous phase. The planning is done using a rolling horizon approach, using a smaller sampling time $t_s^{(2)} = t_s^{(1)}/k$, $k > 1$. In the simplest case this means that at route point p, the $p+2$-th point is set as a target. We use a temporal logic formula to ensure that the agent will be in the prescribed proximity of the point $p+1$, and planning is done on a horizon of $T_2 = 2 \cdot k$. This can be easily extended to include more intermediary route points. This planning step is repeated when the agent reaches the route point $p+1$. It can be shown, that by choosing this Δ proximity correctly, we can guarantee, that no collision between the agents can occur. The concept of the proof is that if the distance threshold from the first phase is equal to the half of the distance, an agent can move in the rough plan in one step ($\delta = K/2$), and we restrict the agents to remain in the $\Delta = \delta/2$ proximity of the given point for the respective time interval, then the agents are always located inside different non-overlapping $\delta \times \delta$-sized squares. The concept is illustrated in Figure 1.

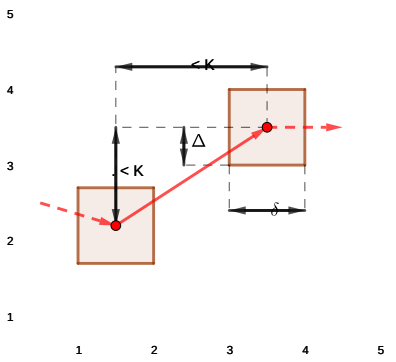


Figure 1: Distances kept by the agents during the first and second phase planning

4. ILLUSTRATIVE EXAMPLE

In order to illustrate the concept, a case study containing ten agents was carried out. As it is visible in Fig. 2, we have a 16×10 units floorplan, containing 7 obstacles that must be avoided. The first-phase planning is running on a 40 step long horizon, which means at least $10 \times 40 \times 2 = 800$ continuous variables (x and y directions for all agents and all discrete time points). The number of temporal logic formulas is 10×7 (obstacles) $+ 10 \times 9/2$ (collisions) $= 115$, resulting in approximately 400×40 constraints for the optimization problem. The exact number of variables in our simulation (produced by the STL parser) was 10660 continuous and 33110 integer variables. The solution of the problem on an average consumer-grade laptop using two processor cores was 217.7 seconds.

To illustrate the detailed (second-phase) planning, we show the planned routes and the generated input signals for only one agent corresponding to obstacle avoidance. The situation is shown in Fig. 3. Here, we have the first-phase plan generated for the agents, and we use the second phase computation to calculate the detailed plan. The discretization time for the second phase was $t_s^{(2)} = t_s^{(1)}/10$, with a horizon of $T = 50$ steps. The rough plan is marked by the blue circles and consists of 5 intermediary points. The detailed plan is marked by red stars. As it is visible in the figure, the agent remains in the prescribed proximity of the first-phase plan. It must be noticed, that the agent correctly avoids the obstacle's corner, which was to be hit following directly the first phase rough route. The input signals (acceleration commands) generated for the agent are displayed in Fig. 4

5. CONCLUSION

In this paper, we presented a possible way for describing route planning problems as optimisation problems, using the formalism offered by signal temporal logic. The planning phase is divided into two parts: in the first phase, we use a low-complexity model for creating a rough plan, taking into consideration the obstacles and vehicle-vehicle interactions as well. The algorithm calculates a set of consecutive intermediary route points for each agent, ensuring conflict-free behavior provided that agents are in the given proximity of the points for the respective time interval. In the second phase, each agent computes its own path, considering the points and intervals given in the first phase. Thus, vehicle-vehicle interactions need not to be checked on the detailed planning level, only smooth maneuvering and obstacle avoidance between the points are required.

Acknowledgements

This work has been supported by the research program titled "Exploring the Mathematical Foundations of Artificial Intelligence" (2018-1.2.1-NKP-00008). The partial support of the projects GINOP-2.3.2-15-2016-00002 and EFOP-3.6.3-VEKOP-16-2017-00002 is also acknowledged. T. Péni is the grantee of the Bolyai János Research Scholarship of the Hungarian Academy of Sciences. The work has been also supported by the UNKP-19-4-BME-29 New National Excellence Program of the Ministry of Human Capacities.

6. REFERENCES

[1] Gawrilow, E., Köhler, E. and Möhring, R. and Stenzel, B. Dynamic Routing of Automated Guided Vehicles in Real-time (2008). In: *Krebs HJ., Jäger W. (eds) Mathematics – Key Technology for the Future. Springer, Berlin, Heidelberg* DOI: 10.1007/978-3-540-77203-3_12.

[2] B. Csutak, T. Péni and G. Szederkényi. An optimization based algorithm for conflict-free navigation of autonomous guided vehicles In *Pannonian Conference on Advances in Information Technology* (2019), pp. 90-97.

[3] T. Nishi, M. Ando and M. Konishi. Distributed route planning for multiple mobile robots using an augmented Lagrangian decomposition and coordination technique In *IEEE Transactions on Robotics*, vol. 21, no. 6, pp. 1191-1200, Dec. 2005.

[4] E. S. Kim, S. Sadraddini, C. Belta, M. Arcak and S. A. Seshia. Dynamic contracts for distributed temporal logic control of traffic networks In *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, Melbourne, VIC, 2017, pp. 3640-3645.

[5] Raman, V. and Donzé, A. and Maasoumy, M. and Murray, R. and Sangiovanni-Vincentelli, A. and A Seshia, S. (2014). Model Predictive Control with Signal Temporal Logic Specifications In *53rd IEEE Conference on Decision and Control* (2014): 81-87.

[6] A. Donzé and V. Raman. BluSTL Controller Synthesis from

Signal Temporal Logic Specifications In *1st and 2nd International Workshop on Applied verification for Continuous and Hybrid Systems*, 2015

[7] S. Coogan and M. Arcak. Freeway traffic control from linear temporal logic specifications In *2014 ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)*, Berlin, 2014, pp. 36-47.

[8] N. Mehr, D. Sadigh, R. Horowitz, S. S. Sastry and S. A. Seshia. Stochastic predictive freeway ramp metering from Signal Temporal Logic specifications In *2017 American Control Conference (ACC)*, Seattle, WA, 2017, pp. 4884-4889.

[9] S. Karaman, and E. Frazzoli. Linear temporal logic vehicle routing with applications to multi-UAV mission planning. *International Journal of Robust and Nonlinear Control*, 21:1372–1395, 2011.

[10] C. I. Vasile, and C. Belta. Sampling-Based Temporal Logic Path Planning. In *Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 4817-4822, 2013.

[11] Z. Liu, J. Dai, B. Wu, and H. Lin. Communication-aware Motion Planning for Multi-agent Systems from Signal Temporal Logic Specifications. In *Proceedings of the 2017 American Control Conference*, 2516-2521, 2017.

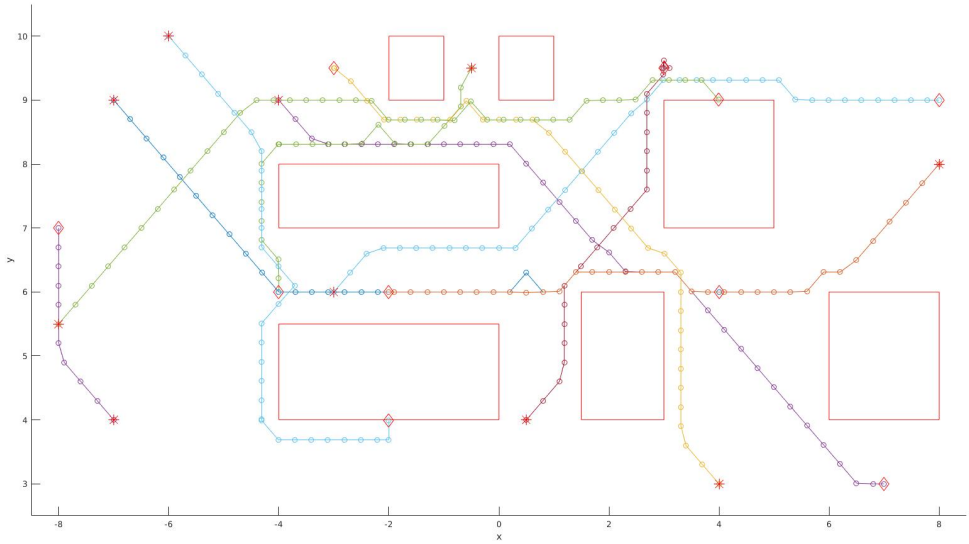


Figure 2: Paths created by the coarse first-phase planning for 10 agents
The start and target positions of the agents are marked by stars and diamonds, respectively.

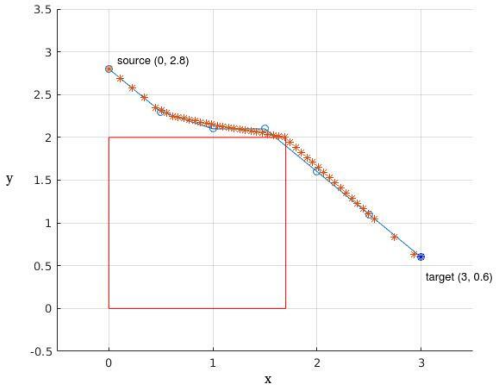


Figure 3: The first- and second phase plans for one agent

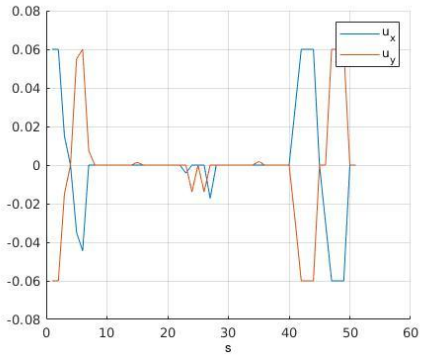


Figure 4: Input signals computed for the agent

Decolorization of Digital Pathology Images: A Comparative Study

Krishna Gopal Dhal[1], Swarnajit Ray[2], Arunita Das[3], Iztok Fister Jr.[4], Sanjoy Das[5]

[1]Dept. of Computer Science and Application, Midnapore College (Autonomous), Paschim Medinipur, West Bengal, India. *Email: krishnagopal.dhal@midnaporecollege.ac.in*
[2]Learningmate Solutions Pvt. Ltd., Kolkata, West Bengal, India. *Email: swarnajit32@gmail.com*
[3]Dept. of Information Technology, Kalyani Govt. Engineering College, Kalyani, Nadia, India. *Email: arunita17@gmail.com*.
[4]Faculty of Electrical Eng. and Computer Sc., University of Maribor, Slovenia, *Email: iztok.fister1@um.si*.
[5]Dept. of Eng. and Technological Studies, University of Kalyani, Kalyani, India, *Email: dassanjoy0810@hotmail.com*

ABSTRACT

The major demerit of color to gray conversion is the loss of visually important image features. Digital pathology images are treated as the gold Standard for detection of various diseases, especially for the different types of cancer. Digital pathology images are color in nature, i.e. each pixel is a color vector represented by three values. Thus, the processing of these images requires high computational time. If these color images are converted into one dimensional gray images, then processing time can be reduced, which will help the Computer-Aided Diagnosis (CAD) system significantly for accurate classification and detection of different types of diseases. Therefore, this study focuses on the fast conversion of color digital pathology images into gray images. In order to do that, seven well established color to gray conversion, techniques have been employed for producing gray images with salient features. The outcomes have been validated visually and numerically.

KEYWORDS

Digital Pathology Images, Decolorization, Color to Gray Conversion, Gray Scale, RGB2GRAY.

1. Introduction

Computer assisted pathology and microscopy image analysis, assist the decision making for automated disease diagnosing, as they provide digital images related to certain kinds of disease using Computer-Aided Diagnosis (CAD) systems, which facilitates quantitative and qualitative medical results with a high throughput processing rate [1, 2, 3]. At present, automated medical diagnosing has attracted the attention of several pathologists in research and clinical practice , since CAD systems reduce human error, false positive results and time complexity, while pathology imaging provides more accurate results, faster and reproducible image analysis. Digital pathology images are stored as high-resolution color images, i.e. each pixel is represented as a three-dimensional vector, namely R, G, and B, and, due to that, they are of the order M×N×3, where M and N indicate the number of row and column respectively. Therefore, several image processing techniques, like enhancement, segmentation, require high computational effort. In order to overcome this issue, if these high dimensional images can be reduced to the order M×N with each pixel as a single scalar value, then the computation for applying these techniques reduces drastically. Another benefit is that this conversion facilitates the application of single-channel algorithms on color images, like Canny operator for edge detection [4]. In literature, this dimension reduction is considered as color to gray scale image conversion, or decolorization.

Several color to gray scale conversion techniques have been developed by following the human perception of brightness and contrast, and they proved their efficiency in the traditional color image decolorization field [5-12]. However , the utilization of decolorization techniques in the Digital Pathology domain is a little bit different. Information loss minimization for a specific image is the main aspiration. Therefore, this study utilizes these developed color to gray conversion techniques for the decolorization of pathology images to prove their efficacy in this medical image domain. All color to gray conversion techniques are categorized into three classes, namely Local, Global, and Hybrid. In local processing based techniques [5, 6], the same color pixel within an image can be mapped into different gray values, depending on the local distributions of colors, which is generally not desired. Compared to local, global processing methods [4, 7-12] are able to produce natural looking images. Several hybrid methods have also been developed by considering global and local contrast or features for conversion [13, 14], but, it is also true that local processing and utilization of local information based statistics take large computational time. Therefore, this letter considers only global processing based techniques [4, 7-12], which are discussed in the next Section.

The paper is organized as follows: Section 2 discusses all the global color to gray conversion techniques. Section 3 describes the experimental results and the paper is concluded in section 4.

2. Decolorization Models

The existing global decolorization, methods have been presented in this Section.

2.1. MATLAB Procedure (rgb2gray)

DOI: https://doi.org/10.26493/978-961-7055-82-5.49-52

Generally, global mapping techniques convert a color image (RGB) into a grayscale image (G) by a linear weighting of the R, G, and B channels, i.e. $\mathrm{G(i,j)} = \sum_{\mathrm{c=R,G,B}} w_c RGB_{\mathrm{c}}\,(\mathrm{i,j})$, where $\sum_{\mathrm{c=R,G,B}} w_c = 1$. Here, the three linear weighting parameters w, should be estimated on the basis of some models.
In the MATLAB (Matrix Laboratory) software, developed by MathWorks [11], an RGB image converts into gray-scale by the following weighting formula:

$$G = 0.2989 \times R + 0.5870 \times G + 0.1140 \times B \quad (1)$$

2.2. Color2Gray

This decolorization model was developed by Gooch et. al. in 2005 [7]. The proposed model uses CIELAB color space, and maintains color contrast between pixel pairs by optimizing an objective contrast function.

2.3. Real-Time Contrast Preserving Decolorization (RTCPD)

It has previously been said that $\mathrm{G(i,j)} = \sum_{\mathrm{c=R,G,B}} w_c RGB_{\mathrm{c}}\,(\mathrm{i,j})$. In 2009, Lu et. al. [8] also developed a decolorization model called Real-Time Contrast Preserving Decolorization (RTCPD) by optimizing the linear weights w_c by minimizing the gradient error energy function.

2.4. Gradient Correlation Similarity for Decolorization (GcsDecolor)

The GcsDecolor [9] model was proposed by Liu et. al. in 2015, which is the variant of RTCPD. Gradient correlation similarity (Gcs) measure were utilized in GcsDecolor. Two variants of GcsDecolor have been developed by the authors. The first one is iterative GcsDecolor and the other is discrete searching GcsDecolor. Discrete searching based GcsDecolor is utilized here, due to its simplicity and run time efficiency.

2.5. Semi-Parametric Decolorization (SPDecolor) model

This Semi-Parametric Decolorization technique is another variant of RTCPD proposed by Liu et. al. in 2016 [4]. SPDecolor has the strength of the parametric contrast preserving method and the non-parametric rgb2gray method.

2.6. Color to Gray Conversion by Correlation (CorrC2G)

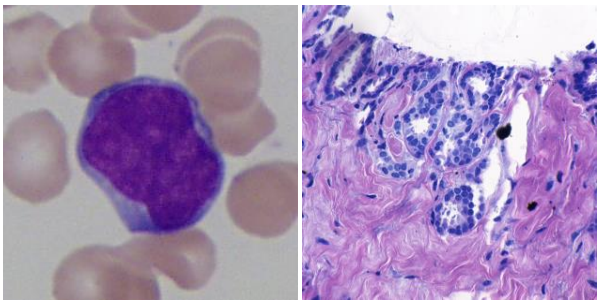
The CorrC2G [10] technique was proposed by Nafchi et. al. in 2017, where the linear weighting parameters (w) have been estimated using the correlation information between each band of RGB image and a contrast image. This method also does not require any edge information or any optimization.

2.7. Parametric ratio-based method for efficient contrast preserving decolorization (PrDecolor)

This PrDecolor was proposed by Xiong et. al. in 2018 [12]. The method is a contrast preserving multivariate parametrical constraint based decolorization model.

3. Experimental Results

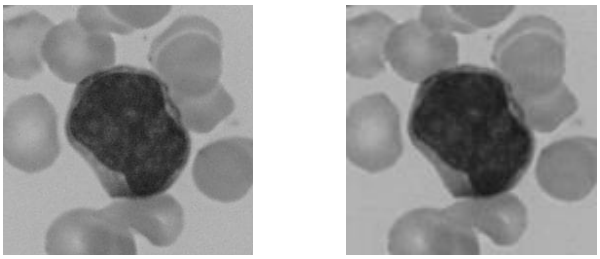
The experiment was performed over 40 color hematopathology and histopathology images with MatlabR2016a and a Windows-10 OS, x64-based PC, RIZEN CPU, 3.6 GHz with 8 GB RAM. The proposed methods were tested on images taken from the ALL IDB dataset [15] and UCSB Bio-Segmentation Benchmark dataset [16, 17].



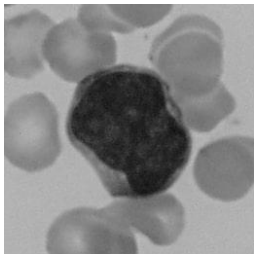
(a) (b)

Fig.1. (a) Original image of Acute Lymphoblastic Leukemia
(b) Original image of Breast histopathology

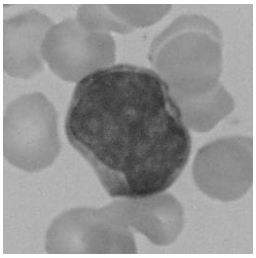
The decolorization efficacy of the proposed models has been judged by computing three quality parameters, namely the Color-to-Gray Structural Similarity (C2G-SSIM) index (C2G-SSIM) [10, 20], Edge based Contrast Measure (EBCM) [18], and Entropy [19]. C2G-SSIM [10, 20] is a color to gray evaluation metric based on the popular image quality assessment metric SSIM. It demonstrates higher correlation with human subjective evaluations. It is expected that the efficient color to gray conversion technique preserves the edge information. Therefore, EBCM has been utilized to measure the edge information, as it is less sensitive to digitization effects and noise. Entropy [19] value reveals the information content in the image. If the distribution of the intensities is uniform, then it can be said that a histogram is equalized and the entropy of the image is more.



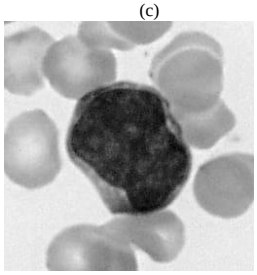
(a) (b)



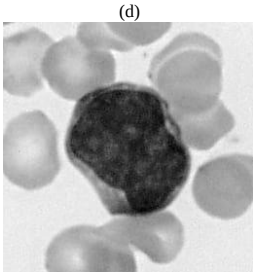
(c)



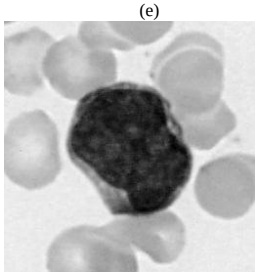
(d)



(e)

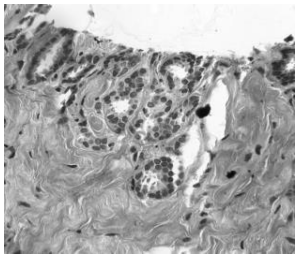


(f)

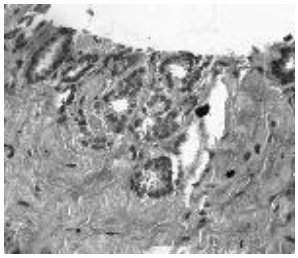


(g)

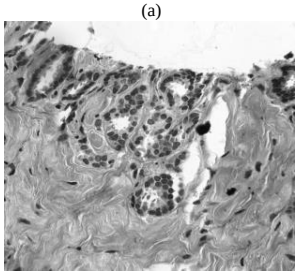
Fig.2. Results of decolorizarion for Fig.1(a): (a) rgb2gray [11] (b) Color2Gray [7] (c) RTCPD [8] (d) GcsDecolor [9] (e) SPDecolor [4] (f) CorrC2G [10] (g) PrDecolor [12].



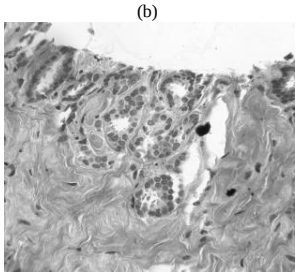
(a)



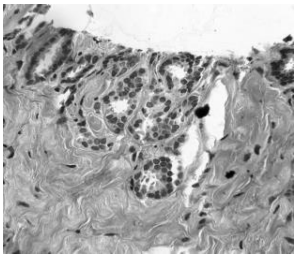
(b)



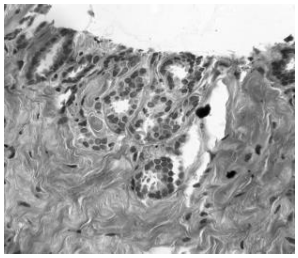
(c)



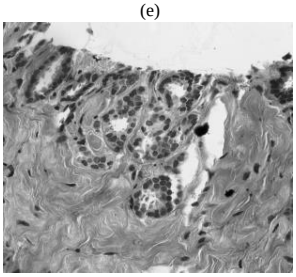
(d)



(e)



(f)



(g)

Fig.3. Results of decolorizarion for Fig.1(b): (a) rgb2gray [11] (b) Color2Gray [7] (c) RTCPD [8] (d) GcsDecolor [9] (e) SPDecolor [4] (f) CorrC2G [10] (g) PrDecolor [12].

Table 1. Average Quality parameters over 40 images

Method	C2G-SSIM	EBCM	Entropy
rgb2gray [11]	0.8912	183.24	7.19
Color2Gray [7]	0.8314	172.13	6.98
RTCPD [8]	0.8914	183.57	7.19
GcsDecolor [9]	0.8598	174.90	7.11
SPDecolor [4]	0.9030	187.38	7.23
CorrC2G [10]	0.9032	187.98	**7.25**
PrDecolor [12]	**0.9035**	**188.74**	**7.25**

/* Best results obtained are given in bold*/

Table 2. Computational time of decolorization methods

Method	Fig.1(a) 257x257	Fig.1(b) 896x768
rgb2gray [11]	**0.0086**	**0.0221**
Color2Gray [7]	157.01	263.23
RTCPD [8]	0.0721	0.0636
GcsDecolor [9]	0.0397	0.0723
SPDecolor [4]	0.0942	1.0967
CorrC2G [10]	0.0187	0.0385
PrDecolor [12]	2.9678	27.8316

/* Best results obtained are given in bold*/

3.1. Analysis of Experimental Results

Performance analysis of the considered seven decolorization models was performed by using three image quality parameters and computational time. Figs. 2 and 3 demonstrate the outcomes of the seven decolorization models over pathology images, represented as Fig. 1. Values of the quality parameters and computational times are given in Tables 1 and 2 respectively. Visual analysis of Figs. 2 and 3 shows clearly that SPDecolor [4], CorrC2G [10], and PrDecolor [12] produce better outcomes compared to other decolorization methods. However, when we compare these methods based on quality parameters, it can be seen that PrDecolor [12] outperforms the other methods.

SPDecolor [4] and CorrC2G [10] provide nearly the same results as PrDecolor [12], as their corresponding numerical values of the quality parameters are almost the same. It can also be seen that these three methods, namely, SPDecolor [4], CorrC2G [10], and PrDecolor [12] outperform the other four methods significantly in terms of quality parameters. When we consider computational time, it can be seen that the MATLAB based rgb2gray [11] method is the best method. However, among the SPDecolor [4], CorrC2G [10], and PrDecolor [12] methods, CorrC2G is associated with the lowest computational time.

4. Conclusion

This paper presents a comparative study among seven existing decolorization methods in the case of digital pathology images. The visual and decolorization quality parameters prove clearly that PrDecolor [12], proposed by Xiong et. al., provided the best outcomes compared to the other six methods. Computational time shows that the MATLAB based rgb2gray method outperformed the others, although CorrC2G [10] produced nearly the same outputs as the PrDecolor [12] method, but within the second less computational time. One challenging future direction of this study can be the application of nature-inspired optimization algorithms to set the parameters of the parametric decolorization methods by considering different objective functions.

REFERENCES

1) Gurcan, M. N., Boucheron, L. E., Can, A., Madabhushi, A., Rajpoot, N. M., & Yener, B. (2009). Histopathological image analysis: A review. IEEE reviews in biomedical engineering, 2, 147-171.
2) Irshad, H., Veillard, A., Roux, L., & Racoceanu, D. (2014). Methods for nuclei detection, segmentation, and classification in digital histopathology: a review—current status and future potential. IEEE reviews in biomedical engineering, 7, 97-114.
3) Hinojosa, S., Dhal, K. G., Elaziz, M. A., Oliva, D., & Cuevas, E. (2018). Entropy-based imagery segmentation for breast histology using the Stochastic Fractal Search. Neurocomputing, 321, 201-215.
4) Liu, Q., Liu, P. X., Wang, Y., & Leung, H. (2016). Semiparametric decolorization with Laplacian-based perceptual quality metric. IEEE Transactions on Circuits and Systems for Video Technology, 27(9), 1856-1868.
5) Neumann, L., Čadík, M., & Nemcsics, A. (2007, June). An efficient perception-based adaptive color to gray transformation. In Proceedings of the Third Eurographics conference on Computational Aesthetics in Graphics, Visualization and Imaging (pp. 73-80). Eurographics Association.
6) Smith, K., Landes, P. E., Thollot, J., & Myszkowski, K. (2008, April). Apparent greyscale: A simple and fast conversion to perceptually accurate images and video. In Computer Graphics Forum (Vol. 27, No. 2, pp. 193-200). Oxford, UK: Blackwell Publishing Ltd.
7) Gooch, A. A., Olsen, S. C., Tumblin, J., & Gooch, B. (2005, July). Color2gray: salience-preserving color removal. In ACM Transactions on Graphics (TOG) (Vol. 24, No. 3, pp. 634-639). ACM.
8) Lu, C., Xu, L., & Jia, J. (2012, November). Real-time contrast preserving decolorization. In SIGGRAPH Asia 2012 Technical Briefs (p. 34). ACM.
9) Liu, Q., Liu, P. X., Xie, W., Wang, Y., & Liang, D. (2015). GcsDecolor: gradient correlation similarity for efficient contrast preserving decolorization. IEEE Transactions on Image Processing, 24(9), 2889-2904.
10) Nafchi, H. Z., Shahkolaei, A., Hedjam, R., & Cheriet, M. (2017). CorrC2G: Color to gray conversion by correlation. IEEE Signal Processing Letters, 24(11), 1651-1655.
11) MATLAB and Image Processing Toolbox Release 2012b, The MathWorks, Inc., Natick, Massachusetts, United States.
12) Xiong, J., Lu, H., Liu, Q., & Xu, X. (2018). Parametric ratio-based method for efficient contrast-preserving decolorization. Multimedia Tools and Applications, 77(12), 15721-15745.
13) Du, H., He, S., Sheng, B., Ma, L., & Lau, R. W. (2014). Saliency-guided color-to-gray conversion using region-based optimization. IEEE Transactions on Image Processing, 24(1), 434-443.
14) Jin, Z., Li, F., & Ng, M. K. (2014). A variational approach for image decolorization by variance maximization. SIAM Journal on Imaging Sciences, 7(2), 944-968.
15) Labati, R. D., Piuri, V., & Scotti, F. (2011, September). All-IDB: The acute lymphoblastic leukemia image database for image processing. In 2011 18th IEEE International Conference on Image Processing (pp. 2045-2048). IEEE.
16) Dhal, K. G., Ray, S., Das, S., Biswas, A., & Ghosh, S. Hue-Preserving and Gamut Problem-Free Histopathology Image Enhancement. Iranian Journal of Science and Technology, Transactions of Electrical Engineering, 1-28.
17) Dhal, K. G., Fister Jr, I., Das, A., Ray, S., & Das, S (2018) Breast Histopathology Image Clustering using Cuckoo Search Algorithm. StuCoSReC. 5th Student Computer Science Research Conference, 47-54.
18) Beghdadi, A., & Le Negrate, A. (1989). Contrast enhancement technique based on local detection of edges. Computer Vision, Graphics, and Image Processing, 46(2), 162-174.
19) Dhal, K. G., Ray, S., Das, A., & Das, S. (2018). A Survey on Nature-Inspired Optimization Algorithms and Their Application in Image Enhancement Domain. Archives of Computational Methods in Engineering, 1-32.
20) Ma, K., Zhao, T., Zeng, K., & Wang, Z. (2015). Objective quality assessment for color-to-gray image conversion. IEEE Transactions on Image Processing, 24(12), 4673-4685.

Solving multi-depot vehicle routing problem with particle swarm optimization

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

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

ABSTRACT

Multi-depot vehicle routing problem (MDVRP) is an optimization problem with practical real-world applications in the commercial transportation sector. It deals with the optimization of the time and cost of the transportation of goods from and to customers from numerous predefined serving depots. The *multi-depot* variant adds constraints of multiple serving depots and variable customer serving capacity and is a NP hard problem. In this paper, we present an application of Particle Swarm Optimization (PSO) for continuous optimization to MDVRP, where nature-inspired optimization framework NiaPy is used. As MDVRP is a discreet optimization problem, but NiaPy is suited to work only with continuous optimization problems, a transformation with the repairing mechanism must be used. Our proposed approach is presented in detail and is tested on several standard MDVRP benchmark sets to provide a sufficient evidence of usability of the approach.

Keywords

Multi-depot Vehicle Routing Problem, Continuous optimization, Particle Swarm Optimization

1. INTRODUCTION

Optimization is a daily problem we face with in an increasing number of different logistics services, such as for example mail delivery, passenger transportation and other transportation of goods [1], [2], [4]. Because solving such problems is - due to restrictions on the route often quite difficult, we mostly rely on computer intelligence. By this we mean different algorithms that have some heuristics rules, which result in good solutions. They are classified as metaheuristic algorithms and often also referred to as nature-inspired or evolutionary algorithms (EA) [6], [9].

In order to execute the experiment of a comparison between different EA, we firstly developed a system that allows application of any EA to the vehicle routing problem (VRP). Then we tackled the VRP using five different evolutionary algorithms, which are genetic algorithm (GA), evolution strategy (ES), differential evolution (DE), particle swarm optimization (PSO) and harmony search (HS). Considering the obtained results, we decided to pay more attention to the PSO algorithm.

PSO is a popular algorithm for solving many complex optimization problems, including routing problems. In 2008 Mohemmed et al. [16] used PSO for simple routing problem with custom priority-based encoding and heuristic operator to prevent loops in the path construction. Next in 2009, Ai and Kachitvichyanukul [17] presented PSO with multiple social structures to solve VRP with simultaneous pickup and delivery. Yao et al. [18] proposed custom particle swarm optimization algorithm for carton heterogeneous vehicle routing problem. Kumar et al. [19] proposed a PSO for vehicle routing problem with time window constraint. More recently, Norouzi et al. [20] extended VRP with time window problem with additional fuel consumption constraint and solved the problem also with PSO. All these approaches treat routing problem as a discreet problem. As we used optimization framework, which only works with continuous optimization problem, the approaches from the referenced papers could not be used and a transformation was necessary.

The remaining of the paper is structured as follows. Second section presents and formulates MDVRP and PSO. Next section presents our proposed approach and its implementation with NiaPy framework. Fourth section presents the results of the experiment of the proposed approach on the standard benchmark sets. Last section discusses the results of the experiments and finishes with the concluding remarks.

2. SOLVING MDVRP

Logistic companies and other carriers, tasked with transporting or picking up shipments, face with route optimization on a daily level. The well-optimized route, taken by their fleets of vehicles, means saving fuel and thus reducing overall daily cost. From this we can see that similar scenarios are present in in everyday life and present a problem worth solving well.

The described problem is called VRP and was first addressed by George Dantzig and John Ramser in 1959, as a solution to optimize fuel delivery. It is a NP hard combinatorial optimization problem, generalized from travelling salesman problem, so there is no polynomial time solution known to it [1], [3]. The result of the problem is the optimal set of routes for multiple vehicles, transporting goods to or from customers, subject to restrictions along the routes. We also need to keep in mind that only one vehicle can visit specific customer at a time [1], [7].

Over time, different classifications of the problem have formed due to differences in constraints. The most common versions of the problem are VRP with capacity constraints (CVRP), multi-depot VRP (MDVRP) and VRP with time windows constraints (VRPTW). In addition, just about every problem classification also has a certain distance limit of the individual vehicle [1], [4].

We focused on solving MDVRP classification of the problem, which is also less frequently referred to as multi-depot capacitated vehicle routing problem (MDCVRP) [4]. The version of the problem, in addition to multiple customers, consists of multiple depots and thus more predefined vehicles - each vehicle can carry

DOI: https://doi.org/10.26493/978-961-7055-82-5.53-56

a certain payload, travel a certain distance and must eventually return to its starting depot [3], [11]. If any of the restrictions is violated, penalty function is used to punish the vehicle with a certain mark-up value. Because MDVRP is represented as a directed graph, every node of customer and depot has its own x and y coordinate pair [4], [5]. When looking for a solution to the problem, we usually divide it into three phases, collectively referred to as decision making or decision hierarchy in MDVRP. We call them the merging phase, the routing phase and the scheduling phase. In the first phase, we try to allocate customers to the individual depots according to the distance, which is present between them. The second phase, with previously divided customers, draws up several routes, which vehicles will take. After that, each path is sequenced in the third phase [3], [11].

Figure 1 is an example of MDVRP problem, where letters A and B represent depots or vehicles with a maximum capacity of 10 units of weight, and circles with numbers represent different customers. There are four paths between depots and customers, based on genotype numbers that are converted to phenotype numbers by the first conversion method. The routes were determined according to the previously mentioned MDVRPs decision hierarchy.

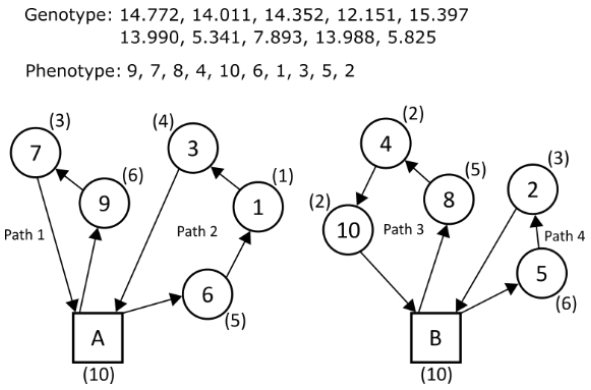


Figure 1. MDVRP example.

2.1 Particle Swarm Optimization for MDVRP

The system we developed for experiment purposes, supports importing any EA from the library or microframework called NiaPy. It was developed by researches from the Faculty of Electrical Engineering and Computer Science and the Faculty of Economics and Business from University of Maribor. Main purpose for developing the framework was lack of easy and fast use of EA, since the own implementation of a single algorithm is often difficult and time-consuming. The library architecture is divided into two parts, which we call algorithms and benchmarks. In addition to the developed normal EA versions, we can also find hybrid variants, such as hybrid bat algorithm and self-adaptive differential evolution algorithm. The framework supports EA startup and testing with predefined and generated comparisons, while also allowing the export of results in three different formats, which are LaTeX, JSON and Excel [13].

Although some similar frameworks for managing nature-inspired algorithms already exist, NiaPy differs mainly in minimalism and ease of use. Its main functions are weak coupling, good documentation, user friendliness, fair comparison of algorithms, quick overview of results and friendly support community. NiaPy project is designated as open source and licensed under an MIT license. Because the framework is developed in Python programming language, installation is possible on all systems, which have the support for the language and installed PIP package manager. Due to further development, new algorithms are being added to the framework and the previously implemented algorithms are being further improved [13].

One of the algorithms implemented in NiaPy is also PSO, which is a population stochastic optimization algorithm and belongs to the EA group. The algorithm, which is based on the behavior of the swarms of animals such as fish and birds was first developed by James Kennedy and Russel Eberhart in the mid-90s of the 20th century. It was created as a by-product of the desire to graphically represent various flight patterns of animals [6], [8], [14], [15].

The PSO population is referred to as a swarm and instances inside of it flying particles, which are constantly moving inside of a hyperdimensional search space. The position between particles is determined with social-psychology tendency to be better and to imitate other, closer instances [6], [8]. Each particle is moving with its own speed, knows its previous locations and never goes extinct, because there is no selection, mutation or recombination [10]. Velocity is changed in every generation/iteration. For changing velocity, we have three different parts, namely previous velocity, cognitive component and social component

First component represents the memory of the previous direction motion and prevents the current flight direction from drastically changing. Second component expresses the performance of the current particle with respect to past results. Lastly, third component expresses the performance of the current particle with respect to some group of particles or neighbors around it [8].

During the operation, we need to store some other information like information about the best found location of each particle (pbest), information about the best found location of the currently selected part of the swarm (lbest) and information about the best found location of the particle of any swarm (gbest). When finding new top locations, current values need to be updated [6], [8]. The PSO algorithm uses fitness function for guidance of the search over the search space. When the stopping condition is meet, algorithms returns global best solution found [10].

3. IMPLEMENTATING PSO FOR MDVRP

For the purpose of the experiment we used programming language Python to develop a system, which allows application of any EA from NiaPy library to the different MDVRP examples. The system can handle CSV cases of VRP made by Cordeau [12].

The system consists of several different classes, one of which is class Evaluation, which is responsible for solving the problem. In the class we firstly convert given genotype from imported EA to an appropriate phenotype, which can be then used to tackle the problem. First genotype to phenotype conversion assigns ascending index number to each gene, according to its value in the array. Second conversion assigns genes, representing nodes or customers, to specific vehicles based on the value scale, made of number of depots.

The fitness function of the program itself is quite simple, since it only adds up all the paths made to the total distance. This then represents the fitness value of the current instance. It is also important to add penalty to the result, if solution violated any of the limitations of the MDVRP. This is done by checking the limits during the program operation and in case of a violation, send the current result into the penalty function, which then adds a certain distance unit to the distance already completed.

Another important class is Graph, which is responsible for drawing different graphs and connecting nodes of customers and depots on it with paths. Each customer and depot object have its own coordinate pair x and y, which is then plotted on a graph and connected to paths, obtained from the current result object. The final graph thus illustrates all the paths made by vehicles between all the customers and depots. The class can also draw a final bar graph of all fitness values across generations, received through the objects of all solutions. Image Class on the other hand, captures an image from a drawn graph and saves it to the appropriate directory. It can also use previously stored images to generate animated gifs that show the composition of the found route.

The program itself starts in its main function, where we specify desired parameters, such as population size, number of generations to be made, number of instances inside one generation, seed value and genotype to phenotype conversion. Results are displayed on the console at the end of the solving.

4. RESULTS OF THE EXPERIMENT

The experiment we conducted was performed on five MDVRP cases and with five competitive evolutionary algorithms (Particle Swarm Optimization **PSO**, Evolutionary Strategy **ES**, Genetic Algorithm **GA**, Differential Evolution **DE** and Harmony Search **HS**), using settings of 10 generations, 5 instances and 20 units of distance as the penalty value. Each of five test cases was run with a random seed value between 1.000 and 10.000 and with first genotype to phenotype conversion. Unfortunately, the testing was only performed once due to poor hardware capabilities, which were processor Intel Core i5-6267U 3.300Ghz, graphic card Intel Iris Graphics 550, Kingston 8Gb RAM and Liteon 250Gb SSD. The experiment was performed on the Linux operating system Ubuntu 18.04.

The exact NiaPy algorithms, which were used in the testing are GeneticAlgorithm for GA, EvolutionStrategyMpL for ES, DifferentialEvolution for DE, ParticleSwarmAlgorithm for PSO and HarmonySearch for HS. Settings of individual evolutionary algorithm were left at default values from the NiaPy framework.

When testing on the first and simplest example of the problem *pr00 (2 depots/vehicles and 10 customers)*, PSO fund the fourth best route, which is not exactly good. It was overtaken by all the algorithms except the ES, which achieved an even worse fitness result. Comparing on the execution time of the algorithms, the PSO achieved the best value, although it was quite like the DE and HS values. All the results described are shown in Table 1.

Table 1. Test results of solving the example *pr00*

Algorithm	Fitness (unit of distance)	Run time (seconds)
GA	688.02	168.59
ES	774.52	155.93
DE	711.88	148.61
PSO	749.02	148.43
HS	679.10	148.78

GA achieved the best fitness value in solving the second MDVRP case *pr04 (4 depots/vehicles and 192 customers)*. Again, the PSO found the fourth best route with 13603.86 units of distance, and the worst path was found by the ES. This time, PSO solved the problem with the second fastest time, as it was overtaken by the DE for only one second. Results are shown in Table 2.

Table 2. Test results of solving the example *pr04*

Algorithm	Fitness (unit of distance)	Run time (seconds)
GA	13282.01	3906.96
ES	13640.05	3808.14
DE	13476.95	3713.49
PSO	13603.86	3714.37
HS	13573.96	3865.54

PSO achieved the second-best fitness value when solving the third MDVRP case - *pr08 (6 depots/vehicles and 144 customers)*, which is interesting because the example was a bit easier that he previous one. It also solved the problem as the fastest optimization algorithm of all tested. Overall, the algorithm proved to be a good choice for solving the specific case. Test results are recorded in Table 3.

Table 3. Test results of solving the example *pr08*

Algorithm	Fitness (unit of distance)	Run time (seconds)
GA	13282.01	3906.96
ES	13640.05	3808.14
DE	13476.95	3713.49
PSO	13603.86	3714.37
HS	13573.96	3865.54

The fourth MDVRP case *pr14 (4 depots/vehicles and 192 customers)* was similar in complexity to the second, but with one depot less. PSO solved the problem well again, with its fitness result reaching second place, and ES reaching last place. With 3709.32 seconds, PSO solved the problem fastest once again. All the results can be seen in Table 4.

Table 4. Test results of solving the example *pr14*

Algorithm	Fitness (unit of distance)	Run time (seconds)
GA	13723.88	3896.00
ES	13885.56	3732.49
DE	13494.85	3755.60
PSO	13500.67	3709.32
HS	13873.96	3950.66

The toughest test case *pr20 (6 depots/vehicles and 288 customers)* was with 21085.43 units of distance best resolved by PSO – the rest of the algorithms were left behind by about 200 units of distance or more. It achieved the third-best computing time, and interestingly GA achieved first, although its fitness value wasn't very good. All the test results are shown in Table 5.

Table 5. Test results of solving the example *pr20*

Algorithm	Fitness (unit of distance)	Run time (seconds)
GA	21610.55	7341.09
ES	21397.89	7345.13
DE	21224.04	7487.69

PSO	21085.43	7461.45
HS	21293.09	7656.94

If we look at Table 6, we can see the rankings of the PSO algorithm in fitness scores and runtime relative to the other algorithms. The PSO reached an average of 2.6 for fitness rankings and 1.6 for the runtime, and thus solved MDVRP cases the best of all the tested algorithms. It handled more difficult cases better but achieved the fastest resolution times for all of the MDVRP examples.

We ran the test cases again with second conversion of genotype to phenotype but did not get any different results – all fitness scores and running times have overall deteriorated.

Table 6. The PSO algorithm ranks

Order of place	pr00	pr04	pr08	pr14	pr20
Fitness	4	4	2	2	1
Run time	1	2	1	1	3

5. CONCLUSIONS

In this paper we present the application of particle swarm optimization algorithm with the usage of NiaPy optimization framework on the multi-depot capacitated vehicle routing problem. Our approach differs from the similar relevant approaches in the way we represent the optimization problem. In the literature it is normal to treat routing problems as discreet optimization problems. As this was not possible with the usage of NiaPy optimization framework, we presented a method on how to solve MDVRP as the continuous optimization problem.

The proposed method was tested on several standard MDVRP benchmark sets and the results of PSO were compared with several evolutionary algorithms. The results of the experiment show that PSO of continuous optimization is a viable and competitive method for solving MDVRP, especially in the speed of the optimization – it was the fastest in three cases out of five sets. Our proposed PSO reached the best (shortest) route in only one case and it resulted with second shortest routes in two other cases. Thus, we can conclude that PSO can effectively solve MDVRP problem with competitive solutions in fastest run times out of all five included algorithms.

Future work includes the implementation of advanced PSO operators from the referenced literature and customizing them for the continuous optimization. Also, there are numerous other VRP variants, which should be tested with our proposed approach.

6. ACKNOWLEDGMENTS

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

7. REFERENCES

[1] W. Cao and W. Yang, “A Survey of Vehicle Routing Problem,” MATEC Web Conf., vol. 100, pp. 1–6, 2017.

[2] I.-M. Chao, E. Wasil, and B. L. Golden, “A new heuristic for the multi-depot vehicle routing problem that improves upon best-known solutions,” Am. J. Math. Manag. Sci., vol. 13, no. 3–4, pp. 371–406, 1993.

[3] W. Ho, G. T.S. Ho, P. Ji, and H. C.W. Lau, “A hybrid genetic algorithm for the multi-depot open vehicle routing problem,” Eng. Appl. Artif. Intell., vol. 21, pp. 401–421, 2008.

[4] S. Karakatič and V. Podgorelec, “A survey of genetic algorithms for solving multi depot vehicle routing problem,” Appl. Soft Comput. J., vol. 27, pp. 519–532, 2015.

[5] G. Laporte, M. Gendreau, J.-Y. Potvin, and F. Semet, “Classical and modern heuristics for the vehicle routing problem,” Int. Trans. Oper. Res., vol. 7, pp. 285–300, 2000.

[6] S. Luke, Essentials of Metaheuristics, Second Edi. 2013.

[7] B. M. Baker and M. A. Ayechew, “A genetic algorithm for the vehicle routing problem,” Comput. Oper. Res., vol. 30, no. 5, pp. 787–800, 2003.

[8] A. P Engelbrecht, “Computational Intelligence.” p. 597, 2007.

[9] A. Shukla, R. Tiwari, and R. Kala, Real Life Applications of Soft Computing. 2012.

[10] R. Storn and K. Price, “Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces,” J. Glob. Optim., pp. 341–359, 1997.

[11] P. Surekha, Sumathi, and Dr.S., “Solution To Multi-Depot Vehicle Routing Problem Using Genetic Algorithms,” World Appl. Program., vol. 1, no. 3, pp. 118–131, 2011.

[12] N. and E. O. G. University of Málaga, “Multiple Depot VRP with Time Windows Instances,” 2013. [Online]. Available: http://neo.lcc.uma.es/vrp/vrp-instances/multiple-depot-vrp-with-time-windows-instances. [Accessed: 31-Aug-2019].

[13] G. Vrbančič, L. Brezočnik, U. Mlakar, D. Fister, and I. Fister Jr., “NiaPy: Python microframework for building nature-inspired algorithms,” J. Open Source Softw., vol. 3, p. 613, 2018.

[14] X.-S. Yang, “Firefly algorithms for multimodal optimization,” Springer-Verlag Berlin Heidelb., vol. 5792 LNCS, pp. 169–178, 2009.

[15] X.-S. Yang and X. He, “Firefly algorithm: recent advances and applications,” Int. J. Swarm Intell., vol. 1, no. 1, pp. 36–50, 2013.

[16] Mohemmed, A.W., Sahoo, N.C. and Geok, T.K., 2008. Solving shortest path problem using particle swarm optimization. Applied Soft Computing, 8(4), pp.1643-1653.

[17] Ai, T.J. and Kachitvichyanukul, V., 2009. A particle swarm optimization for the vehicle routing problem with simultaneous pickup and delivery. Computers & Operations Research, 36(5), pp.1693-1702.

[18] Yao, B., Yu, B., Hu, P., Gao, J. and Zhang, M., 2016. An improved particle swarm optimization for carton heterogeneous vehicle routing problem with a collection depot. Annals of Operations Research, 242(2), pp.303-320.

[19] Kumar, R.S., Kondapaneni, K., Dixit, V., Goswami, A., Thakur, L.S. and Tiwari, M.K., 2016. Multi-objective modeling of production and pollution routing problem with time window: A self-learning particle swarm optimization approach. Computers & Industrial Engineering, 99, pp.29-40.

[20] Norouzi, N., Sadegh-Amalnick, M. and Tavakkoli-Moghaddam, R., 2017. Modified particle swarm optimization in a time-dependent vehicle routing problem: minimizing fuel consumption. Optimization Letters, 11(1), pp.121-134.

Recognizing the subject exposure from the EEG signals with artificial neural networks

Sašo Pavlič
University of Maribor, Faculty of Electrical Engineering and Computer Science
Koroška cesta 46
Maribor, Slovenia
saso.pavlic@student.um.si

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

ABSTRACT
The paper presents the analysis of Electroencephalography (EEG) brain waves from the Emotiv Insight device with machine learning, more specifically neural networks. The captured EEG data represents the input data into a machine learning model, which was used to determine when and where the required patterns appear. The experiment of the developed method of capturing data and model usage was carried out by exposing the test subject to the alternating selected images and capturing the EEG brain waves with the Emotiv Insight device. The captured EEG data served as a dataset from which the artificial neural network classification model learnt to successfully recognize when a test subject was exposed to one type of image and when to another. Convolutional and recurrent neural network models were constructed and tested to evaluate the performance of recognition of subject exposal.

Keywords
Electroencephalography, Neural Networks, Machine Learning, EEG signals

1. INTRODUCTION
Recently the analysis of Electroencephalography (EEG) data has gained much attention with the development of new measuring techniques and the advancement of the machine learning algorithms and methods. Simpraga et al. [1] proposed a machine learning technique for detection of cholinergic and Alzheimer's disease. Boashash and Ouelha presented a method with machine learning for detection of seizures of newborns [2]. Vanegas et al. presented a machine learning method for detecting of Parkinson's disease [3]. In the same manner, our research was focused on the analysis of EEG data and recognition of subject exposures based on the EEG data with machine learning. Recognizing the simple subject visual exposures can be used in various fields, from user experience, marketing and numerous psychology experiments [4], but there is a lack of research demonstrating the usage of neural networks for this case. This paper intends to fill in this gap.

1.1 Overview of EEG
There are four different EEG frequency bands.

Delta (0.5–3 Hz)

The lowest frequency of brain waves moving below 3 Hz occurs primarily in deep sleep. This frequency is prevalent in infants up to one year of age. It is also present between the 3rd and 4th stages of sleep. Delta waves are reduced in very intense concentration and when we use our thinking processes very actively. Interest is found in individuals who have problems with comprehension and learning. They naturally magnify delta waves; when they want to gather, they fail to reduce it. It is for this reason that the phenomena limit their ability to direct concentration and learning. In this state, we find the brain in a locked-in repetitive state, because in that state we dream or are drowsy.

Theta (3–8 Hz)

Also classified as slower brain activity. The connection can be made with creativity, intuition, daydreaming and fantasy. It also covers memories, emotions and feelings. Theta waves can be expressed through prayer, meditation and spiritual capture. It can be said to occur between waking consciousness and sleeping. When theta wave is optimal, it allows for flexible and complex behavior structures such as learning and remembering. The imbalance of these waves may indicate illness or stress present.

Alfa (8–12 Hz)

Normal alpha status allows for fast and efficient task management. In this condition, most people feel relaxed and calm. You could say that this wave is like a bridge between the conscious and the unconscious. The alpha state is associated with extraversion, creativity (when solving a problem or listening), and having mental work. When the alpha waves are at the optimum range, we experience well-being, see the world positively, and feel a sense of calm. This situation is one of the most important when learning and using information already learned, such as work and education.

Beta (12–38 Hz)

The ripple is typical of "fast" activities. This wave is taken as a normal rhythm and is the dominant wave when the person is collected or upset with the eyes open. Waves also occur in listening, thinking, analytical problem solving, decision making, information processing, etc. Because of its relatively wide range, this wave is divided into low, medium and high beta waves.

Gama (38–42 Hz)

It is a unique frequency wave that is present in all parts of our brains. When they have to process certain information from different parts, it is precisely the 40 Hz frequency that combines the necessary brain regions for simultaneous processing of data. When we remember something well, it's at 40 Hz activity.

2. READING EEG AND ANALYSIS
For recording the brainwaves we have been using BCI Emotiv Insight, which has the excellent API for accessing that data directly from the device using Bluetooth protocol. With the API we managed to get the raw EEG values for each sensor out of five. That data was received in JSON format. Next, to the values from the device, we have been also adding the marker which was the

DOI: https://doi.org/10.26493/978-961-7055-82-5.57-60

indicator for us to know which type of the image was user looking when specific values from the channel were recorded. In the end, our recorded dataset had the following structure:

- ID,
- TIMESTAMP,
- RAW_CQ (raw value of the quality of the signal),
- Values from the electrodes in mV for each location on the head (Af3, T7, P7, T8, Af4) (semanticscholar, 2019),
- EXPOSURE (0 = nature images, 1 = food images).

First, we created the desktop version of the application, which was used as a bridge between the connecting the device with the PC and the presentation media to display the images to the user. Application's basic workflow was that first, we established the connection between the device and the PC, we had to choose the type of image dataset, and the interval of the presentation.

3. IMPLEMENTATING THE EEG ANALYSIS FRAMEWORK

With all that set we run the recording and first displayed to the user the blank screen, just for calibrating the data when a user is having closed/open eyes and watching the monitor.

Figure 1. The experimental setup.

After 30 sec of each, the pictures (Figure 2) from the selected dataset started to switch in the selected interval. Our recordings lasted maximum to 30 minutes, depending on the calmness, relaxation of the user. It was really hard to stay concentrated for a while, just looking at the pictures, without thinking and moving much.

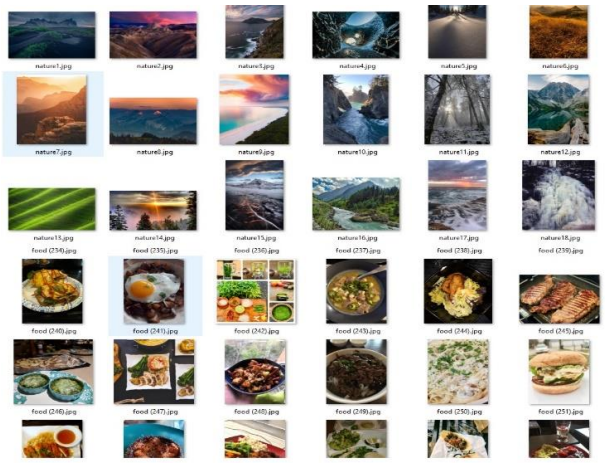


Figure 2. The picture sets for the experiment.

We repeated the recordings on the same user for multiple hours for each type of dataset. Our recording room was isolated from the outside noise and with the constant lighting and minimalism of the objects in the room. We knew that we have to eliminate as much as possible distractions to get reliable recordings from the consumer-based device. All that data was saved in a CSV format per recording, on the interval of 100ms (this is the interval of each read of data made by the device).

ID	Timestamp	Raw_value	Af3	T7	P7	T8	Af4	Marker
0	115	500	4279.487	4250.256	4254.872	4225.641	4291.795	0
1	230	500	4252.821	4209.231	4249.231	4165.641	4246.667	0
2	345	1023	4267.692	4235.897	4192.308	4167.179	4249.231	0
3	460	500	4253.333	4229.744	4276.41	4169.744	4240.513	0
4	575	1023	4262.564	4203.077	4183.077	4165.641	4246.154	0

Table 1. Collected dataset of EEG signals from Emotive Insight device.

It was a mixture of the recorded values from different datasets. Our entire dataset with the length about 50.000 rows. This was the result of almost 2 hours of recording the brainwaves. We split the list into two groups for having the data for testing and learning process in ML. The ratio was 80:20 for the learning process. With all that ready we continued our work, with creating the artificial neural network model.

3.1 Analysis with Neural Network

Python framework Keras helped us to create the following neural model. The type of model we have created is called the sequential model and allows you to create multiple levels one by one. It is limited in that it does not allow you to create models that share layers or have multiple inputs or outputs

In the input level it is required to define the dimension of the data in the first level, because at the beginning the model cannot know what data will come to the input. Input data in our case were the values from the electrodes (Af3, T7, P7, T8, Af4).

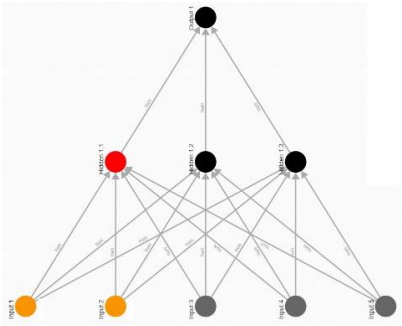


Figure 3. Construction of neural network model with three layers.

Normal distribution was used to initialize the weights, which initialized the weights according to the function results. The activation parameter was defined with a well-established and relatively simple Relu function in the input and hidden levels of the model. For the output level, however, we had to define a sigmoid function that allows us to get a result between 0 and 1.

We also added a so-called level dropout among the individual levels, which serves to ensure that randomly selected neurons are omitted during the learning phase. This means that the results of the forward pass activation function are removed, as well as any weight

change is not applied to the backward pass. All these procedures help to make the model generalizable.

When we had finished defining the model, we still had to compile the model with a function that would define a loss (optimizer) that would be able to update the weights according to the result and a function that would return us the result, with what precision in percent, the model predicted the result with given parameters (weights and bias).

```
model.compile(loss="binary_crossentropy",
              optimizer="adam",
              metrics=['accuracy'])
```

Algorithm 2. Compilation of neural network model.

In the next step, we still had to set the parameters for how the model would learn.

```
model.fit(x_train,
          y_train,
          batch_size=124,
          epochs=1000,
          validation_data=(x_test, y_test))
```

Algorithm 3. Fitting of the neural network model.

As the first two parameters, we sent the data to be provided for learning along with the results, and we also defined the size of the data packets in one epoch. In practice, this meant more or less that we repeatedly started learning with different values of the size of the batch and epoch.

To determine how well our neural model performed, we used the evaluate function, which returns the loss and metric values for the model under test.

We obtained a result from the results, which showed that our model predicted the result to reach an accuracy of 55%, which was not an impressive result.

3.2 Analysis with Recurrent Neural Network

We can more easily imagine this type of neural model by thinking about how our thoughts work, they always relate to our previous thoughts, we could say that we never start thinking from scratch. This is also how the recurrent-type neural model (RNN) works, which does not restart at every iteration. Traditional NNs do not do this, which is a disadvantage.

Our brainwaves data read on a given image is like a list in which at a given moment, through different positions (Af3, T7, P7, T8, Af4), we can find out which image was shown to the user. However, these values cannot provide us with high reliability of the result, as there were significantly too many factors present when recording data such as electrode reliability, deconcentrating of the user, sudden movements of the user's head, physiological processes in the body, etc. For these reasons, it might be better to want to get that search result across multiple data records, because within those records we can define a pattern in our data that could give us a more reliable result. Because even in the presence of external factors that interfere with our data reliability, through a large amount of records these values are limited to a given stock of values.

We also used the Keras API to build the RNN model, using the `SimpleRNN` layer, which is a fully connected RNN, where the output from the previous step is sent back to the input for a new learning step. The RNN model accepts a 3-dimensional input type for the input, and our data is in the 2-dimensional type, the third data will represent the step here. Data transformation was achieved by the numpy.reshape method.

Once we had the data in the required type and form, we started building the RNN model. We used again as a input data values from the electrodes (Af3, T7, P7, T8, Af4).

For which we defined the model type as sequential. We added a `SimpleRNN` level with 32 dimensions and two `Dense` levels to the model, the first with 8 dimensions and the output with one. For simplicity, we used activation function Relu.

Finally, when we built the model, we further defined the optimization function as `RmsProp` which is designed for RNN models and the loss function as the average squared error.

```
model = Sequential()
model.add(SimpleRNN(units=32,
                    input_shape=(1, 5),
                    activation="relu"))
model.add(Dense(1))
model.compile(loss='mse',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

Algorithm 6. Construction of recurrent neural network model.

After learning the model, we got a good start (approx. 62% accuracy), which was a good starting point for optimizing our model. We have added another Dense level to our model, which should help with the intermediate learning steps, since we want to get better results with the RNN model than with the previous NN, but we need to know that there are many more factors (parameters, hidden levels, understanding the flow of data) that make it difficult to understand and refine the model.

In the end, we obtained an accuracy that defines the accuracy of the result of 80%, which in our case is a satisfactory result, but in practice, unfortunately, it would not be enough to use the model for commercial purposes or for research. Therefore, we decided to change our model by removing the second last level with eight neurons, as it improved the learning result compared to the previous model but still did not succeed in getting a better result. What happened here was that the model became too complex and because of this, the activation functions failed, leading the learning phase to better weight adjustments that would give better learning results.

In the end, our model looked the same as it did at the beginning of defining the model. Here, we then started from the beginning and, before `SimpleRNN`, inserted a new entry level (`Embeding`), which transforms positive data with numbers into dense vectors, for example:

`[[4], [20]] → [[0.25, 0.1], [0.6, - 0.2]]`

This level therefore does not require 3D data at the beginning, so we also removed the line from our code for converting 2D data into 3D. The output type from `Embeding` layer is in 3D, which suited us in passing the data to our `SimpleRNN` layer.

```
model = Sequential()
model.add(Embedding(10000, 124))
model.add(SimpleRNN(32))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss = "mse",
              optimizer = "rmsprop",
              metrics=['accuracy'])
```

Algorithm 8. The final RNN model.

4. RESULTS OF THE EXPERIMENT AND CONCLUSION

In our scenario, we had a collection of data that represented the read values from the BCI device per channel and a marker to indicate

which image the user observed while reading the values. So we had 6 properties that we split into two lists. The first with values from the device and the second with the marker that represented the result. All of this information was further divided into a learning and test list.

We used two different machine learning models in our process. With the classic NN model, we got worse results because it was obvious that we had too little data despite recording data for hours on the user. Also for this reason NN performed worse because it only changes weights at the end of learning. Meanwhile, the RNN changed weights during the learning step itself, eventually making significantly better predictions, despite the small amount of learning data, as it used internal memory to use the result from the previous output to improve the new one.

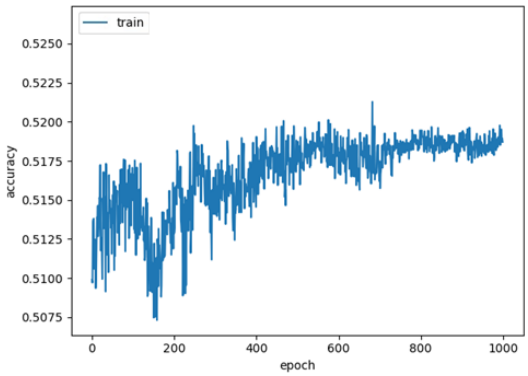


Figure 4. Accuracy through the epochs of the training process for the first simple NN model.

Figure 4 shows how the first simple NN model learned over training process. We can see that the model improved its accuracy in the first half of learning, but towards the end more or less came closer to the same values. If we significantly increased the number of epochs, the result did not improve over time.

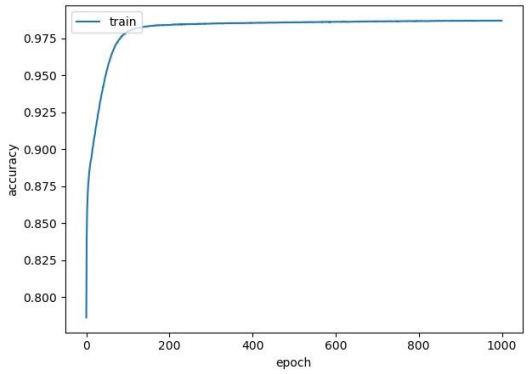


Figure 5. Accuracy through the epochs of the training process for the RNN model.

However, when learning the RNN model (Figure 5), we can see that the accuracy of the model has increased dramatically from the beginning of learning. This change was aided by a new type of data in which the learning process then had a better ability to adjust weights during learning, as well as the RNN model sending the learned state from the previous output to the input, which contributes to the "current" learning. Due to all the running factors, this model performed great compared to a regular NN.

We can also see the loss in the Figure 6 as it declined over time, which meant that our model was getting better at telling the result or the user was looking at pictures of nature or food.

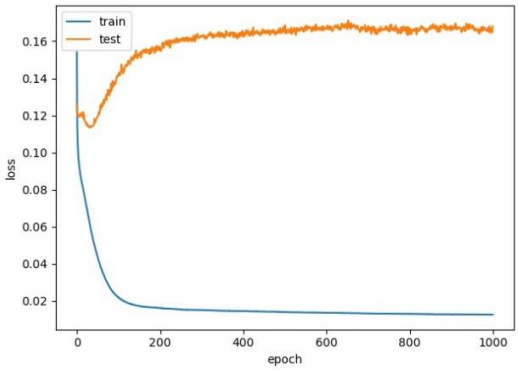


Figure 6. The loss the epochs of the training process for the RNN model.

All these results were still highly dependent on all the external factors that were present in the initial data capture. Initially, the quality and precision of the BCI device must be considered, as it is intended for commercial users and not for professional use, then we must know that user recording is very demanding, and a special gel must be present on the device to help increase electrode conductivity to capture electromagnetic waves, the electrodes should be also as close to the scalp as possible and with as large surface as possible. Another factor is, the mood of the user himself when scanning whether he was always asleep, steady, relaxed, focused are all arguments that should always be the same. All these factors influence the quality of the data and the later classification with machine learning.

5. ACKNOWLEDGMENTS

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

6. REFERENCES

[1] Simpraga, S., Alvarez-Jimenez, R., Mansvelder, H.D., Van Gerven, J.M., Groeneveld, G.J., Poil, S.S. and Linkenkaer-Hansen, K., 2017. EEG machine learning for accurate detection of cholinergic intervention and Alzheimer's disease. Scientific reports, 7(1), p.5775.

[2] Boashash, B. and Ouelha, S., 2016. Automatic signal abnormality detection using time-frequency features and machine learning: A newborn EEG seizure case study. Knowledge-Based Systems, 106, pp.38-50.

[3] Vanegas, M.I., Ghilardi, M.F., Kelly, S.P. and Blangero, A., 2018, December. Machine learning for EEG-based biomarkers in Parkinson's disease. In 2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM) (pp. 2661-2665). IEEE.

[4] Subha, D.P., Joseph, P.K., Acharya, R. and Lim, C.M., 2010. EEG signal analysis: a survey. Journal of medical systems, 34(2), pp.195-212.

Transfer Learning Tuning Utilizing Grey Wolf Optimizer for Identification of Brain Hemorrhage from Head CT Images

Grega Vrbančič
University of Maribor, Faculty of Electrical Engineering and Computer Science
Koroška cesta 46
SI-2000 Maribor, Slovenia
grega.vrbancic@um.si

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

Vili Podgorelec
University of Maribor, Faculty of Electrical Engineering and Computer Science
Koroška cesta 46
SI-2000 Maribor, Slovenia
vili.podgorelec@um.si

ABSTRACT

Most commonly, diagnosing the brain hemorrhage - a condition caused by a brain artery busting and causing bleeding is done by medical experts identifying such pathologies from the computer tomography (CT) images. With great advancements in the domain of deep learning, utilizing deep convolutional neural networks (CNN) for such tasks has already proven to achieve encouraging results. One of the major problems of using such an approach is the need for big labeled datasets to train such deep architectures. One of the efficient techniques for training CNNs with smaller datasets is transfer learning. For the efficient use of transfer learning, many parameters are needed to be set, which are having a great impact on the classification performance of the CNN. Most of those parameters are commonly set based on our previous experience or by trial and error. The proposed method addresses the problem of tuning the transfer learning technique utilizing the nature-inspired, population-based metaheuristic Grey Wolf Optimizer (GWO). The proposed method was tested on a small head CT medical imaging dataset. The results obtained from the conducted experiments show that the proposed method outperforms the conventional approach of parameter settings for transfer learning.

Keywords

Convolutional Neural Network, Transfer Learning, Optimization, Biomedical images, Classification

1. INTRODUCTION

Most commonly used medical imaging technique to assess the severity of brain hemorrhage, also termed as a cerebral hemorrhage, intracranial hemorrhage or intracerebral hemorrhage is the computer tomography or shortly CT. As reported in [24], each year intracerebral hemorrhage (ICH) affects 2.5 per 10,000 people worldwide and is associated with high mortality that only 38% of ICH patients could survive over one year. Besides, more than 80% of people are suffering due to being born with a weak spot in their major brain arteries. However, the early diagnosis of the condition and receiving immediate and relevant treatment can be a lifesaver for the affected patient. Traditionally, the tools helping in diagnosing such conditions are CT images obtained from the CT scan, which are then examined by the expert such as an experienced doctor, who has the ability to identify important symptoms of the disease from the image by a naked eye [3].

With the expansion of deep learning field and with the great achievements of deep convolutional neural networks (CNN) for the image and video recognition tasks [26, 27] are such approaches and methodologies also being used for addressing various medical areas such as medical image analysis [1] and classification [31, 12], biomedical signal segmentation [23] and detection of various human organ activities [30].

In recent studies [12, 4, 11], the authors have already addressed the problem of identifying various kinds of brain hemorrhages utilizing different kinds of more or less complex deep CNNs. However, the problem with the training of such deep CNN architectures remains the same. In order to achieve acceptable performance, the training of such networks requires a lot of resources in terms of time and processing power. Additionally, a big dataset of images, hand-labeled by experts is also required. Given the fact that such high-quality big datasets of biomedical images are hard to obtain, researchers are trying various approaches and techniques to overcome this problem. One of the most popular techniques for training deep CNNs on small datasets is transfer learning, which has already proven to achieve great results [4, 14]. But the transfer learning techniques also comes with the downsides. Most commonly, the biggest problems when utilizing the transfer learning approaches are finding out which and how many layers to fine-tune and how to set the training parameters for the fine-tuning of the CNN in order to obtain the acceptable outcome.

Based on the encouraging results of transfer learning technique being used to train CNNs for the task of classification of biomedical images and our previous experience on optimizing various training parameters [32], we set our goal to develop a method for an automatic optimization of transfer learning utilizing nature-inspired population-based Grey Wolf Optimizer (GWO) algorithm named GWOTLT.

The rest of the paper is organized as follows. In Section 2, we briefly describe methods which were used. In Section 3, we present the proposed GWOTLT method, while in Section 4 we describe the experimental setup of conducted experiments, the results of which are presented in Section 5. Conclusions and final remarks are gathered in Section 6.

DOI: https://doi.org/10.26493/978-961-7055-82-5.61-66

2. METHODS

In this section, the methods utilized in our proposed GWOTLT method are briefly presented.

2.1 Convolutional Neural Network

In the 1980s, the CNNs were first presented in Fukushima's paper [10]. The author proposed a deep learning approach for visual recognition, called neocognitron, which was based on the hierarchical layers trained with the utilization of the stochastic gradient descent algorithm. The major breakthrough with CNNs occurred in 1998 with the LeCun's LeNet5 [17] proposed architecture which is considered to be one of the key factors that started the enormous expansion of the deep learning field.

Initially, the deep CNNs were defined as 2-dimensional constrained neural networks with alternating convolutional and subsampling or pooling layers which are fully connected at the end, combining three architectural ideas [17]:

- local receptive fields,
- shared weights, and
- spatial and temporal subsampling.

Most commonly the convolutional layer is composed of several so-called feature maps. Those feature maps are calculated with different weight vectors, which enable us to extract multiple features from each location. The results of the convolutional calculation are obtained from a convolutional operation performed between feature maps of the previous layer and convolution kernel of the current layer in addition to the activation function. A subsampling layer or pooling layer reduces the dimension of feature maps, while preserving the important extracted features, usually performing local averaging and subsampling. The fact, that extracted features' real locations are not important as long as their approximate positions relative to others remain the same, is making subsampling possible [17].

Although the researchers have through the years developed various complex CNN architectures which proven to be highly successful in the large-scale image and video recognition such as Krizhevsky's AlexNet [15], Szegedy's GoogleNet [27] and Simonyan's VGG16 [26], the challenges regarding image and video recognition still exist. Such major challenges are primarily the need for large datasets in order to train the CNNs and the time complexity of the training process.

2.2 Transfer Learning

One of the most popular approaches to address the time complexity of deep CNN training process as well as the problem of not having large dataset is known as a transfer learning. Transfer learning can be defined as the improvement of learning a new task through the transfer of knowledge from a related task that has already been learned. In machine learning terms, the transfer learning roughly translates to transferring the weights of already trained deep neural network model for one task, to the model tackling second related task [13]. Based on previous work [16, 2, 25], such approaches work especially well if we have a small, insufficient dataset.

Transfer learning is most commonly used in two ways [2, 21]:

- Fine-tuning in which the weights of the pre-trained CNN base model are preserved (frozen) on some of the layers and fine-tuned (trained) in remaining layers of CNN.
- CNN as a feature extractor, where the general idea is to access features of any layers and using those encoded features to train a classifier of your choice.

Generally, the first (top) layers of the CNN preserve more abstract, generic features applicable to other tasks, while the layers closer to the bottom provide more specific features that can benefit from fine-tuning as they will be adjusted specifically for the targeted task. For the fine-tuning approach to transfer learning, there is no general recipe or rule to follow on selecting which layers to tune and which ones to preserve as they are. Also, another challenge utilizing the fine-tuning approach is deciding how many layers to add to the bottom of the pre-trained convolutional base, and which optimizer and learning rate to use in the process of fine-tuning.

2.3 Grey Wolf Optimizer

In recent years, swarm intelligence and bio-inspired algorithms for solving the optimization problems are quite popular and proven to be very efficient in solving real-world problems [9].

One of the most popular representatives of such optimization algorithms is a Grey Wolf Optimizer or simply GWO [19]. The inspiration of GWO is adapted from a strict leadership hierarchy and hunting mechanisms of grey wolfs (Canis lupus). The grey wolf leadership hierarchy is divided into four dominance groups, i.e. alpha, beta, delta and, omega. Besides the leadership hierarchy, group hunting is also an interesting social behavior of grey wolfs. As defined by authors in [20] main phases of grey wolf hunting are as follows [19]:

- Tracking, chasing and approaching the prey.
- Pursuing, encircling, and harassing the prey until it stops moving.
- Attack towards the prey.

The GWO algorithm implementation is mathematically modeling the mentioned hunting technique and the social hierarchy in order to perform optimization. The basic pseudo-code of GWO algorithm is presented in Algorithm 1.

3. PROPOSED METHOD

The basic concept of our proposed method for tuning of transfer learning approach based on the GWO algorithm, named as GWOTLT is presented in Figure 1. The GWO algorithm is used to find the optimal parameters for the

Algorithm 1 Pseudo code of the GWO algorithm.

```
Initialize the grey wolf population X_i (i = 1, 2, ..., n)
Initialize a, A, and C
Calculate the fitness of each search agent
X_α = the best search agent
X_β = the second best search agent
X_δ = the third best search agent
while i < Maximum iterations do
    for each search agent do
        Update the position of the current search agent
    end for
    Update a, A, and C
    Calculate the fitness of all search agents
    Update X_α, X_β, and X_δ
    i = i + 1
end while
return X_α
```

fine-tuning transfer learning process. In our case, the goal is to find a number of neurons in the last fully connected layer, dropout probability of dropout layer and the most suitable optimizer and learning rate value.

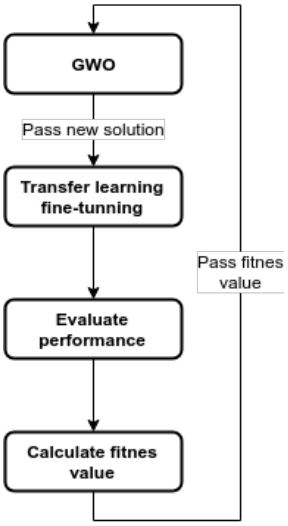


Figure 1: The conceptual diagram of the proposed GWOTLT method.

Given the number of optimized parameters for fine-tuning of the transfer learning process, the GWOTLT is producing the solution with the dimension of 4. The individuals of GWOTLT produced solutions are presented as real-valued vectors:

$$\mathbf{x}_i^{(t)} = (x_{i,0}^{(t)}, \ldots, x_{i,n}^{(t)}), \quad \text{for } i = 0, \ldots, Np - 1, \qquad (1)$$

where each element of the solution is in the interval $x_{i,1}^{(t)} \in [0, 1]$.

In next step, the real-valued vectors (solutions) are mapped as defined in equations 2, 3, 4 and 5, where y_1 presents the number of neurons in last fully connected layer, y_2 dropout probability, y_3 optimization function and y_4 learning rate. Each y_1 value is mapped to the particular member of the population $N = \{64, 128, 256, 512, 1024\}$ according to the members position in the population, which represents a group of available numbers of neurons in last fully connected layer. All of the y_3 values are mapped to the specific member of population $O = \{adam, rmsprop, sgd\}$, which represents a group of available optimizer functions, while each y_4 values are mapped to the member of population $L = \{0.001, 0.0005, 0.0001, 0.00005, 0.00001\}$, which represents a group of learning rate choices.

$$y_1 = \begin{cases} \lfloor x[i] * 5 + 1 \rfloor; y_1 \in [1, 5] & x[i] < 1 \\ 5 & \text{otherwise,} \end{cases} \qquad (2)$$

$$y_2 = x[i] * (0.9 - 0.5) + 0.5; y_2 \in [0.5, 0.9] \qquad (3)$$

$$y_3 = \begin{cases} \lfloor x[i] * 3 + 1 \rfloor; y_3 \in [1, 3] & x[i] < 1 \\ 3 & \text{otherwise,} \end{cases} \qquad (4)$$

$$y_4 = \begin{cases} \lfloor x[i] * 5 + 1 \rfloor; y_4 \in [1, 5] & x[i] < 1 \\ 5 & \text{otherwise,} \end{cases} \qquad (5)$$

To evaluate each solution produced by GWOTLT the fitness function was defined as follows:

$$f(x) = 1 - AUC(x) \qquad (6)$$

where $f(x)$ is the fitness value for solution x and the $AUC(x)$ is an area under the ROC curve calculated on test split of the search dataset sub-sample.

4. EXPERIMENT SETUP

To evaluate the performance of our proposed method, we conducted two experiments. The experimental settings, dataset, evaluation methods and metrics used are in-depth presented in the following subsections.

The proposed method was implemented in Python programming language with the following external libraries: Numpy [28], Pandas [18], scikit-learn [22], NiaPy [29], Keras [5] and Tensorflow [7].

All of the conducted experiments were performed using the Intel Core i7-6700K quad-core CPU running at 4 GHz, 64GB of RAM, and three Nvidia GeForce Titan X Pascal GPUs each with dedicated 12GB of GDDR5 memory, running the Linux Mint 19 operating system.

4.1 Dataset

Given the task - identification of brain hemorrhage from CT images, we used a publicly available dataset of manually

collected head CT scan images called Head CT - hemorrhage [8]. The dataset contains in total of 200 images of various sizes. Of those 200 images, half of them are images of normal head CT slides without any brain pathologies, and the other half are the images containing some kind of brain hemorrhage. Also, each image is collected from a different person.

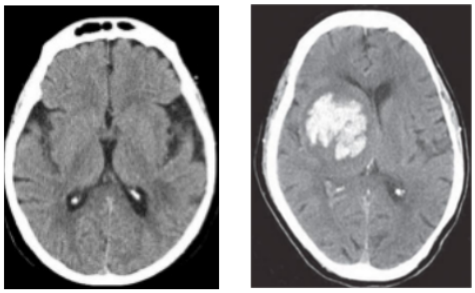


Figure 2: Example images of head CT scans, where a) represents normal head CT scan image. while b) represents the head CT scan image with brain hemorrhage present.

4.2 Grey Wolf Optimizer settings

To initialize the GWO algorithm, tackling the problem of finding the best suitable set of parameters to achieve the best performance of transfer learning fine-tuning, the GWO parameter settings presented in Table 1 were used.

Parameter	Value
Dimension of the problem	4
Population size	10
Number of function evaluations	50
Lower bound	0.0
Upper bound	1.0

Table 1: The initial GWO parameter settings.

4.3 Baseline Convolutional Neural Network

For the convolutional base of our proposed method, we utilized the VGG16 [26] CNN architecture presented in Figure 3, pre-trained on the imagenet [6] dataset. As we can observe from the figure, the VGG16 CNN is comprised of 5 convolutional blocks, which together form a convolutional base. At the bottom of the convolutional base a flatten layer, two fully connected layers and one fully-connected layer with softmax activation function forming a classifier layer are chained. By default, VGG16 CNN on the input receives an image of size 224 x 224 pixels and at the bottom classifies fed images into 1000 classes, while each of the convolutional layers of VGG architecture utilizes the ReLU activation function.

Performing the transfer learning based on the VGG16 CNN convolutional base, we have persisted the top four convolutional blocks and enabled for fine-tuning only last convolutional block. At the bottom of this convolutional base, we have then chained a flatten layer, a dropout layer, fully connected layer and classifier with softmax activation function, classifying images into two target classes - images with and images without brain hemorrhage present.

For the baseline experiments, we have set the parameters which are we optimizing to the values presented in Table 2.

Parameter	Value
Number of neurons on the last fully connected layer	256
Dropout probability	0.5
Optimizer function	RMSprop
Learning rate	10^{-5}

Table 2: Baseline experiment parameter settings for transfer learning fine tuning.

With the presented parameter settings, we trained the CNN for 50 epochs utilizing an efficient mini-batch training, with batch size set to 32. As presented in the dataset section, the collected image sizes vary from 100 x 100 px to 300 x 300 px, thus we have decided to resize all images to the VGG16 default input size of 224 x 224 px.

4.4 GWOTLT settings

As presented in the previous section, the GWOTLT fine-tuning transfer learning parameters are set based on the produced GWO solution. The overall architecture of the convolutional base and the appended classification layers at the bottom are the same as in the baseline experiment. Due to the iterative nature of our proposed method, we had to split the given train set in ratio 80:20, where we used the larger subset for training different GWOTLT produced solutions and evaluating them - calculating the AUC on the remaining smaller subset of the initial training set. In each run of the GWOTLT, 50 evaluations of produced possible solutions are conducted, from which the best - the one with the highest fitness value is selected. To evaluate each solution, we train each solution for 10 epochs and then evaluate its performance. The selected solution is then trained for full 50 epochs on the whole given train dataset and finally evaluated on the given test set.

4.5 Evaluation method and metrics

Using the described experimental setup, we conducted two experiments, one using the CNN transfer learning approach without any optimization reported as a *Baseline* and one utilizing the presented GWOTLT method reported as *GWOTLT*. For each of the experiments, we obtained six performance metrics: time - reported in seconds, AUC, $F-1$ score, precision, and recall, reported in percents and kappa coefficient presented as a real value on the interval between 0 and 1.

To objectively evaluate the performance of the proposed method, we adapted the gold standard 10-fold cross-validation methodology, where a dataset is divided into train and test sets in a ratio 90:10. Using the images from 9 out of 10 folds for the training and performing the performance evaluation on the remaining one fold. In the same manner, we repeated the whole process in total 10 times, each time leaving different fold out for the performance evaluation. The reported values are presented as average values over 10 folds if not specified otherwise.

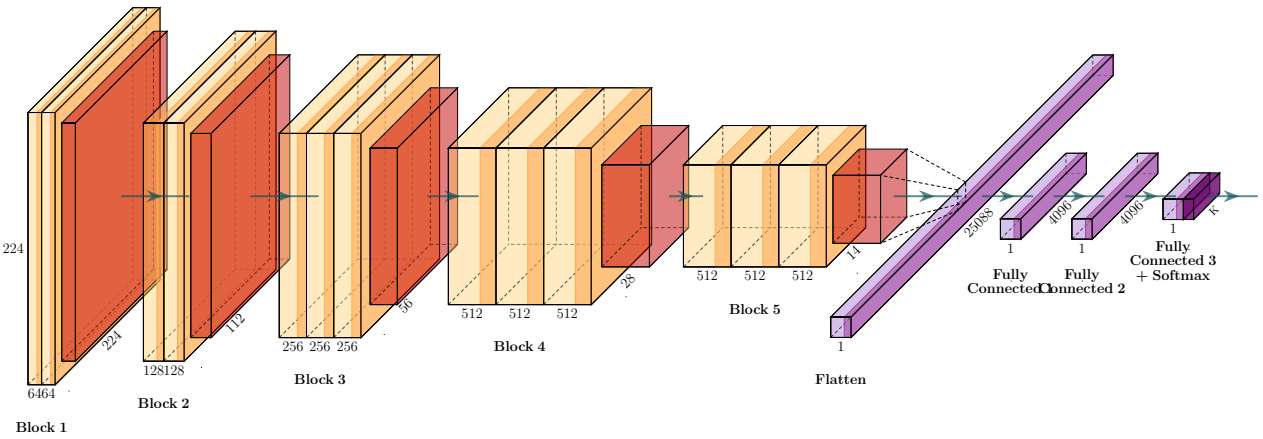


Figure 3: The architecture of the VGG16 convolutional neural network.

5. RESULTS

The obtained performance results from the conducted experiments are summarized in Table 3. Focusing on the time metrics, the reported results are expected, with the lowest time complexity being achieved by the *Baseline* method. On the other side, the proposed *GWOTLT* method is expected to have a higher time complexity in general due to the iterative nature of the proposed method. In our case, the *GWOTLT* method performed worse in the aspect of time complexity, roughly by a factor 15.

Analyzing presented classification performance metrics, the *GWOTLT* method is standing out with achieved best results on all of the reported performance metrics. The AUC, $F-1$, precision and recall metrics are higher by a margin of 4%, 5.18%, 2.27%, 7% respectively in comparison to the baseline method. Focusing on the kappa coefficient values, we can observe that the *GWOTLT* achieved a near-perfect agreement with kappa coefficient at 0.82 and outperformed the baseline method by a margin of 0.08. Looking at the standard deviations of the reported classification average metric values, we can observe that for all classification metrics, except for the precision, the best performing *GWOTLT* method is showing the smallest standard deviation. The greatest improvement of lowering the standard deviation the *GWOTLT* achieved for the recall metric by a margin of 10.38%, while the worst standard deviation is obtained for the precision metric where the *GWOTLT* lacks behind just by 0.99%.

Metrics	*Baseline*	*GWOTLT*
Time [s]	**49.10 ± 1.85**	759.10 ± 59.67
AUC [%]	87.00 ± 9.19	**91.00 ± 7.75**
$F-1$ [%]	86.27 ± 11.03	**91.45 ± 6.81**
Precision [%]	88.62 ± **10.37**	**90.89** ± 11.36
Recall [%]	86.00 ± 17.13	**93.00 ± 6.75**
Kappa	0.74 ± 0.18	**0.82 ± 0.15**

Table 3: Comparison of average times, accuracies, AUCs, $F-1$ scores, precisions, recalls and kappa coefficients with standard deviations over 10-fold cross-validation.

6. CONCLUSIONS

In this paper, we presented the GWOTLT method which is a nature-inspired, population-based metaheuristics method for tuning the transfer learning approach of training the deep CNN. The GWOTLT method was implemented utilizing the GWO optimization algorithm and applied to the problem of identification of brain hemorrhage from the head CT scan images. The results obtained from the conducted experiments have proven that the proposed GWOTLT method seems to be very promising for the task of transfer learning tuning achieving higher classification performance for all of the measured classification metrics.

In the future, we would like to expand our work to include various CNN architectures as a convolutional base for our GWOTLT method and also evaluate the performance of the proposed method against various medical imaging datasets.

Acknowledgments

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

7. REFERENCES

[1] S. U. Akram, J. Kannala, L. Eklund, and J. Heikkilä. Cell segmentation proposal network for microscopy image analysis. In *Deep Learning and Data Labeling for Medical Applications*, pages 21–29. Springer, 2016.

[2] E. Al Hadhrami, M. Al Mufti, B. Taha, and N. Werghi. Transfer learning with convolutional neural networks for moving target classification with micro-doppler radar spectrograms. In *2018 International Conference on Artificial Intelligence and Big Data (ICAIBD)*, pages 148–154. IEEE, 2018.

[3] U. Balasooriya and M. Perera. Intelligent brain hemorrhage diagnosis system. In *2011 IEEE International Symposium on IT in Medicine and Education*, volume 2, pages 366–370. IEEE, 2011.

[4] P. Chang, E. Kuoy, J. Grinband, B. Weinberg, M. Thompson, R. Homo, J. Chen, H. Abcede,

M. Shafie, L. Sugrue, et al. Hybrid 3d/2d convolutional neural network for hemorrhage evaluation on head ct. *American Journal of Neuroradiology*, 39(9):1609–1616, 2018.

[5] F. Chollet et al. Keras, 2015.

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[7] M. A. et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[8] Felipe Kitamura. Head CT - hemorrhage, 2018. Available at https://www.kaggle.com/felipekitamura/head-ct-hemorrhage, Accessed: 2019-02-21.

[9] I. Fister Jr, X.-S. Yang, I. Fister, J. Brest, and D. Fister. A brief review of nature-inspired algorithms for optimization. *arXiv preprint arXiv:1307.4186*, 2013.

[10] K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position, BioL Cybem. 36 (1980) 193-202. *S. Shiotani et al./Neurocomputing 9 (1995) Ill-130*, 130, 1980.

[11] M. Grewal, M. M. Srivastava, P. Kumar, and S. Varadarajan. Radnet: Radiologist level accuracy using deep learning for hemorrhage detection in ct scans. In *2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018)*, pages 281–284. IEEE, 2018.

[12] A. Helwan, G. El-Fakhri, H. Sasani, and D. Uzun Ozsahin. Deep networks in identifying ct brain hemorrhage. *Journal of Intelligent & Fuzzy Systems*, (Preprint):1–1, 2018.

[13] M. Hussain, J. J. Bird, and D. R. Faria. A study on cnn transfer learning for image classification. In *UK Workshop on Computational Intelligence*, pages 191–202. Springer, 2018.

[14] K. Jnawali, M. R. Arbabshirani, N. Rao, and A. A. Patel. Deep 3d convolution neural network for ct brain hemorrhage classification. In *Medical Imaging 2018: Computer-Aided Diagnosis*, volume 10575, page 105751C. International Society for Optics and Photonics, 2018.

[15] A. Krizhevsky. One weird trick for parallelizing convolutional neural networks. *arXiv preprint arXiv:1404.5997*, 2014.

[16] D. Larsen-Freeman. Transfer of learning transformed. *Language Learning*, 63:107–129, 2013.

[17] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[18] W. McKinney. Data structures for statistical computing in python. In S. van der Walt and J. Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 51 – 56, 2010.

[19] S. Mirjalili, S. M. Mirjalili, and A. Lewis. Grey wolf optimizer. *Advances in engineering software*, 69:46–61, 2014.

[20] C. Muro, R. Escobedo, L. Spector, and R. Coppinger. Wolf-pack (canis lupus) hunting strategies emerge from simple rules in computational simulations. *Behavioural processes*, 88(3):192–197, 2011.

[21] K. Nogueira, O. A. Penatti, and J. A. dos Santos. Towards better exploiting convolutional neural networks for remote sensing scene classification. *Pattern Recognition*, 61:539–556, 2017.

[22] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[23] R. Rouhi, M. Jafari, S. Kasaei, and P. Keshavarzian. Benign and malignant breast tumors classification based on region growing and cnn segmentation. *Expert Systems with Applications*, 42(3):990–1002, 2015.

[24] L. Shi, S. Xu, J. Zheng, J. Xu, and J. Zhang. Blood Pressure Management for Acute Intracerebral Hemorrhage: A Meta-Analysis. *Scientific Reports*, 7(1):14345, 2017.

[25] H.-C. Shin, H. R. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues, J. Yao, D. Mollura, and R. M. Summers. Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning. *IEEE transactions on medical imaging*, 35(5):1285–1298, 2016.

[26] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[27] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[28] S. Van Der Walt, S. C. Colbert, and G. Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22, 2011.

[29] G. Vrbančič, L. Brezočnik, U. Mlakar, D. Fister, and I. Fister Jr. NiaPy: Python microframework for building nature-inspired algorithms. *Journal of Open Source Software*, 3, 2018.

[30] G. Vrbancic, I. J. Fister, and V. Podgorelec. Automatic Detection of Heartbeats in Heart Sound Signals Using Deep Convolutional Neural Networks. *Elektronika ir Elektrotechnika*, 25(3):71–76, jun 2019.

[31] G. Vrbancic and V. Podgorelec. Automatic Classification of Motor Impairment Neural Disorders from EEG Signals Using Deep Convolutional Neural Networks. *Elektronika ir Elektrotechnika*, 24(4):3–7, aug 2018.

[32] G. Vrbančič, I. Fister, Jr., and V. Podgorelec. Swarm intelligence approaches for parameter setting of deep learning neural network: Case study on phishing websites classification. In *Proceedings of the 8th International Conference on Web Intelligence, Mining and Semantics*, WIMS '18, pages 9:1–9:8, New York, NY, USA, 2018. ACM.

System for remote configuration and over the air updates in restricted environments

Marko Zabreznik
University of Maribor
Faculty of Electrical Engineering
and Computer Science
Koroška cesta 46, Maribor
marko.zabreznik@student.um.si

Jernej Kranjec
University of Maribor
Faculty of Electrical Engineering
and Computer Science
Koroška cesta 46, Maribor
jernej.kranjec@um.si

ABSTRACT
This paper illustrates a system for configuring, updating, command execution, and data retrieval via limited communication links of restricted embedded real-time operating system. A custom domain-specific language design and reference implementation are proposed, which would simplify the creation of custom program tasks while keeping data transfers low and facilitating differential updates to any software component. The proposed implementation extends the FreeRTOS real-time operating system running on an ARM-based microcontroller and connects to a remote command server via a limited network connection. External factors such as power shortage, component failure, and connection loss are anticipated and handled by preset priority-based scenarios.

Keywords
real time operating system, remote control, domain specific language for remote task execution, data structures, remote sensing

1. INTRODUCTION
In recent years, the cost and accessibility of world-wide communication channels, low cost of sensor equipment, and accessible computer modules have given researchers new sources of data acquisition. Systems used for such applications need to operate autonomously, are remote or completely inaccessible while also potentially under limited power, intermittent network connection, harsh or unpredictable weather, and other environmental hazards. Depending on those conditions, the objective or priorities might change during the lifetime of the mission.

The goal of this paper is to introduce a solution for embedded systems that require executing multiple different tasks (e.g., collecting data from various sensors, data processing, information storage and transmission), autonomous control over task execution based on external parameters (e.g., available power or sensor activity), remote configuration, and software updates. Presented solution would provide for more straightforward creation of such systems as it would allow for modular hardware components to be assembled into various configurations while only require to produce missing software from module templates, the behavior of which is controlled by a known set of parameters. To accomplish this, we propose a domain-specific language extension for a widely supported real-time operating system FreeRTOS to define scenarios that have associated tasks, conditions, and priorities. The same scenario definition is also used to partition the software into blocks, allowing for over-the-air updates and the addition of new scenarios remotely. Furthermore, the scenario definition is also used to prioritize the limited connection and system resources, and to allow for direct control.

2. PROPOSED DESIGN
The proposed design incorporates many dynamic components and thus vulnerable to corrupt, misconfigured, or buggy software. For that reason, the software is divided into a safe, minimal system (Bootstrap), and the schedule based configurable (Operating System) with the Meta-Scheduler (see Figure 1).

Both the Bootstrap and the Operating System can understand the basic programming instructions that work in all cases. Once an error in execution or corruption of the software is detected and can not be recovered from, the microcontroller reboots into the safe Bootstrap mode sends out a distress signal and waits for commands.

2.1 Scenario
A Scenario (see Figure 1) is the basic unit of the design that is independent of any other generic task, carries its configuration, and takes up any predefined blocks on the flash. The scenario can define a schedule and period of execution, the priority it needs, and the power it requires. We use those parameters to decide when and if the task is to run. The parameters are defined using a domain-specific language at compile-time but can change in the runtime with configuration stored in the microcontroller flash for each scenario.

2.1.1 Settings
The settings parameter defines the size of the binary blob the scenario can use for configuration data and is generally

DOI: https://doi.org/10.26493/978-961-7055-82-5.67-70

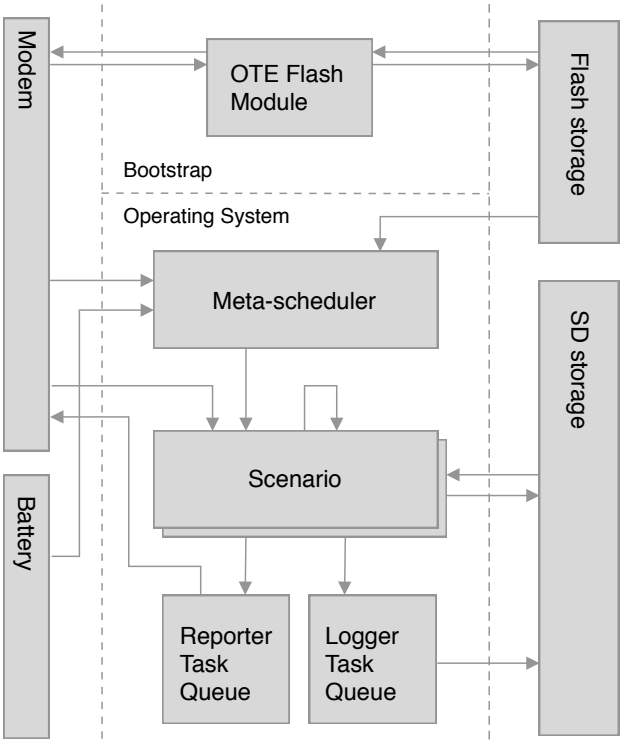


Figure 1: Overview of main logical software components and example hardware modules

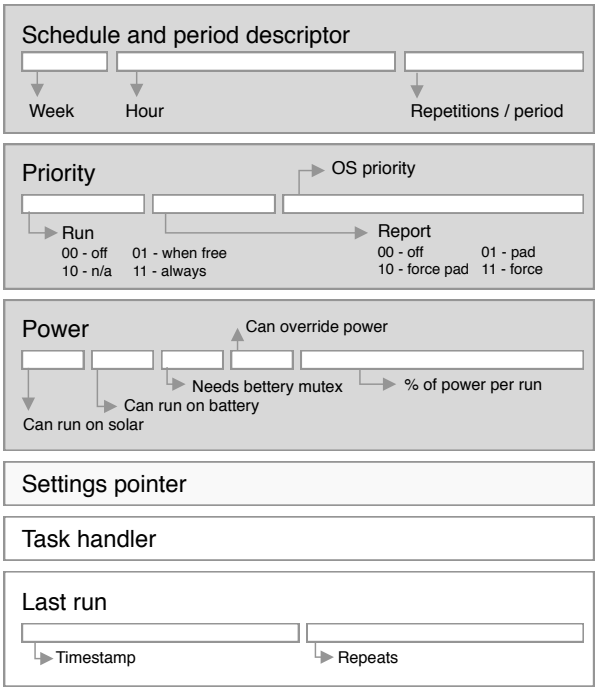


Figure 2: Scenario configuration (read-only variables in gray and runtime variables in white blocks)

only read from the scenario it belongs to. The settings are stored in the microcontroller flash and are accessed when a task requires them. The action of writing settings is done only via server commands.

2.1.2 *Schedule and period*

The basis for the schedule is time intervals in which the scenario runs and repetitions that happen in those defined intervals. The Meta-Scheduler uses the configuration to wake up the tasks at the specified hours and gives the task a timeout that co-responds with the set period. In this way, we can run the meta-scheduler in periods not more than one hour. The meta-scheduler is idempotent, so running it multiple consecutive times will not affect the running tasks.

2.1.3 *Priorities*

Priorities are the way the Meta-Scheduler decides if a task should run at all, or if it should only run when there is enough power in the budget or always in the case of critical tasks. The report priority is used for reporting and is there to keep a budget on power availability and network traffic.

2.1.4 *Power*

The power setting is a way to tell the Meta-Scheduler how much power a task needs to be able to run so that significant power-consuming tasks will only run when there is enough energy available. Specific scenarios can execute depending on power status. Configuration assumes the following conditions: the battery is full, the solar array produces power, the battery is charging, or when the battery is almost drained, but we still want the task to run regardless of any power budgets. The power budget for a task is defined in the percentage of the total power available to the probe.

2.1.5 *Task Notifications*

A task is based on the FreeRTOS task [1] and its notification variable that has, by default 4 bytes. The first 3 bytes are used for custom messages send to the task, and the last byte is used as flags to set repeats (timeouts) that the task should run. Zero repeats mean the task will only run once in that time-frame and put themselves into infinite sleep mode after that. One or more repeats places the task into a timeout of 60 minutes divided by the repeat rate, at a minimum of 5-minute interval. More granular repeats can be made within the task itself and are not registered with the Meta-Scheduler. Any task with an interval will run until the next time the Meta-Scheduler is run, and it stops the execution.

2.1.6 *Logger*

The Logger is a special scenario, available to all tasks for logging purposes. It runs in a best-effort manner, using multiple queues with messages grouped by severity. The logs are handled with the notification variable of the header.

The logs can be optionally saved on the SD card and only if there is enough power and enough time has passed since the last write. Since the queue could have been filled up since the last write, the queue is cycled with the oldest messages being reused. In the event of a hardware failure, the Logger is disabled.

Each entry has a timestamp, a scenario id, and a fixed size data blob to store the log message or data.

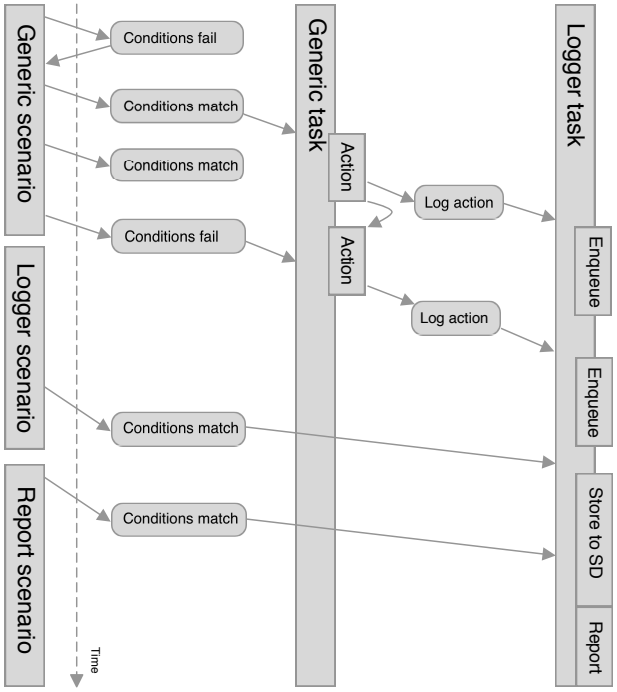


Figure 3: Timeline of an example generic task, logger, and reporter

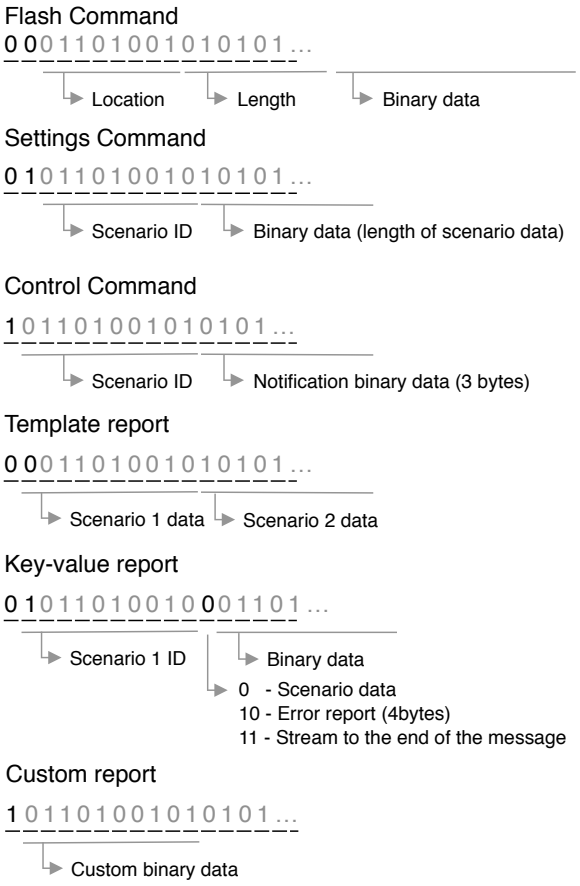


Figure 4: Command and report encoding format

2.2 Communication

Communication size is reduced to a minimum with compression based on common-knowledge. Since the server has the same template, we can use that to avoid sending headers before each data blob in some cases. The packet size, in our example, the Iridium satellite network, is limited to 360 bytes from the probe and 270 bytes to the probe with charge intervals of 50 bytes [2]. We use those limits to optimize expenses on low-value data.

2.2.1 Reporting

The reporter is a special scenario, available to all tasks for reporting purposes. Each priority level has a queue, and repeat entries are overwritten. Scenarios send notifications to the Reporter task using the notification variable and are saved to the queue, awaiting packet construction. At predefined intervals, the packet is constructed using the available reports (see figure 1). Critical reports can be sent and are flushed to the modem instantly using a special flag in the report command.

To save space, we encode the reports into several kinds of formats, denoted by the first few bits in the message.

2.2.2 Templated Report

The primary way to report is to use the section arrangement on the microcontroller flash and the priority values to order the scenarios. The first bit in the stream of each scenario reports if the bits that follow are from the scenario. The domain-specific language denotes the length of the data the scenario will report. If the report bit is zero, we skip to the next scenario with only 1 bit used to determine the scenario has nothing to report. This method will not apply if there are any scenarios with errors to report. If there is enough space at the end of the packet, we fill the rest of the 50-byte section with key-values, as explained in the next section.

2.2.3 Key-Value Reporting

The Key-Value is the most basic report used when there are errors or unknown data type in the templated report. The scenarios are again listed one after the other, prefixed with the scenario key and the type of data prefix. Unlike the templated report, the type of data can denote if the following bytes are fixed width or if a length follows.

2.2.4 Custom Report

The custom report is a special report that can be only sent on demand from a scenario and has no structure beyond the first bit. This type of report is intended for crash reports and critical errors and is not meant to be automatically handled.

2.3 Commands

Commands are packets sent from the server to configure or run scenarios and to update any part of the microcontroller flash.

2.3.1 Flash Commands

Flash commands (prefix 00) are simple write commands that can be used either in the Bootstrap or the full Operating System mode. The predefined nature of the flash allows us to overwrite any scenario and core software. The command starts with a location in the microcontroller flash, followed by the length of the data to be written and the binary data itself. The structure can repeat to the end of the packet.

2.3.2 *Settings Commands*

The settings command (prefix 01) is attached to a scenario and therefore used with the Scenario id to find the location of the settings block in microcontroller flash. The predefined length provides safety from overflows. This structure can repeat to the end of the packet.

2.3.3 *Control Commands*

Control commands are used to control task using their notification variable directly and can, in most cases, be only run within the full operating system mode. This structure is composed of 3 bytes that are available as parameters sent to the task.

2.3.4 *System Control Commands*

System control commands use the same pattern as simple control commands, but they use the predefined system namespace. These commands can be used in either Bootstrap or Operating System mode and are used for tasks like flashing, rebooting, and other non-scenario tasks.

2.4 Over-the-air flash procedure

Changing any part of the software, including the Bootstrap, system procedures, and scenarios can is accomplished with the flash procedure using system control commands.

The procedure should be started by resetting the mutable image on the flash storage with an exact copy of the original software in the microcontroller flash. The next steps are done using the flash commands to write changes into the mutable image. The last step is sending the flash system command with the hash of the image that we want to write into the boot storage.

If the prepared image hash does not match the provided value, a critical message is sent to the server, and the procedure is broken off. If the hash does match, the system is rebooted, the new image is written to the appropriate sector, and the bootstrap procedure started.

In the event of a critical failure, the server can send a command that flashes the original image to the appropriate location and repeats the bootstrap procedure. Alternatively, if desired, a complete custom image cand be sent at the expense of increased network data usage.

2.5 Bootstrap

Bootstrapping the system involves loading all the scenarios into memory. All the generic scenarios are stored in consecutive fixed width blocks with a header. The loader reads the header of each block for a magic number to see if the block contains a scenario and tests the checksum. If successful, the settings pointer is checked and if needed, the settings

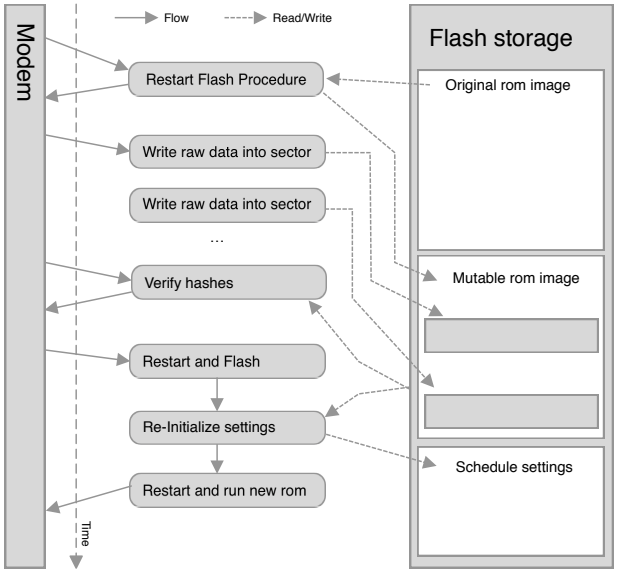


Figure 5: Timeline of a ota flash and re-initialization (left) with the flash layout (right)

block is initialized. Finally, the tasks of the scenario are run and placed into a infinite wait state.

2.6 Conclusion

The inspiration for designing such a system derives from a separate student project designing a floating sensor device, intended for data gathering in the ocean. As such, proposed implementation centers around the need for an embedded autonomous system with the primary goal of collecting data depending on detected conditions, available energy reserves, and network availability or cost. The lifetime of such a device is expected to exceed a year. Therefore the system also takes into account the need for remote over the air system updates and changing of execution parameters.

Current anticipated technical difficulties, regarding a working implementation on an actual microcontroller, revolves around physical memory or flash partitioning for remote updates, memory allocation and number of tasks being able to run or ques being able to exist as buffers and operating data storage, and the ability to correctly estimate the power available to the system for proper task execution.

3. REFERENCES

[1] R. Goyette. An analysis and description of the inner workings of the freertos kernel. *Carleton University*, 5, 2007.

[2] J. Hutcheson and M. Laurin. Network flexibility of the iridium (r) global mobile satellite system. 1995.

Covering problems and Influence maximization

Gyöngyvér Vass
Institute of Informatics
University of Szeged
Szeged, Hungary
Vass.Gyongyver@stud.u-szeged.hu

Boglárka G.-Tóth
Institute of Informatics
University of Szeged
Szeged, Hungary
boglarka@inf.szte.hu

ABSTRACT

Influence maximization is a very popular problem in the social sciences. It seeks a given number of seed points (vertices) in a network to maximize the number of influenced vertices starting from the seeds. Influencing may occur trough the edges, which indicate the connection between people (vertices). There are different ways to define influence, but up to our knowledge, finding the seeds from which maximal influence can be reached is a difficult task in general.

Coverage models belong to facility location problems, where centers are to be located such that the covered demand points (vertices in a graph within a given distance) are maximized. These models are motivated by problems where some services are only available within a fixed radius, like ambulances or fast food delivery. These problems are solvable for large graphs, and our long term aim is to use the most appropriate and/or adjusted covering models for solving influence maximization problems. As a first step, in this paper, we compare influence maximization and coverage models and analyze their differences.

As we will show, there are many similarities between the models, however, the main difference is that covering is a static action while influencing is dynamic. We show when this difference can be resolved, and how different are the results when not.

Categories and Subject Descriptors

G.2.2 [**Mathematics of Computing**]: Graph TheoryNetwork problems; G.1.6 [**Mathematics of Computing**]: Optimization

General Terms

Theory, application

Keywords

Covering models, Influence maximization, Mixed Integer Programming

1. INTRODUCTION

The influence maximization problem explores that from a given seed point how many other individuals can be reached by the information in a social network. The transmitting of the information and the terms of the transmitting can be different, but usually, the capability of the influencing is impressed by the weight or strength of the edge between two points. Influence maximization is also known as information diffusion. For a survey on the subject see [2, 3].

In facility location, the covering problem explores the point or points that can cover other points in a demand network within a given covering distance. For a good introduction to facility location problems, see [1].

Up to the knowledge of the authors, these models have not yet been compared in the literature, thus results in this paper are completely novel. As covering models are well studied and efficient algorithms exist to solve large scale problems, it is interesting to investigate if the influence maximization problem (or its slight modification) can be solved by any approach made for covering models. Thus, we aim to compare influence maximization and covering models as a first step.

2. INFLUENCE MAXIMIZATION

Formally, the influence maximization problem can be defined as follows. There is a simple graph $G = (V, E)$, where the vertices of the graph $v \in V$ represent the individuals and the edges represent the connections between them. Hereafter we can regard as $|V| = n$ and $|E| = m$, thus, in the social network there are n people, and m connections between them. We assign a weight to each edge $e \in E$, which will give us a probability: $f(e) : E \to [0, 1]$ the probability that information can spread trough a given edge e, also it can be seen as the strength of the relationship.

For the influence maximization model, there is a set $S \subset V$, from which the information will start, called seed set. The cardinality of the set S is fixed, we will denote it with s ($s < n$). This gives us a general diffusion model, where the diffusion function is: $\sigma(S) : V \to [s, n]$, that is, the number of vertices influenced by the seed set S. We seek for the seed set maximizing the influence, i.e. $\max_{S \subset V} \sigma(S)$.

DOI: https://doi.org/10.26493/978-961-7055-82-5.71-74

In the general model described above the processing of the influencing can be differently modeled. The most known models are the linear threshold model and the independent cascade model, see [3], however for our purposes the so-called triggering model is the most appropriate.

2.1 Triggering Model

In the triggering model, for each $v \in V$ we independently choose a random T_v triggering set according to some distribution over subsets of its incoming neighbours. The T_v set is a subset of the neighbours of vertex v. At the beginning of the process, the seed set S will be active. An inactive point becomes active at time instant t, if any element of the selected triggering set T_v becomes active at time instance $t-1$. Formally, for $v \in V \backslash S$ if exists $v' \in T_v$ such that $v' \in A_{t-1}$, then $v \in A_t$.

It can be seen that in this model, probability and threshold, used in the threshold model and the independent cascade model [3], are replaced by an influencer set that represents the way information is spread. This also means that this model is deterministic from the point where the triggering sets are chosen.

We have designed a mathematical model for this problem so that it can be solved with a Mixed Integer Programming (MIP) solver. As a parameter, we need the maximum number of steps of the influencing process, $t_{\max}$. It is not known beforehand but can be taken as the diameter of the graph, which is a good upper bound for the run time. The rest of the data is the graph itself and the triggering set for each vertex T_j, which includes a subset of the neighbours of j and also j.

Decision variables:

$$Z_{jt} = \begin{cases} 1, & \text{if point } j \text{ is active at step } t, \\ 0, & \text{otherwise} \end{cases}$$

$$\max \quad \sum_j Z_{jt_{\max}} \tag{1}$$

$$\text{s.t.} \quad \sum_j Z_{j0} = s \tag{2}$$

$$\sum_{i \in T_j} Z_{it} \geq Z_{j,t+1} \quad \forall j \in V,\ 0 \leq t < t_{\max} \tag{3}$$

Variables $Z_{jt_{\max}}$ gives us the resulting influenced nodes after $t_{\max}$ steps, thus the objective function is to maximize their sum. Condition (2) sets the number of initial seeds to s, while in (3) the influencing is defined. Namely, a vertex j is influenced if its neighbours in T_j are influenced. By maximizing the influence, the objective guarantees that $Z_{j,t+1}$ will always take value 1 if the left-hand side of (3) allows it, so no lower bounding condition on $Z_{j,t+1}$ is necessary.

3. COVERING PROBLEMS

Covering problems belong to the field of facility location. We interpret it on a network and consider vertices as demand points. Our goal is to place facilities or service units at some vertices of the network that can cover as many demand points as possible. Distance between two vertices is defined by the length of the shortest path between the two points, where edge lengths may be given or considered unit length. The covering distance or covering radius is also given to the problem, which tells us in what distance the new facility can cover the demand points. In real life, it may depend on the "size" of the facility, for example, in hospitals, its floor area affects how many beds can be laid out and thus how many patients can be accommodated, or on the maximal distance a service can be realized (ambulance, fast food delivery for example). Formally, we consider a simple graph $G = (V, E)$ with the usual notation. The concept of coverage can be defined as follows. Our goal is either to maximize the covered demand by a fixed number of companies or to cover the entire network minimizing the number of centers, or building costs. We will only discuss the first model, called maximal covering.

3.1 Maximal Covering

In this model, we aim to maximize the number of covered demand points locating a fixed number of facilities at the vertices of the network. The number of facilities to be located is s, and the covering radius R should also be known for the problem. Formally, we can write the model as follows.

Parameters:

$$a_{ij} = \begin{cases} 1, & \text{if point } i \text{ can cover point } j, \text{ i.e. } d(i,j) \leq R \\ 0, & \text{otherwise} \end{cases}$$

Decision variables:

$$X_j = \begin{cases} 1, & \text{if a company at point } j \text{ is located,} \\ 0, & \text{otherwise} \end{cases}$$

$$Y_j = \begin{cases} 1, & \text{if point } j \text{ is covered,} \\ 0, & \text{otherwise} \end{cases}$$

Having these parameters and decision variables, the objective function and constraints can be written in the following way.

$$\max \quad \sum_j Y_j \tag{4}$$

$$\text{s.t.} \quad \sum_j X_j = s \tag{5}$$

$$\sum_i a_{ij} X_i \geq Y_j \quad \forall j \tag{6}$$

The objective of the model described in (4) is to maximize coverage, which is the number of covered demand points. Constraint (5) ensures that exactly s facilities are located. Each demand point j has a constraint that ensures that demand point j is covered only if there is a facility located within the given distance, see (6). Knowing the coverage distance, we can determine the locations which could cover the demand point j.

Table 1: Comparison of the models

Covering problem		**Influence maximization**
graph of demand points and roads	✓	social network (graph)
we want to locate facilities at demand points (vertices)	✓	we want to find the seed points (vertices)
cover other demand points	✓	**influence** other points
Covering: a demand point is covered if there is a company for which their distance is less than the coverage distance	✗	Influencing: a point is influenced, if any neighbour could influence it with the probability of the edge
static, only facilities can cover	✗	dynamic, anyone can influence
edge weights are distances	✗	edge weights are probabilities
deterministic	✗	stochastic

4. COMPARISON OF THE MODELS

Let's first look at the similarities and differences between the two models in general, summarized in Table 1.

Both models work with simple graphs and choose vertices based on the weights of the edges. None of the models need to add any new point, just choose one or more from the existing vertices (these are going to be the seeds or centers), and either spread the information to the neighbours or be the location for the new facilities. The aim of both models is to reach as many points as possible. However, the definition and the way the seeds reach the other vertices are different. We count a point as covered if there is at least one facility where their distance, that is the total weight of the shortest path between the demand point and the new company, is less than the given covering distance. We count a point as influenced if at least one of its neighbours influence it with the edge probability between them. Thus, one of the largest difference is that the maximal covering problem is static, while the information diffusion is dynamic.You: Besides these, another difference is, that in the covering model only the new facility or facilities can cover the demand points, while in the influence maximization problem every influenced point can further influence its neighbours. In the first case, there is a concrete distance, whilst in the second case, there are only probabilities for the spreading, where the spreading can take any number of steps. It also means that covering is deterministic, as the coverage is always the same, while information diffusion is stochastic, as every time we generate random values to simulate the spreading.

4.1 Comparison of a modified triggering model and the maximal covering model

A modified triggering model may provide a solution to overcome these differences. To repeat, in the triggering model, each point independently selects a random set (triggering set T_j) based on the distribution of subsets of the neighbours. If one point in T_j becomes active, so does the point j. This will result in two types of edges, "live" and "blocked" as they belong to the triggering set or not. Now, considering the sub-graph with only the "live" edges, and restricting the run time of the triggering model to R number of steps, the problem equivalent to the maximal covering problem on the sub-graph with covering radius R.

Therefore, we propose to modify the triggering model by setting a maximum time for the spread of information, that is the number of steps.

The modified triggering model can be solved by the optimization problem in (1-3) where the maximum number of steps is set to R, and T_j is the triggering set for each vertex, being the set of all the neighbours of j.

Let us show that the model (1-3) is equivalent to the maximal covering model (4-6). As a first step, let us rewrite the condition (6) without the parameter a_{ij}, only relying on $d(i,j)$, the distance of vertices i and j.

$$\sum_{i \in V : d(i,j) \leq R} X_i \geq Y_j \quad \forall j \tag{7}$$

Now, in order to show the equivalence, we write the models side by side, where in each line the corresponding parts are given.

$$\max \sum_j Y_j \qquad \max \sum_j Z_{jR} \tag{8}$$

$$\text{s.t.} \sum_j X_j = s \qquad \text{s.t.} \sum_j Z_{j0} = s \tag{9}$$

$$\sum_{j \in V : d(i,j) \leq R} X_j \geq Y_i \qquad \sum_{i \in T_j} Z_{it} \geq Z_{j,t+1} \tag{10}$$

$$\forall i \qquad \forall j, 0 \leq t < R$$

For the variables, $Y_j \equiv Z_{jR}$, and $X_j \equiv Z_{j0}$, which makes the equivalence in lines (8-9) obvious. In order to see that the conditions in (10) are defining the same constraints, let is rewrite the influence condition as follows.

Aggregating $\sum_{k \in T_i} Z_{k,t-1} \geq Z_{it}$ and $\sum_{i \in T_j} Z_{it} \geq Z_{j,t+1}$ we obtain

$$\sum_{k \in T_i} \sum_{i \in T_j} Z_{k,t-1} \geq Z_{j,t+1}$$

and following this for all the R steps, the condition becomes

$$\sum_{i_0 \in T_{i_1}} \sum_{i_1 \in T_{i_2}} \cdots \sum_{i_{R-1} \in T_j} Z_{i_0,0} \geq Z_{j,R}$$

Now, if we take into account, that the above sums are only adding those $Z_{i_0,0}$, where $i_0 \in T_{i_1}, i_1 \in T_{i_2}, \ldots, i_{R-1} \in T_j$, the requirement is actually the same as $d(i_0, j) \leq R$. There-

fore the condition can be written as

$$\sum_{j \in V : d(i,j) \leq R} Z_{i,0} \geq Z_{j,R} \quad \forall j \tag{11}$$

which is equivalent to (7) changing the names of the variables.

4.2 Computational comparison of the models

To compare the models we implemented them in AMPL, a mathematical programming language. We use (1-3) to define the triggering model and (4-6) for the maximum covering model. The problems were solved by the commercial optimization solver CPLEX.

Let us show the results on an example graph drawn in Figure 1. We run the maximum covering problem on this graph and empirically showed that the same result is obtained when we include all neighbours in the triggering sets.

In order to generate the triggering sets for the general model, we use a selection rate r, which gives the proportion of the neighbours to be selected to the triggering sets. If the selection rate $r = 1$, we take every neighbours of each vertex to its triggering set. If $r < 1$, we select randomly from the neighbouring edges until we have the required number of edges in the triggering sets.

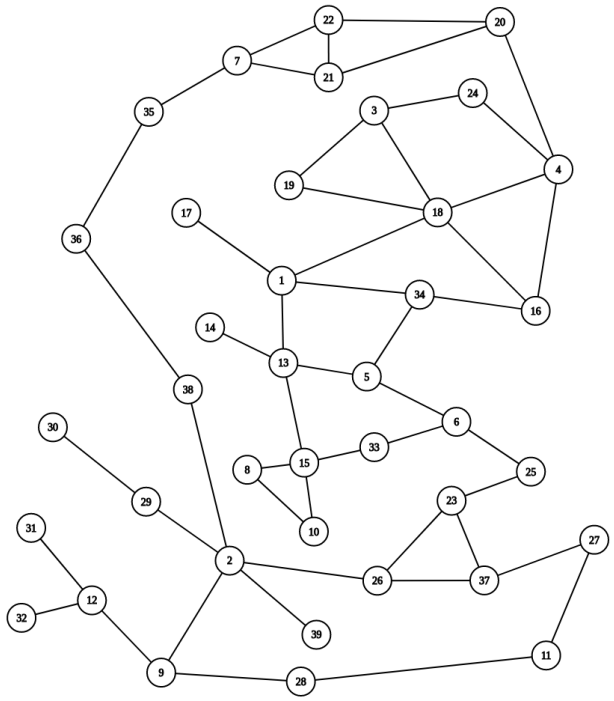


Figure 1: Example graph with 39 nodes.

We summarized the results in Table 2. We set the selection rate from 0.3 to 1 per 0.1. First, we set the cardinality of the seed set $|S|$ to 1 and the run time R to 4, and reported the average of the objective value after 50 runs together with the standard deviation for each selection rate. Not surprisingly, decreasing the rate, the average number of influenced points is also decreasing. The standard deviation is rather high, which can be explained by the structure of the graph. There

Table 2: Computational result for the modified triggering model with different number of seeds $|S|$ and run time R.

	$\|S\| = 1, R = 4$		$\|S\| = 2, R = 3$		$\|S\| = 3, R = 2$	
Rate	Avg	Dev	Avg	Dev	Avg	Dev
0.3	9.1	2.0	14.3	2.0	15.8	1.9
0.4	11.1	2.4	15.7	1.8	17.6	1.6
0.5	14.8	2.0	20.6	2.2	20.7	1.7
0.6	16.6	2.2	24.4	2.5	24.8	2.0
0.7	17.9	1.7	26.7	2.2	26.1	1.6
0.8	20.8	0.8	32.9	2.8	29.8	1.4
0.9	22.0	0.0	35.9	0.5	31.9	0.3
1.0	22.0	0.0	36.0	0.0	32.0	0.0

are long chains in the graph, like the paths 2-7 or 9-37, from which any edges become blocked, the influenced set can change easily. This also means that the seed is very unstable between the different runs, even for high selection rates.

In the next columns we report the results for $|S| = 2$ and $R = 3$, and also for $|S| = 3$ and $R = 2$. The obtained results are quite similar as before, although we can see that the deviation of the results is smaller for the last case ($|S| = 3$, $R = 2$). From these results, we can see that for high selection rates the models give quite similar results in terms of the objective value, but we have seen the seed sets are quite different except for rate 1.

5. CONCLUSIONS

It has been intuitively shown that maximal covering and influence maximization deal with a similar problem. The research revealed to us what similarities and differences are. In both cases, we start from a very similar problem and have a similar goal. However, we also show that one of the largest differences is the static nature of facility location and the dynamic nature of information diffusion. We have seen a solution to this by using the modified triggering model. It is planned to compare the results of the models for a large set of networks and to analyze more information diffusion models in the near future.

6. ACKNOWLEDGMENTS

The project was supported by the European Union, co-financed by the European Social Fund (EFOP-3.6.3-VEKOP-16-2017-00002).

7. REFERENCES

[1] M. S. Daskin. *Network and Discrete Location: Models, Algorithms and Applications.* John Wiley and Sons, New York, 1995.

[2] V. Homolya. Analysis of information diffusion in networks (in hungarian), 2017. Hungarian Conference for Students (OTDK).

[3] G. D. Nittis and N. Gatti. How to maximize the spread of social influence: A survey. *CoRR*, abs/1806.07757, 2018.

Strong deep learning baseline for single lead ECG processing

BOTOS Csaba
Pázmány Péter Catholic University
1083 Práter utca 50/A
Budapest, Hungary
botos.csaba@hallgato.ppke.hu

HAKKEL Tamás
Pázmány Péter Catholic University
1083 Práter utca 50/A
Budapest, Hungary
hakkel.tamas@hallgato.ppke.hu

GODA Márton Áron*
Pázmány Péter Catholic University
1083 Práter utca 50/A
Budapest, Hungary
goda.marton.aron@itk.ppke.hu

REGULY István Z.†
Pázmány Péter Catholic University
1083 Práter utca 50/A
Budapest, Hungary
reguly.istvan@itk.ppke.hu

HORVÁTH András‡
Pázmány Péter Catholic University
1083 Práter utca 50/A
Budapest, Hungary
horvath.andras@itk.ppke.hu

ABSTRACT

Objective: Atrial fibrillation (AF) is one of the most common serious abnormal heart rhythm conditions, and the number of deaths related to atrial fibrillation has increased by an order of magnitude in the past decades. We aim to create a system, which can provide help for cardiologist, classifying and highlighting important segments in recordings.

Approach: In this paper, we propose a novel approach for AF detection using only a deep neural architecture without any traditional feature extractor for real-time automated suggestions of possible cardiac failures that can detect class invariant anomalies in signals recorded by a single channel portable ECG device.

Results: Detecting the four categories: *Normal*, *AF*, *Other* and *Noisy* in terms of the official, F1 metric of hidden dataset maintained by the organizers of PhysioNet Computing in Cardiology Challenge 2017, our proposed algorithm has scored 0.88, 0.80, 0.69, 0.64 points respectively, and 0.79 on average.

Keywords

deep learning, residual network, fully convolutional network, time-series, signal processing, ECG, atrial fibrillation, AF detection

*Corresponding author.
†Corresponding author.
‡Corresponding author.

1. INTRODUCTION

Cardiovascular diseases are responsible for the highest percentage of fatal outcomes among health problems in the modern world. One of the most common and serious abnormal heart rhythm conditions is atrial fibrillation, which affects about 2% to 3% of the population in Europe and North America [1]. It is associated with an increased risk of heart failure, dementia, and stroke. Additionally, the number of deaths related to atrial fibrillation has increased by an order of magnitude in recent decades: growing from 29,000 in 1990 up to 193,300 in 2015. Researchers project that by 2030 cardiovascular diseases will account for more than three-quarters of deaths worldwide [2].

While it is essential to develop efficient algorithms to automatize detection for monitoring patients with small portable or wearable devices, and promising methods [3, 4] are already available, there is still no completely satisfying solution due to the low signal-to-noise ratio of portable ECG devices, as well as the multiple types and the episodic manner of atrial fibrillation. Unfortunately, detecting atrial fibrillation poses a significant challenge even for the most experienced cardiac exerts. As a result, a larger time window has to be recorded and examined by experts to arrive at a diagnosis.

To promote the solution and draw the attention of the scientific community to this problem, a challenge was introduced by PhysioNet [5], which targets the algorithmic classification of atrial fibrillation signals. In this challenge, 8528 short, single-channel recordings were provided produced by a low-cost, portable device called KardiaMobile, manufactured by AliveCor Inc. [6]. The length of the recordings ranged from 9.0 seconds to 61.0 seconds with an average of 32.5 seconds. These samples were divided into four different classes: atrial fibrillation, normal, noisy signals, and recordings from patients with other cardiac diseases, consisting of 771, 5154, 46, and 2557 samples, respectively.

DOI: https://doi.org/10.26493/978-961-7055-82-5.75-83

2. RELATED WORKS

Using clinically meaningful features, good quality ECG measurements could be flawlessly classified by simply applying traditional machine learning techniques (i.e. logistic regression, random-tree, support-vector-machines). On the other hand, real-life samples often pose too much noise and high variance that could mislead handcrafted rules, and yet state-of-the-art approaches are still relying heavily on feature engineering for AF detection. Accordingly, three of the four winners of the CinC Cardiology 2017 challenge combined only medically relevant feature-extractors and did not incorporate any neural network-based features [7, 8, 9].

Only one of the four winner approaches fused expert features with other descriptors extracted by a neural network. Hong et al.[10] proposed an algorithm concerning 64 features learned by a Deep Neural Architecture, namely a time-invariant hierarchical feature extractor network with 4 residual blocks [11] combined with a Bi-directional Long Short-term Memory network (LSTM [12]) resulting in a 32 dimensional continuous descriptor and a Uni-directional LSTM trained separately using centerwave input to extract 32 time related features. While the final classifier was applied on a feature space with more than 600 dimensions, after ranking by importance, the top 20 were made up of 17 deep learned features and only the 3 remaining were clinically relevant or external statistical features.

At the same time, many other participants of the Challenge also used neural networks [13, 14, 15, 16] as feature detector in addition to their traditional feature extractors. One of them was Andreotti et al. [17], who compared their feature-based classifiers to residual neural networks. They concluded that their neural networks outperform their feature-based classifiers, showing the strength of the purely neural network-based approach. Parvaneh et al. [18] improved a dense convolutional network by signal quality index and by the transformation of signal to the frequency domain. Their approach was similar to ours as they applied a neural network to extract frequency-domain features. Xiong et al. [19] tried multiple methods with success, which we utilized as well, including skip connections, and a neural network trained on the spectrogram.

	Normal	AF	Other	Noise	Avg.
Teijeiro et al.	0.90	**0.85**	0.74	0.56	**0.83**
Datta et al.	**0.92**	0.82	**0.75**	0.52	**0.83**
Zabihi et al.	0.91	0.84	0.73	0.50	**0.83**
Hong et al.	0.91	0.81	**0.75**	0.57	**0.83**
ours	0.88	0.80	0.69	**0.64**	0.79

Table 1: F1 scores on the hidden test set of the CinC Challenge 2017. The winner algorithms (first 4 rows) excel in different tasks, since they utilize different pools of features. An important note is that in order to reduce prediction uncertainty many have submitted ensembles which improve the overall accuracy; however, does not reveal the true generalizing capabilities of the underlying algorithm.

3. METHODS

3.1 Extending Dataset with Alternative Annotation

We compete with human performance; however, we do not know much about that. There is no better reference currently than human annotation (possibly by experts). Atrial fibrillation is a human category, there is no mathematically exact definition for it, thus we need humans to define its characteristics. Unfortunately, these definitions are vague and fuzzy from the algorithmic point of view as there is always an inherent ambiguity in all applications when we try to approximate human definitions using mathematical models. To come around this problem we have created a dataset in which every sample was annotated by multiple (in our case two) experts to allow us measuring the variation of the annotations as well.

We asked two doctors to help us: Dr Sz. Herczeg PhD student in the field of cardiac arrhythmia (Expert-1), and Dr I. Osztheimer cardiologist consultant (Expert-2), both working at the Heart and Vascular Center of Semmelweis University, in Budapest. Our goal was to examine the difference between the decisions of experts of the Challenge, our doctors, and a model trained on this dataset. By this, we aimed to have an approximation of the accuracy of human performance. Then we wanted to explore which features are the most important ones our model is looking for. Finally, we made some efforts to highlight these important features to help human specialists.

To solve that task, we developed a website that displays the recordings and provides a graphical user interface to annotate the currently displayed recording. Asking our doctors to use that website, we obtained an alternative annotation that helped us to validate the data set, i.e. which the obvious cases are and which samples are too ambiguous to make a clear diagnosis. The website picks recordings randomly, selecting recordings from the four different classes uniformly.

3.2 Neural Network Architecture

Based on empirical evidence in the field of computer vision, to reduce training time and to make the resulting detector more robust, we applied recently published methods such as ADAM[20], SELU[21], dilated convolutions [22], residual blocks[23] - for which we will provide a quick overview in this section, and a more detailed description and summary of resulting improvements in appendix A. While several image recognition baseline NN architectures (such as ResNet and VGG) could be re-designed to fit the AF detection task, we developed domain-specific ensembles from core building blocks of the aforementioned baseline architectures. Alongside with the proposed networks, we have applied pre- and post-processing steps: forked feature extraction on both temporal and spectral domain, and merging encoded feature vectors from different domains directly under the final classifier layer.

Despite the moderate improvements on the temporal and spectral domains by the application of the advanced building blocks (Figure 1), the extension of the logistic regression on multi-domain feature representations resulted in an architecture that could significantly outperform the most robust

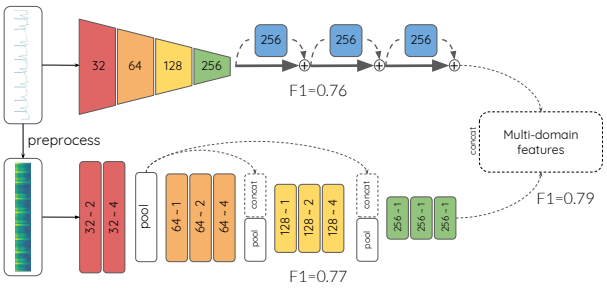


Figure 1: Learning representations from raw time-series and equal length spectrogram.

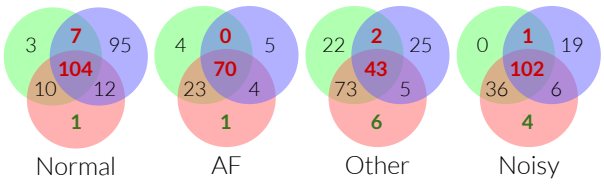


Figure 2: Venn diagrams showing the overlap between the 3 annotations: the official labels created by the cardiologists of the Challenge (green circle), Expert-1's annotation (blue circle) and the classification of the neural network (red circle).

networks on individual domains. We performed hyperparameter tuning in each domain separately, and then we selected the best performing models for the joint optimization.

The upper branch operating on the raw signal is inspired by **EncodeNet** [24] and uses consecutive residual blocks, which operate with high numbers of channels on the downscaled sample. The lower branch operating on spectral-domain is the **SkipFCN** [25], which connects the early low-level representations to latent layers to reduce over-fitting while stabilizing gradient flow.

More importantly, we wanted our research to give valuable feedback to doctors. Therefore, we inspected features that our trained AF detector has learned from samples provided by the Challenge to check whether these features were matching with ones recognized by professionals. These results are analyzed in the next section. For a more detailed description of the network architecture, see appendix A.

4. RESULTS

4.1 Main Results

Detecting the four categories: *Normal*, *AF*, *Other* and *Noisy* in terms of the official, F1 metric of the challenge, our proposed algorithm has scored 0.88, 0.80, 0.69, 0.64 points respectively, and 0.79 on average. The fact that our F1 score 0.64 on Noise detection was even higher then the same score of winning teams (see 2) shows that one is capable of reaching performance close to the state of the art methods without professional feature-engineering.

4.2 Ambiguity of Annotations

Using the website we designed for the alternative annotation, Expert-1 annotated 500 recordings. Comparing that annotation to the annotation of the cardiologists of the Challenge, we found that the two annotations showed matching only in 65% of cases, underlining the fact that classification of data set we worked on is challenging even for experts. Moreover, it turned out that considering only those recordings when both Expert-1 and the cardiologists of the challenge agreed on the classification (i.e., in case of the "evident recordings") the model also presented an almost identical (97.35%) classification.

To visualize the overlap between the 3 annotations we also drew Venn diagrams, see Figure 2. We created diagrams for each class and marked with red the number of the samples that were unambiguous for cardiologists (i.e., the classification of Expert-1 and the cardiologists of the Challenge was the same). It is clearly visible that for the vast majority of these evident samples, the neural network predicted the same class as the cardiologists did. Additionally, only a tiny fraction of samples (their number is colored to green) were classified by our model in such a way that none of the cardiologists agrees with its prediction. Unfortunately, it is obvious that no statistical analysis is possible due to the low number of evaluated samples (500 compared to $8,528$ available samples) and because only one expert completed that evaluation. However, these numbers strongly suggest that our results might have some medical relevance as our algorithm appears to mimic diagnosis of doctors.

4.3 Confidence of The Classifier

We created another web page to show recordings which were the easiest or the hardest to classify for our algorithm. To measure the "confidence" of the decision, we used the output of the last layer of our network (also called soft-max layer) which has three neurons and each neuron produces a number that correlates with the assumed probability of belonging to the *normal*, *AF*, or *other* classes, respectively. The fourth, *noisy* class is predicted by a separate network. We fed all recordings to the model and picked the top 10 recordings which produced the highest value from the neuron responsible for the *normal* class, and we repeated that experiment for the *AF* class. We assumed that those recordings can also be interesting that were difficult to decide for our model, and therefore we selected the worse 10-10 recordings that resulted in the lowest value from the neurons of the *normal* and *AF* class. The reason why we excluded *other* and *noisy* classes from examination are that these classes are only technically necessary for defining the problem, but they have no medical relevance regarding AF detection.

Then, we asked our experts to try to find some common features of the samples classified into the same classes by our model, and tell whether recordings classified "confidently" (i.e. with high output value) by our model were also evident for them. Similarly, we wanted to know whether they found the least "confidently" classified recordings obscure, too. They answered independently from each other, but their remarks were very similar in most aspects.

They both agreed that in the case of most confident predictions recordings had low noise contamination and this contamination could be easily distinguished from the signal.

Also, they both mentioned that the main difficulty of classifying recordings classified with the lowest confidence was the high amplitude noise and the irregular baseline changes as it made P-wave detection very difficult. In case of hardly detectable P-waves, they both would rather look at RR intervals to inspect whether they are regular or not.

In addition, Expert-2 noted that the model appeared to recognize the regularity of RR-intervals and used it as a strong evidence of *normal* class (doctors also consider it as a sign of healthy heart rhythm). Unfortunately, it was misleading sometimes because some recordings exhibited both unusually regular RR-intervals and also some clear signs of AF, and thus the model predicted *normal* class (with low confidence) instead of *AF*. Besides, Expert-1 noticed that all of the confidently classified AF recordings have arrhythmia absoluta (i.e., the RR intervals are always changing), and most of these recordings have high BPM value, while recordings of low confidence prediction have much lower BPM on average. She mentioned that arrhythmia absoluta and BPM are two of the most common features cardiologists are looking for in real-life clinical practice (along with the absence of P-wave). Thus, she acknowledged that our model appeared to learne some medically relevant feature without explicitly programming to do so.

4.4 Most relevant segments of recordings

We divided the signal to 50ms long segments and calculated the output of the neural network for each of them. As we did previously when we measured the "confidence" of the decision of the model, we took again the output of the three neurons of the soft-max layer, responsible for *Normal*, *AF* and *Other* classes, respectively. For each 50 ms long segment, the neuron that produced the highest value determined the color of the background behind the current section. When the neuron responsible for the *normal* class produced the highest value, then the background was colored to green. Similarly, the blue background indicated that the neuron of the *other* class had the highest output value, and red indicated *atrial fibrillation*. Additionally, higher values are translated to darker colors, so the darkness of background indicates the "confidence" of the prediction at a certain segment. An example of these graphs can be seen in Figure 3. By implementing that algorithm, we aimed to help doctors by highlighting the most relevant regions of ECG recordings. While our algorithm cannot substitute doctors, it might be a good tool to speed up the evaluation of long ECG recordings while unburdening physicians drawing their attention to the most important parts of the signal.

4.5 Computational Complexity

From the perspective of practical applicability in real-life medicine, our method is not just designed for classification, but performs well as a real-time detector by the nature of Fully Convolutional Networks: after the initial warm-up delay of 1.2 sec, we can generate new responses in less than 2 msec taking the last 20 second history into consideration. If the evaluation is centralized and we allow to compute responses in batches, the time required per sample is less than 0.5 msec.

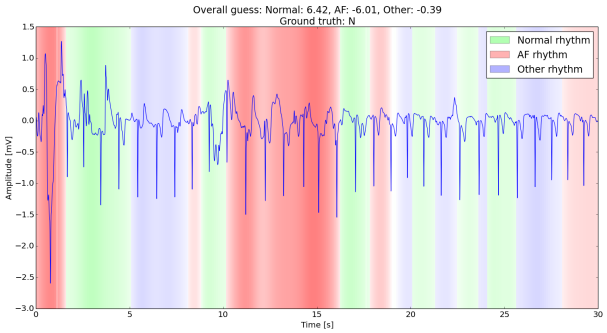


Figure 3: Example graph with colored background. Sections with green background are detected as normal rhythm sections, red indicates AF, and other arrhythmias are highlighted with blue.

5. CONCLUSION

While it is clear that current conditions demand expertise in both domains of cardiology and machine learning, the emergence of cheap hand-held devices creates a niche for approaches capable of utilizing larger amount of data, and that gives rise to adaptive and scalable algorithms, such as Deep Neural Networks.

We carried out an extensive architecture and hyperparameter search, and reported our findings on the Computing in Cardiology 2017 Challenge. To contribute to the field, we have open-sourced our project providing a general training environment for practitioners to quickly evaluate baseline performances on their dataset.

Our proposed algorithm provides visual reasoning and feedback for decision making that can significantly boost efficiency of AF detection in collaboration with experts. For deeper analysis of the performance, see appendix B. The website of our project is available at `http://physionet.itk.ppke.hu/`

Lastly, to help doctors to analyze long ECG recordings easily and quickly, we designed a tool that colors the background of the ECG plot highlighting the segments according to the prediction of the model.

6. REFERENCES

[1] C. A. Morillo, A. Banerjee, P. Perel, D. Wood, and X. Jouven, "Atrial fibrillation: the current epidemic," *Journal of geriatric cardiology: JGC*, vol. 14, no. 3, p. 195, 2017.

[2] B. B. Kelly, V. Fuster *et al.*, *Promoting cardiovascular health in the developing world: a critical challenge to achieve global health*. National Academies Press, 2010.

[3] G. H. Tison, J. M. Sanchez, B. Ballinger, A. Singh, J. E. Olgin, M. J. Pletcher, E. Vittinghoff, E. S. Lee, S. M. Fan, and R. A. e. a. Gladstone, "Passive detection of atrial fibrillation using a commercially available smartwatch," *JAMA Cardiology*, vol. 3, no. 5, p. 409, 2018.

[4] S. P. Shashikumar, A. J. Shah, Q. Li, G. D. Clifford, and S. Nemati, "A deep learning approach to

monitoring and detecting atrial fibrillation using wearable technology," *2017 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI)*, 2017.

[5] "AF Classification from a short single lead ECG recording: the PhysioNet/Computing in Cardiology Challenge 2017." [Online]. Available: https://physionet.org/challenge/2017/

[6] 2019. [Online]. Available: https://store.alivecor.com/products/kardiamobile

[7] T. Teijeiro, C. A. García, D. Castro, and P. Félix, "Arrhythmia classification from the abductive interpretation of short single-lead ECG records," *CoRR*, vol. abs/1711.03892, 2017. [Online]. Available: http://arxiv.org/abs/1711.03892

[8] S. Datta, C. Puri, A. Mukherjee, R. Banerjee, A. D. Choudhury, R. Singh, A. Ukil, S. Bandyopadhyay, A. Pal, and S. Khandelwal, "Identifying normal, af and other abnormal ecg rhythms using a cascaded binary classifier," *2017 Computing in Cardiology (CinC)*, pp. 1–4, 2017.

[9] M. Zabihi, A. B. Rad, A. K. Katsaggelos, S. Kiranyaz, S. Narkilahti, and M. Gabbouj, "Detection of atrial fibrillation in ecg hand-held devices using a random forest classifier," in *2017 Computing in Cardiology (CinC)*, Sept 2017, pp. 1–4.

[10] S. Hong, M. Wu, Y. Zhou, Q. Wang, J. Shang, H. Li, and J. Xie, "Encase: An ensemble classifier for ecg classification using expert features and deep neural networks," in *2017 Computing in Cardiology (CinC)*, Sept 2017, pp. 1–4.

[11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *arXiv:1512.03385 [cs]*, Dec. 2015, arXiv: 1512.03385. [Online]. Available: http://arxiv.org/abs/1512.03385

[12] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, pp. 1735–80, 12 1997.

[13] P. Warrick and M. N. Homsi, "Cardiac arrhythmia detection from ecg combining convolutional and long short-term memory networks," *2017 Computing in Cardiology Conference (CinC)*, 2017.

[14] F. Plesinger, P. Nejedly, I. Viscor, J. Halamek, and P. Jurak, "Automatic detection of atrial fibrillation and other arrhythmias in holter ecg recordings using rhythm features and neural networks," *2017 Computing in Cardiology Conference (CinC)*, 2017.

[15] M. Limam and F. Precioso, "Atrial fibrillation detection and ecg classification based on convolutional recurrent neural network," *2017 Computing in Cardiology Conference (CinC)*, 2017.

[16] M. Zihlmann, D. Perekrestenko, and M. Tschannen, "Convolutional recurrent neural networks for electrocardiogram classification," *2017 Computing in Cardiology Conference (CinC)*, 2017.

[17] F. Andreotti, O. Carr, M. A. F. Pimentel, A. Mahdi, and M. De Vos, "Comparing feature based classifiers and convolutional neural networks to detect arrhythmia from short segments of ecg," *2017 Computing in Cardiology Conference (CinC)*, 2017.

[18] S. Parvaneh, J. Rubin, R. Asif, B. Conroy, and S. Babaeizadeh, "Densely connected convolutional networks and signal quality analysis to detect atrial fibrillation using short single-lead ecg recordings," *2017 Computing in Cardiology Conference (CinC)*, 2017.

[19] Z. Xiong, M. Stiles, and J. Zhao, "Robust ecg signal classification for the detection of atrial fibrillation using novel neural networks," *2017 Computing in Cardiology Conference (CinC)*, 2017.

[20] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014. [Online]. Available: https://arxiv.org/abs/1412.6980

[21] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter, "Self-Normalizing Neural Networks," *arXiv:1706.02515 [cs, stat]*, Jun. 2017, arXiv: 1706.02515. [Online]. Available: http://arxiv.org/abs/1706.02515

[22] F. Yu and V. Koltun, "Multi-Scale Context Aggregation by Dilated Convolutions," *arXiv:1511.07122 [cs]*, Nov. 2015, arXiv: 1511.07122. [Online]. Available: http://arxiv.org/abs/1511.07122

[23] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[24] L. A. Gatys, A. S. Ecker, M. Bethge, A. Hertzmann, and E. Shechtman, "Controlling perceptual factors in neural style transfer," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[25] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3431–3440. [Online]. Available: http://www.cv-foundation.org/openaccess/content_cvpr_2015/html/Long_Fully_Convolutional_Networks_2015_CVPR_paper.html

[26] A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "WaveNet: A Generative Model for Raw Audio," *arXiv:1609.03499 [cs]*, Sep. 2016, arXiv: 1609.03499. [Online]. Available: http://arxiv.org/abs/1609.03499

[27] F. N. Iandola, M. W. Moskewicz, K. Ashraf, S. Han, W. J. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <1mb model size," *CoRR*, vol. abs/1602.07360, 2016. [Online]. Available: http://arxiv.org/abs/1602.07360

[28] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *arXiv:1409.1556 [cs]*, Sep. 2014, arXiv: 1409.1556. [Online]. Available: http://arxiv.org/abs/1409.1556

[29] P. Rajpurkar, A. Y. Hannun, M. Haghpanahi, C. Bourn, and A. Y. Ng, "Cardiologist-Level Arrhythmia Detection with Convolutional Neural Networks," *arXiv:1707.01836 [cs]*, Jul. 2017, arXiv: 1707.01836. [Online]. Available: http://arxiv.org/abs/1707.01836

[30] I. Goodfellow, Y. Bengio, and A. Courville, "Deep learning," 2016, book in preparation for MIT Press. [Online]. Available: http://www.deeplearningbook.org

APPENDIX

A. DETAILS OF THE NEURAL NETWORK

A.1 Activation Function

We have found that one major reason of over-fitting where the network rather memorizes training samples than generalizes was the non-linearity applied after parametrized layers, the Rectified Linear Unit (ReLU): $relu(x) = max(0, x)$

The phenomenon of "dead neurons" is a well known and a frequent issue among networks that apply ReLU. Generally speaking, ReLU suppresses any inhibiting activation (by clipping off negative values), excluding a notable portion of neurons in the preceding layer from succeeding layers; thus, activation spectrum of the layer will be saturated. Practically it is acceptable if different samples cause different neurons to be mitigated; however, during train time some of the neurons can become completely silent since the gradient optimization prevents muted neurons to be trained (their gradient is 0). Because of this property, there is a considerable risk that numerous nodes will not be able to influence the response of classifier neurons.

Instead of ReLU, we used SELU activation function published in Self-Normalizing Neural Networks [21]. By changing the activation function, we were able to overcome the variance-problem of the networks applied, i.e. distance between training- and test-performance was reduced for identical architectures. For benchmarks on VGG models see Figure 4.

On the left side of Figure 4 two separate trends are revealed. Apparently, ReLU (bold lines) outperforms identical networks applied with SELU (light lines), almost reaching ideal performance. *On the right side of Figure 4*: We can see ReLU networks reaching their top test-performance in early stages of training, and by continuing their training their accuracy decreased. In contrast, the accuracy of SELU networks gradually improves throughout the entire training. *Naming convention*: ADAM stands for gradient optimization method, 16/19 for the number of layers that have adjustable weights, and double/halved/quart suffixes refers to the depth of each convolutional filter applied in corresponding VGG baseline networks.

A.2 Dilated Convolution

Receptive field problem was another obstacle we encountered while setting up the baselines experiments. Simply by changing the 2-dimensional convolutions (3x3 filters) to their 1-dimensional equivalent (1x9 filters), we ended up with a network that could barely cover multiple heartbeats. Since we have learned that atrial fibrillation can be episodic, it was essential extending search space of architectures that could cover entire episodes. By applying causal dilated convolutional filters used by [26], the receptive field was exponentially increased further improving our models' accuracy without involving variance problems (like max-pooling does) or sacrificing evaluation speed since applying dilated convolution results in minimal overhead compared to the traditional operation. For the visual example see Figure 5.

A.3 Spectrogram

Representing a prerecorded signal to the frequency domain with Fast Fourier Transform is a favoured approach in the field of signal processing. Frequency analysis reveals each frequency band's presence in the signal, which may reveal periodic and aperiodic traits of the sample. In practice when a sample of the length N is transformed with FFT, it produces two arrays of values of the same length N representing complex (Im and Re) valued frequency coefficients. Usually, these two arrays are merged by the following formula $r = \sqrt{Im^2 + Re^2}$ resulting in Power Spectrum (PS), while phase (being less informative about the signal) is omitted thus making the transformation irreversible. The problem with taking PS is that it discards temporal patterns (such as Q-T, R-R distance etc.) making convolutional layers useless. Furthermore, PS is not casual by design, meaning that the whole signal must be provided before obtaining PS. A frequently applied technique in speech recognition is taking multiple FFT of short overlapping windows sliding over input audio sample and concatenating short samples' PS into a multi-channel array. Another slight detail is to apply piece-wise natural logarithm on every element of the resulting array to increase the variance of signal and prevent strong frequencies repress ones which are weaker with orders of magnitude.

The main advantage of that method is that it preserves time-domain (temporal) patterns, while it reveals the presence of different frequencies in the signal. Furthermore, there is only a slight difference when we apply different weighting on internal values of the sliding window, while window size and stride (i.e. inverse degree of overlapping) heavily influences how long our resulting array will be and how many frequency bands will represent it (i.e. resolution of PS at given time instance). We have found it not just incredibly convenient, but also surprisingly effective to choose the highest possible degree of overlapping (window stridden by 1), and resampling resulting spectrogram to match the length of the original signal. Taking the original sample and redundant representation of ECG-recording (a 64 channel spectrogram) of the same length allowed us to apply two concurrent NN on each domain (temporal and spectral), and to concatenate resulting representations in-depth without being forced to reduce temporal dimension since both feature vectors were of the same length.

A.4 Multi-Domain Representation Learning

While both input spaces, temporal and spectral, had their challenges, we saw that - by designing preprocess steps for spectrogram training - the feature extractor network produced output of the same length as a time-domain equivalent algorithm. That led us to try and concatenate these pretrained feature extractors together to test whether multi-domain representation could help the final layer to overcome issues specific to separate domain classification by complementing each other. Indeed, we found that the general behaviour of the time-domain network was as follows: increasing the accuracy of a single class at the expense of severe forgetting in other classes disappeared using multi-domain features. At the same time, spectral-domain networks struggled with variance problems, even with extremely low capacity. Also, networks trained in frequency domain were more dependent on the choice of training / evaluating data. These traits are omitted when feature extractors are working in

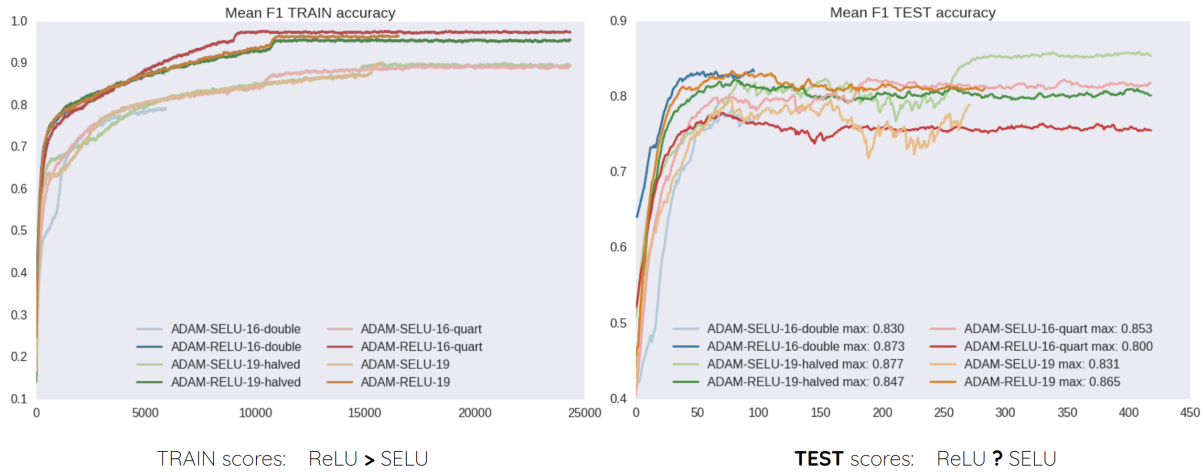


Figure 4: Mean F1 train accuracy

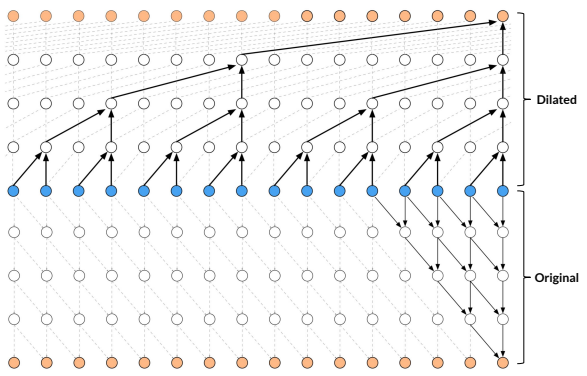


Figure 5: Using the same number of parameters at each node (2) with dilated convolutions, we can increase receptive field exponentially instead of linearly expanding receptive field of traditional convolutions

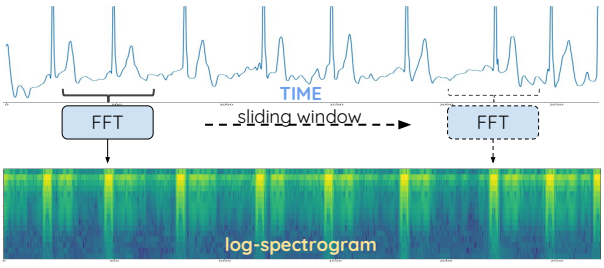


Figure 6: Using stride of 1 for 255 wide FFT windows resulted in almost identical length of original sample with 128 channels. In order to have a completely matching size in temporal dimension (horizontal axis), we resampled the log-spectrogram with nearest-neighbour interpolation.

parallel as well. For a detailed description of our proposed model for CinC Challenge of 2017, see Figure 1.

A.5 Training

Each network was trained for 420 epochs on 80% of the challenge dataset, which we have resampled for each training instance. Depending on the complexity of the underlying architecture, training took 3-12 hours on a K80 GPU. Our proposed algorithm is designed in such a way that applied operations are completely causal; thus, the input can be processed on-line (also in real-time) without the presence of a complete sequence.

Inspired by their simplicity we have re-designed the classic ImageNet models: SqueezeNet v1.1 [27], VGG-16/19 [28], ResNet18/152 [23]. We also re-implemented a network proposed by Rajpurkar et *al.* [29], which was developed to perform well on a more balanced dataset of over sixty thousand single lead ECG samples, annotated by lead expert cardiologists. We reference this network in this writing as StanfordNet.

Deep representation learning algorithms tend to outperform humans when the network is shown more data than its opponent during its lifetime. While the problem itself could be a truly complex task from the perspective of traditional algorithmic solutions, it is less difficult with human references.

Referring to the rule of thumb mentioned in [30], it is increasingly evident that state-of-the-art supervised training methods can reach or even exceed human performance *in general* when 5000 samples are provided per class, and the overall dataset contains millions of training samples. That rule seems to apply to our case as well. For example, the dataset provided by the organizers of the Challenge contains over five thousand samples of healthy sinus-rhythm samples for which mean of test F1 scores are above 0.90, but considering that the whole training dataset contains only 8528 samples in total, it implies that getting deep neural networks to work requires a few workarounds. The usual recipe for training classifiers containing tens of millions of parameters

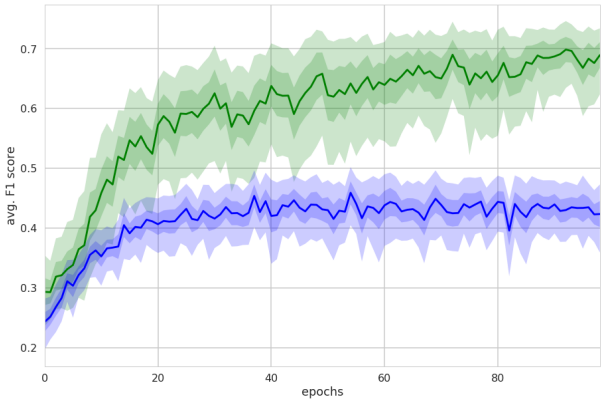


Figure 7: Results on running 18 different network architectures on a manually designed difficult training/test set separation (Blue), and on a random split in average (Green)

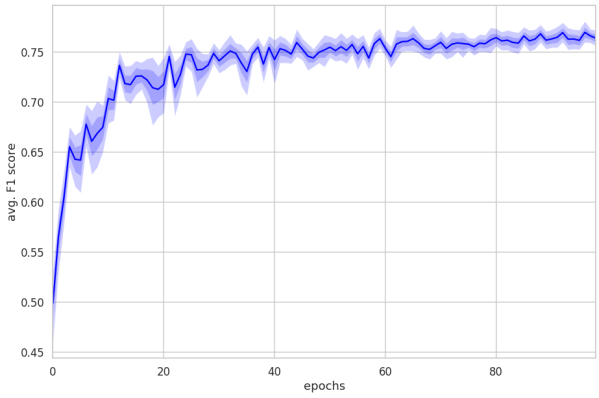


Figure 8: Test results computed from 10-fold cross validation using the best performing model.

tells us that it is beneficial to find a model first that can be at least over-fitted on the training data, and then incrementally decrease degree of complexity by removing layers and shrinking the number of convolutional filters, while applying more and more regularization to avoid high variance. During the development phase we observed that by the time the network could not be improved further on the training set, it had already dropped in performance on the test set. Data preparation, training and evaluating scripts required to reproduce the following results are available at our open-source repository: `https://github.com/botcs/itk-physionet-17`

B. PERFORMANCE ANALYSIS

Instability. When we manually set random seeds for separating samples into training-test-evaluation classes, we came across a very interesting phenomenon: one can construct a train-evaluation set separation such that outcome can be extremely encouraging, over avg. F1=0.9 score, and for another choice, it could lead to much worse results, avg. F1=0.5, having every other detail of the training environment fixed. When we retrained 18 randomly chosen networks from our baseline evaluations on a training set which probably covered all problematic samples that may lie close to the decision boundary, independently of the underlying architecture results got better. Using exactly the same environment, with the same initialization of weights on a different choice of training samples, test performance seemed to have an upper bound - even our best-performing networks could not perform better than F1=0.6. For comparison see Figure 7. In Figure 8, we analyze how sensitive is our most robust model: we do 10-fold cross-validation, where each training is evaluated with two different seeds to eliminate the noise coming from random initializing the network.

18 networks initialized and optimized with the same fixed random seed on different training sets, which we have found randomly splitting the original dataset, reveals the greatest challenge in AF detection; namely, there is a tiny set of specific samples that is crucial for generalization.

Many competitors reportedly have overcome that issue by applying a common technique used in deep learning competitions: final prediction is not evaluated by a single network, but by ensembles of neural networks to make the algorithm more robust against the biased dataset. In groups where expert supervision of training data set was available, significant improvement was reported when ambiguous samples were removed from the already moderate-sized data set.

Note: The bands correspond to cover 99% and 68% of test scores and show that when certain training samples are missing, networks fail to generalize, and they are only able to reach a test score F1=0.45 with a small deviation, making it almost impossible to measure improvements on different architectures and optimization methods. At the same time, when essential training examples are available, we have space for experiments, shown by the increased width of the band.

avg. F1 score	0.70	0.72	0.73	0.72	0.76	0.78	0.74	0.74	0.75	**0.81**	0.73
SELU			✓			✓			✓	✓	✓
Dilation		✓	✓		✓	✓		✓	✓	✓	✓
SqueezeNet 1.1	✓	✓	✓								
VGG 19				✓	✓	✓					
StanfordECG							✓	✓	✓		
EncodeNet (ours)										✓	
SkipFCN (ours)											✓
Raw signal	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Log-spectrogram											

Table 2: The average F1 score results in time domain

avg. F1 score	0.65	0.63	0.63	0.69	0.74	0.75	0.53	0.51	0.54	0.68	**0.79**
SELU			✓			✓			✓	✓	✓
Dilation		✓	✓		✓	✓		✓	✓	✓	✓
SqueezeNet 1.1	✓	✓	✓								
VGG 19				✓	✓	✓					
StanfordECG							✓	✓	✓		
EncodeNet (ours)										✓	
SkipFCN (ours)											✓
Raw signal											
Log-spectrogram	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table 3: The average F1 score results in spectral domain

In Table 2 a summary of incremental improvements achieved on original time-series is visible. Among over 100 different alternative architectures, here we list the most revealing and instructive experiences of ours.

In Table 3 the average F1 score results is visible in the spectral domain. Despite that different paths led us to the architecture of *SkipFCN*, the main drive was to achieve comparable performance to time-domain models since input space was heavily redundant. For keeping our evaluations symmetric and providing a reference for evaluating identical architectures on the spectral domain, we list results of the same networks shown in Table 2.

Computational requirements. The delay and complexity of our proposed model, depicted in Figure 1 is mainly characterized by the Fourier transform, and the complexity of the two parallel branches, working on time and frequency domain respectively. In total the model has fewer than 2 million trainable parameters, that takes up 8 MB storage using 32-bit precision representation. We performed extensive time complexity measurements on the proposed model, using a single P100 Graphics Processor Unit. For the training, we used the mini-batch size of 64 with early-stopping after 100 epochs on the training data. On average, each training took 1 hour before reaching a peak of accuracy. In real-time operation (inference) using Fourier Transform window size of 255, the model requires 354 measurement points to evaluate the first response. Based on the sampling rate of the recording device (300 Hz) that translates to a 1.18s effective delay. Furthermore, we measured how fast the model evaluates a 20s window. Without computing samples in parallel, the time-domain branch takes 1.33 ms $\pm$ 821 ns and the frequency-domain branch takes 1.54 ms $\pm$ 1.48 μs to extract features, in total the inference takes 1.68 ms $\pm$ 2.11 μs. Using a batch size of 100 we can parallelize the computations with a sub-linear increase in time cost: 18.6 ms $\pm$ 21.9 μs on time-domain, 32.1 ms $\pm$ 127 μs on the frequency domain, 39.1 ms $\pm$ 482 μs when branches are computed in parallel.

Primerjava osnovnega algoritma po vzoru obnašanja netopirjev in njegove hibridne različice HBA

Žan Grajfoner
Fakulteta za elektrotehniko,
računalništvo in informatiko
Koroška cesta 46
2000 Maribor, Slovenija
zan.grajfoner@student.um.si

Lucija Brezočnik
Fakulteta za elektrotehniko,
računalništvo in informatiko
Koroška cesta 46
2000 Maribor, Slovenija
lucija.brezocnik@um.si

POVZETEK

V prispevku smo se osredotočili na algoritme inteligence roja, ki črpajo navdih iz obnašanja roja živali v naravi. Primerjali smo osnovni algoritem po vzoru obnašanja netopirjev in njegovo hibridno različico. Raziskali smo razlike med osnovnima arhitekturama obeh algoritmov, pripadajoče parametre, kot tudi nekaj drugih hibridnih različic algoritma po vzoru obnašanja netopirjev. Primerjavo smo izvedli na praktičnem primeru optimizacije desetih testnih funkcij na treh različnih dimenzijah problema (10, 20, 30). Prav tako smo raziskali vpliv različnih velikosti populacije (20, 30, 50) pri obeh algoritmih. Ugotovili smo, da so rezultati optimizacije hibridne različice algoritma boljši od osnovne različice algoritma.

Ključne besede

algoritem po vzoru obnašanja netopirjev, evolucijski algoritmi, hibridizacija, inteligenca roja, računska inteligenca

1. UVOD

Algoritmi po vzoru iz narave posnemajo delovanje različnih naravnih in bioloških sistemov [17, 3]. Večinoma se uporabljajo za reševanje kompleksnih optimizacijskih problemov [3], kot so na primer sestava urnikov, izbira najboljše lokacije za postavitev antene in iskanje najkrajše poti na grafu. Med tovrstne algoritme uvrščamo tudi algoritme inteligence rojev [9], ki so dandanes prisotni v številnih realnih aplikacijah [8].

Inteligenca roja (angl. Swarm Intelligence, krajše SI) [1, 2] predstavlja študije kolektivnega obnašanja množičnih sistemov, ki se razvijajo s časom. Takšne sisteme v naravi tvorijo razni insekti (npr. čebele in mravlje) in druge živali (npr. ptice in ribe). Predstavimo primer inteligence roja na čebelah [7]. Čebela sledi splošnim in enostavnim pravilom, vendar lahko v roju z drugimi čebelami rešuje kompleksnejše naloge, kot je na primer izbira prave lokacije za njihov novi dom. Več izvidniških čebel odleti na iskanje potencialne lokacije, vendar dokončno odločitev o najprimernejši lokaciji sprejmejo skupaj.

V prispevku, ki je povzetek diplomskega dela [6], se bomo osredotočili na algoritem po vzoru obnašanja netopirjev (angl. Bat Algorithm, krajše BA), ki ga uvrščamo v skupino SI. Obstoječe študije so pokazale, da lahko algoritem BA hitreje doseže boljše rezultate pri reševanju optimizacijskih problemov glede na njegove predhodnike, kot je na primer genetski algoritem [10]. Algoritem BA posnema naravni fenomen netopirjev – eholokacijo. Netopirji med letenjem spuščajo visoke zvoke, kakršnih človeško uho ne zaznava. Zvok potuje skozi zrak kakor val in se odbije od vsakega objekta. Netopirji poslušajo odbiti zvok in na podlagi tega določijo oddaljenost, velikost ter obliko predmeta.

Ker osnovni algoritem BA ni primeren za reševanje vseh družin problemov, so različni avtorji predlagali več hibridnih različic, ki razširjujejo osnovni algoritem BA z mehanizmi drugih algoritmov [15, 11, 13]. Prav tako nekateri avtorji navajajo različne nastavitve nadzornih parametrov, kot so na primer velikost populacije in minimalna/maksimalna frekvenca.

Glavni cilj prispevka je primerjava osnovnega algoritma BA, ki ga je predlagal Yang [16], in hibridnega algoritma po vzoru obnašanja netopirjev (angl. Hybrid Bat Algorithm, krajše HBA), ki ga so ga predlagali Fister in ostali [5]. Algoritma bomo uporabili pri optimizaciji več testnih funkcij in s pomočjo eksperimenta skušali odgovorili na dve raziskovalni vprašanji (RV).

RV1: Ali hibridni algoritem po vzoru obnašanja netopirjev dosega boljše rezultate kot osnovni algoritem po vzoru obnašanja netopirjev pri večji velikosti populacije?

RV2: Ali hibridni algoritem po vzoru obnašanja netopirjev dosega boljše rezultate kot osnovni algoritem po vzoru obnašanja netopirjev pri optimizaciji funkcij višjih dimenzij?

V drugi sekciji predstavimo algoritem BA in njegovo hibridno različico, tretja sekcija je namenjena orisu zasnove eksperimenta, analiza dobljenih rezultatov pa je zbrana v četrti sekciji. Prispevek strnemo v peti sekciji in podamo smernice za prihodnji razvoj.

DOI: https://doi.org/10.26493/978-961-7055-82-5.85-90

2. ALGORITMI PO VZORU OBNAŠANJA NETOPIRJEV

V sekciji 2.1 predstavimo osnovno različico algoritma BA, v sekciji 2.2 njegovo hibridno različico, sekcija 2.3 pa zajema predstavitev nekaterih drugih hibridnih različic BA.

2.1 Algoritem po vzoru obnašanja netopirjev

Algoritem po vzoru obnašanja netopirjev spada v družino algoritmov SI in ga je leta 2010 razvil Xin-She Yang na univerzi v Cambridgeu [16]. Njegov namen je bil razviti enostaven in učinkovit algoritem, ki bi bil primeren za uporabo v različnih aplikacijah za reševanje optimizacijskih problemov. V algoritmu je poskušal posnemati pojav eholokacije pri mikronetopirjih. Originalni algoritem BA obravnava netopirje kot roj, ki se pomika po prostoru in išče svoj plen. Obnašanje netopirjev je kompleksno in zahtevno, zaradi česar je neposredni prenos v t. i. računalniški sistem prezahteven. Zato je njihovo obnašanje opisal v algoritmu z naslednjimi poenostavljenimi pravili [4]:

1. Vsi netopirji uporabljajo eholokacijo za spremljanje razdalje do ciljnih objektov.
2. Netopirji pri iskanju hrane letijo naključno s hitrostjo v_i na pozicijo x_i s fiksno frekvenco f_{min}, variabilno valovno dolžino λ_i in glasnostjo A_i. Samodejno lahko prilagajajo valovno dolžino (ali frekvenco) pulzov, ki jih oddajajo, in prilagajajo raven oddajanja pulzov $r\_i \in [0, 1]$ glede na bližino cilja.
3. Glasnost variira od največje (pozitivne) A_0 do minimalne vrednosti A_{min}.

BA vzdržuje populacijo virtualnih netopirjev med delovanjem. Vsak položaj netopirja predstavlja rešitev problema in vsaka rešitev problema je predstavljena kot vektor realnih števil (enačba 1):

$$x^{(t)} = \{x_{i,1}^{(t)}, \ldots, x_{i,D}^{(t)}\}, \text{za } i = 1, \ldots, Np, \tag{1}$$

v kateri D označuje dimenzijo problema, t predstavlja trenutno generacijo, Np pa število virtualnih netopirjev v populaciji. Vsak vektor določa položaj virtualnega netopirja v iskalnem prostoru. Netopirji se gibajo v območju proti učinkovitejšim rešitvam in pri tem odkrivajo nova obetavna področja. Algoritem BA omogoča dve strategiji preiskovanja.

Prvo strategijo za gibanje netopirjev prikazujejo naslednje enačbe 2:

$$\begin{aligned} f_i^t &= f_{min} + (f_{max} - f_{min})\beta, \\ v_i^{t+1} &= v_i^t + (x_i^t - x_{best}^t) f_i^t, \\ x_i^{t+1} &= x_i^t + v_i^{(t+1)} \end{aligned} \tag{2}$$

V intervalu $f_i^t \in (f_{max}, f_{min})$ se spreminja frekvenca pulza. Iz uniformne porazdelitve $\beta \in [0, 1]$ se generira naključno število, ki definira velikost izhodnega pulza. Parameter x_{best}^t definira trenutno najboljšo globalno pozicijo oziroma rešitev.

Druga strategija izboljša trenutni položaj virtualnega netopirja po enačbi 3:

$$x^t = x_{best}^t + \epsilon A_i^t \varphi, \tag{3}$$

v kateri $\varphi \in [0, 1]$ definira naključno število iz Gaussove distribucije s srednjo vrednostjo 1 in standardnim odklonom 0, $\varphi > 0$ je skalirni faktor in A_i^t predstavlja glasnost.

Ta strategija je namenjena izkoriščanju najboljše rešitve, saj je usmerjena k raziskovanju območja najboljše rešitve in ponazarja vrsto naključnega sprehoda (angl. Random Walk Direct Exploitation, krajše RWDE) [4]. Algoritem BA nadzorujeta dva parametra in sicer stopnja pulza r_i in glasnost A_i. Matematično lahko to prikažemo z enačbo 4:

$$A_i^{(t+1)} = \alpha A_i^t, r_i^t = r_i^0[1 - exp(-\gamma\epsilon)], \tag{4}$$

v kateri α in γ predstavljata konstante. Obe vplivata na stopnjo konvergence algoritma [16].

Psevdokoda algoritma BA je prikazana v algoritmu 1.

Algoritem 1 Algoritem BA

Vhod: populacija $\mathbf{x_i} = (x_{i,1}, x_{i,2}, \ldots, x_{i,D})$ for $i = 1 \ldots Np$, MAX_FE.
Izhod: najboljša rešitev x_{best} in njena vrednost f_{min}

```
inicializacija_populacije;
eval = oceni_novo_populacijo;
f_min = najdi_najboljšo_rešitev (x_best);
while ustavitveni_pogoj_ni_dosežen do
    for i = 1 to Np do
        generiraj_novo_rešitev (x_i);
        if rand(0,1) > r_i then
            y = izboljšaj_najboljšo_rešitev (x_best);
        end if
        f_new = oceni_novo_rešitev (y);
        if f_new ≤ f_i and rand(0,1) < A_i then
            x_i = y;
            f_i = f_new;
        end if
        f_min = poišči_najboljšo_rešitev (x_best);
    end for
end while
```

2.2 Hibridni algoritem po vzoru obnašanja netopirjev

Hibridni algoritem po vzoru obnašanja netopirjev je eden od prvih hibridnih različic BA. HBA je hibridiziran z mutacijskimi strategijami diferencialne evolucije [12]. Strategija lokalnega iskanja osnovnega algoritma BA je zamenjana s strategijo mutacije DE, ki je prikazana v algoritmu 2.

V algoritmu HBA se na začetku izberejo tri naključne rešitve v populaciji. S pomočjo diferencialne evolucije "DE/rand/1/

bin" generiramo novega potomca y. Izbran potomec je podvržen delovanju normalne selekcije v algoritmu BA [4].

Algoritem 2 Lokalno iskanje v algoritmu HBA

```
if rand(0,1) > r_i then
    r_{i=1...3} = rand(0,1) * Np + 1;
    n = rand(1, D);
    for i = 1 to D do
        if rand(0,1) < CR or n == D then
            y_n = x_{r1,n} + F * (x_{r2,n} + x_{r3,n});
            n = (n + 1)%(D + 1);
        end if
    end for
end if
```

2.3 Primeri ostalih različic hibridnega BA

Prednost BA je, da lahko zelo hitro doseže konvergenco v začetni fazi. Kadar je potrebna hitra rešitev, se algoritem BA izkaže kot zelo učinkovit. Če pa mu dovolimo prehiter preklop v fazo izkoriščanja, tako da se glasnost A in impulz r prehitro spremenita, lahko to povzroči stagnacijo v začetni fazi. Za izboljšanje učinkovitosti in adaptacije osnovnega algoritma so številni avtorji ustvarili nekaj različic. Kot primer lahko omenimo avtorje v [15], ki so hibridizirali osnovni algoritem BA z algoritmom iskanja harmonij (angl. Harmony Search), avtorji v [11] so hibridizirali osnovni algoritem BA z algoritmom ABC (angl. Artificial Bee Colony), avtorji v [13] pa so hibridizirali osnovno različico algoritma BA z algoritmom kresnic (angl. Firefly Algorithm).

3. OPIS EKSPERIMENTOV

V sekciji opišemo vzpostavitev razvojnega okolja in vključitev programske knjižnice NiaPy ter definiramo testne funkcije, ki smo jih uporabili v eksperimentih.

3.1 Vzpostavitev razvojnega okolja

Program je bil napisan v odprtokodnem programskem jeziku Python v integriranem razvojnem okolju PyCharm (IDE: Jetbrains). Pri razvoju smo uporabili odprtokodno knjižnico Python microframework for building nature-inspired algorithms (krajše NiaPy) [14], katere namen je enostavnejša in hitrejša uporaba algoritmov po vzorih iz narave.

3.2 Testne funkcije

Eksperimente smo izvedli na naslednjih desetih različnih testnih funkcijah [14]: Ackley, Griewank, Levy, Rastrigin, Ridge, Rosenbrock, Salomon, Schwefel, Sphere in Whitley.

4. REZULTATI

V nadaljevanju je prikazana statistika zagonov testnih funkcij. V stolpcu *Alg.* imamo kratice imen algoritmov, ki jih testiramo, in sicer algoritma BA in HBA. V stolpcu *Nast.* so prikazane nastavitve parametrov, ki smo jih spreminjali pri zagonih eksperimentov. Spreminjali smo dva parametra: dimenzijo testnih funkcij in velikost populacije. D predstavlja dimenzijo problema testnih funkcij. Testne funkcije smo zaganjali na treh dimenzijah (10, 20, 30). Višja kot je dimenzija, zahtevnejša je optimizacija. Velikost populacije predstavlja število kandidatnih rešitev, ki so uporabljene v iskalnem prostoru. Uporabili smo tri različne velikosti populacij (20, 30, 50). Preostale vrednosti nadzornih parametrov so: $Np = y$, $A = 0{,}5$, $r = 0{,}5$, $f_{min} = 0{,}0$, $f_{max} = 2{,}0$, $F = 0{,}5$ in $CR = 0{,}9$.

Vsako konfiguracijo smo zagnali 25-krat, ustavitveni pogoj pa je bil 1000 * D ovrednotenj funkcije uspešnosti.

Stolpec *Min* predstavlja najboljšo dobljeno rešitev, stolpec *Max* predstavlja najslabšo dobljeno rešitev, stolpec *Mean* pa predstavlja povprečje dobljenih rešitev. Stolpec *Median* predstavlja središče med zgornjo in spodnjo mejo dobljenih rešitev, stolpec *Std* pa predstavlja izračun standardnega odklona iz njih.

Tabela 1: Rezultati Ackley

Alg.	Nast.	Min	Max	Mean	Median	Std
BA	D=10,	14,304	18,630	17,003	17,116	1,187
HBA	Np=20	2E-06	2,013	0,782	1,155	0,809
BA	D=10,	13,772	18,273	16,701	17,002	1,189
HBA	Np=30	2E-06	3,404	0,908	1,155	1,035
BA	D=10,	10,837	18,760	16,925	17,192	1,607
HBA	Np=50	2E-06	2,580	0,999	1,155	0,987
BA	D=20,	15,997	18,960	17,970	18,034	0,690
HBA	Np=20	2,171	5,473	3,494	3,490	0,935
BA	D=20,	16,959	19,360	18,032	18,154	0,679
HBA	Np=30	1,155	5,240	3,240	3,222	1,020
BA	D=20,	16,553	19,120	17,861	17,948	0,710
HBA	Np=50	4E-07	5,650	3,070	3,223	1,203
BA	D=30,	15,879	18,940	17,950	18,062	0,650
HBA	Np=20	3,222	8,236	5,300	4,919	1,504
BA	D=30,	17,211	19,360	18,202	18,162	0,529
HBA	Np=30	2,887	10,020	6,021	6,182	1,831
BA	D=30,	17,255	19,440	18,135	18,003	0,510
HBA	Np=50	2,580	9,475	5,857	5,503	1,913

Tabela 2: Rezultati Griewank

Alg.	Nast.	Min	Max	Mean	Median	Std
BA	D=10,	1,856	4,835	3,244	3,260	0,757
HBA	Np=20	0,027	0,577	0,151	0,140	0,115
BA	D=10,	1,575	4,887	3,122	3,064	0,788
HBA	Np=30	1E-11	0,344	0,122	0,116	0,069
BA	D=10,	1,736	4,698	3,083	3,217	0,876
HBA	Np=50	0,027	0,310	0,124	0,106	0,074
BA	D=20,	3,159	10,540	6,852	6,640	1,678
HBA	Np=20	5E-14	0,101	0,033	0,025	0,035
BA	D=20,	4,725	9,525	6,718	6,560	1,242
HBA	Np=30	2E-14	0,096	0,027	0,020	0,025
BA	D=20,	3,953	11,700	7,220	7,249	1,741
HBA	Np=50	1E-13	0,118	0,026	0,020	0,028
BA	D=30,	5,226	18,170	10,970	10,501	3,019
HBA	Np=20	1E-12	0,127	0,031	0,025	0,035
BA	D=30,	6,596	16,460	10,993	10,982	2,738
HBA	Np=30	2E-12	0,107	0,025	0,015	0,027
BA	D=30,	7,005	15,310	10,785	10,685	2,348
HBA	Np=50	1E-12	0,069	0,020	0,012	0,021

Iz vseh zagonov optimizacij testnih funkcij (tabele 1-10) je razvidno, da je hibridna različica algoritma po vzoru obnašanja netopirjev učinkovitejša od osnovne različice algoritma. Naša raziskava je tako potrdila že obstoječe raziskave [5].

Dodatno smo preizkusili delovanje obeh algoritmov z raz-

Tabela 3: Rezultati Levy

Alg.	Nast.	Min	Max	Mean	Median	Std
BA	D=10,	8E-07	1,002	0,190	0,032	0,320
HBA	Np=20	3E-15	0,182	0,022	4E-14	0,047
BA	D=10,	5E-07	1,793	0,249	0,005	0,475
HBA	Np=30	4E-16	0,851	0,041	4E-14	0,170
BA	D=10,	9E-07	1,215	0,124	0,003	0,307
HBA	Np=50	8E-16	0,942	0,049	1E-14	0,188
BA	D=20,	4E-06	4,734	0,797	0,0003	1,253
HBA	Np=20	2E-16	2,065	0,286	0,091	0,514
BA	D=20,	4E-06	3,430	0,683	1E-05	1,024
HBA	Np=30	4E-16	1,701	0,278	0,182	0,388
BA	D=20,	3E-06	4,842	0,776	2E-05	1,157
HBA	Np=50	2E-16	1,123	0,272	0,091	0,382
BA	D=30,	1E-05	6,243	1,706	1,715	1,446
HBA	Np=20	2E-14	2,246	0,637	0,272	0,676
BA	D=30,	8E-06	4,614	1,179	0,907	1,369
HBA	Np=30	1E-14	2,065	0,466	0,272	0,525
BA	D=30,	1E-05	5,005	1,094	0,851	1,473
HBA	Np=50	6E-15	2,156	0,496	0,272	0,586

Tabela 4: Rezultati Rastrigin

Alg.	Nast.	Min	Max	Mean	Median	Std
BA	D=10,	15,920	77,610	45,052	44,773	15,620
HBA	Np=20	5,970	31,840	17,073	16,914	7,416
BA	D=10,	18,904	80,590	48,236	47,758	19,660
HBA	Np=30	5,970	38,800	16,312	14,742	6,937
BA	D=10,	18,904	79,600	41,828	38,804	16,340
HBA	Np=50	5,970	28,850	16,079	15,919	4,880
BA	D=20,	44,775	153,200	87,000	81,588	26,970
HBA	Np=20	17,909	62,680	42,107	45,768	13,240
BA	D=20,	36,815	161,200	88,712	87,558	31,680
HBA	Np=30	21,889	80,590	49,827	47,758	14,730
BA	D=20,	24,876	170,100	75,339	70,643	32,670
HBA	Np=50	22,884	91,540	48,992	47,758	14,730
BA	D=30,	67,661	242,800	143,120	148,250	46,420
HBA	Np=20	38,803	137,400	78,647	76,612	28,770
BA	D=30,	82,584	241,800	136,910	120,390	47,600
HBA	Np=30	42,783	130,300	73,348	74,622	21,150
BA	D=30,	59,700	222,900	126,680	123,380	37,490
HBA	Np=50	38,803	134,400	77,450	77,607	20,570

Tabela 5: Rezultati Ridge

Alg.	Nast.	Min	Max	Mean	Median	Std
BA	D=10,	7,5E2	6,5E6	4,1E6	4,3E6	1,5E6
HBA	Np=20	3E-07	0,002	0,0002	4E-05	4E-04
BA	D=10,	1E6	7,5E6	4,1E6	4E6	1,6E6
HBA	Np=30	2E-06	0,004	0,0003	3E-05	9E-04
BA	D=10,	1,2E6	8.9E6	4,1E6	3,5E6	2E6
HBA	Np=50	3E-06	2E-04	4E-05	2E-05	5E-05
BA	D=20,	1,9E6	2,5E7	1,4E7	1,4E7	6,6E6
HBA	Np=20	0,0085	4,181	0,397	0,105	0,862
BA	D=20,	5,2E6	2,9E7	1,7E7	1,7E7	6E6
HBA	Np=30	0,0082	2,566	0,302	0,071	0,549
BA	D=20,	5,8E6	2,4E7	1,5E7	1,5E7	4,7E6
HBA	Np=50	0,009	1,668	0,283	0,129	0,423
BA	D=30,	1,1E7	7,9E7	3,4E7	3,2E7	1,6E7
HBA	Np=20	0,464	2,1E6	90,873	5,618	409,032
BA	D=30,	9,5E6	6,6E7	3E7	2,8E7	1,3E7
HBA	Np=30	0,253	67,270	10,771	8,240	12,940
BA	D=30,	8,9E6	4,9E7	2,8E7	2,6E7	1,1E7
HBA	Np=50	0,414	31,860	9,629	4,938	8,573

Tabela 6: Rezultati Rosenbrock

Alg.	Nast.	Min	Max	Mean	Median	Std
BA	D=10,	1,9E8	4E7	1E7	7E6	1E7
HBA	Np=20	6E-05	100,789	7,077	2,633	19,636
BA	D=10,	5E8	2E7	1E7	9E6	6E6
HBA	Np=30	0,0065	7,903	3,641	3,430	2,340
BA	D=10,	2E6	3E7	1E7	8E6	7E6
HBA	Np=50	0,0133	65,412	5,365	2,524	12,642
BA	D=20,	2E7	1E8	5E7	5E7	2E7
HBA	Np=20	1,248	121,418	21,271	12,315	27,305
BA	D=20,	6E6	6E7	3E7	3E7	2E7
HBA	Np=30	0,495	78,608	12,600	10,839	14,593
BA	D=20,	3E6	7E7	4E7	4E7	2E7
HBA	Np=50	1,874	69,171	13,313	11,927	12,300
BA	D=30,	1E7	2E8	6E7	6E7	4E7
HBA	Np=20	4,326	567,123	61,398	22,023	109,953
BA	D=30,	2E7	2E8	7E7	6E7	4E7
HBA	Np=30	0,033	106,160	32,547	21,400	30,564
BA	D=30,	2E7	2E8	8E7	7E7	5E7
HBA	Np=50	9,143	74,178	29,963	24,145	20,055

ličnimi nastavitvami populacije. Eksperimenti so pokazali, da:

- velikost populacije pri obeh uporabljenih algoritmih nima enakega vpliva na rezultate optimizacije;
- ne moremo nedvoumno potrditi, da z večanjem populacije dosegamo boljše rezultate;
- je za vsako testno funkcijo potrebno eksperimentalno določiti optimalno nastavitev populacije, saj glede na rezultate ne moremo potrditi, da obstaja optimalna nastavitev za vse funkcije;
- je algoritem BA pri optimizaciji funkcij višjih dimenzij dosegel zelo slabe rezultate, kar so potrdile tudi že druge študije [5];
- se je algoritem HBA izkazal kot zelo učinkovit pri vseh nastavitvah parametrov. Najboljše rezultate je dosegel pri dimenzijah 10 in 20, pri nekaterih funkcijah pa tudi pri dimenziji 30;
- so bile pri funkcijah Rosenbrock, Sphere in Whitley razlike v rezultatih optimizacije največje, pri funkcijah Ackley (slika 1) in Rastrigin pa najmanjše;

Na podlagi dobljenih rezultatov in njihove analize lahko odgovorimo na zastavljeni raziskovalni vprašanji. Z večanjem populacije pri nekaterih testnih funkcijah dosežemo boljše rezultate, vendar pa to ne velja za vse primere. S tem smo odgovorili na RV1.

HBA dosega boljše rezultate na testnih funkcijah višjih dimenzij kakor njegov predhodnik BA. Razlog za to leži v hibridizaciji s strategijami DE, saj se osnovna različica algo-

Tabela 7: Rezultati Salomon

Alg.	Nast.	Min	Max	Mean	Median	Std
BA	D=10,	507E3	1054E3	828E3	800E3	158E3
HBA	Np=20	0,099	3,582	0,891	0,895	0,759
BA	D=10,	258E3	1603E3	954E3	948E3	353E3
HBA	Np=30	0,099	1,592	0,621	0,398	0,344
BA	D=10,	474E3	1390E3	870E3	871E3	247E3
HBA	Np=50	0,099	2,487	0,605	0,398	0,481
BA	D=20,	1207E3	3342E3	2153E3	2009E3	566E3
HBA	Np=20	0,895	14,326	4,740	3,582	3,006
BA	D=20,	901E3	3124E3	2260E3	2312E3	631E3
HBA	Np=30	0,895	12,038	4,000	2,487	3,171
BA	D=20,	1370E3	3732E3	2225E3	2137E3	641E3
HBA	Np=50	0,895	9,949	4,115	3,582	2,906
BA	D=30,	1950E3	5542E3	3,4E6	336E4	1E6
HBA	Np=20	4,875	39,790	10,912	8,059	7,293
BA	D=30,	265E4	666E4	396E4	386E4	970E3
HBA	Np=30	2,487	39,791	13,733	9,949	8,835
BA	D=30,	2,2E6	6E6	3,9E6	3,8E6	1E6
HBA	Np=50	3,582	35,911	12,070	9,949	8,059

Tabela 8: Rezultati Schwefel

Alg.	Nast.	Min	Max	Mean	Median	Std
BA	D=10,	1559E3	3213E3	2644E3	2707E3	354E3
HBA	Np=20	385E3	2024E3	1246E3	1308E3	481E3
BA	D=10,	2041E3	3101E3	2723E3	2757E3	249E3
HBA	Np=30	355E3	2026E3	1139E3	1108E3	408E3
BA	D=10,	2232E3	3223E3	2685E3	2697E3	270E3
HBA	Np=50	355E3	1906E3	1087E3	952E3	450E3
BA	D=20,	5414E3	6850E3	6164E3	6265E3	415E3
HBA	Np=20	2195E3	3852E3	2752E3	2696E3	424E3
BA	D=20,	5379E3	6607E3	6195E3	6233E3	298E3
HBA	Np=30	1782E3	3990E3	2893E3	2926E3	502E3
BA	D=20,	4477E3	6860E3	6088E3	6130E3	490E3
HBA	Np=50	1682E3	4051E3	2654E3	2645E3	541E3
BA	D=30,	8983E3	1059E4	9778E3	9805E3	350E3
HBA	Np=20	2338E3	6235E3	4483E3	4448E3	839E3
BA	D=30,	9025E3	1084E4	9997E3	10003E3	365E3
HBA	Np=30	3759E3	6319E3	4655E3	4602E3	732E3
BA	D=30,	8424E3	1051E4	9790E3	9780E3	528E3
HBA	Np=50	2694E3	5577E3	4455E3	4379E3	718E3

ritma hitro ujame v lokalni optimum. S tem smo odgovorili na RV2.

5. ZAKLJUČEK

V prispevku smo predstavili algoritem BA. Natančneje smo preučili eno izmed njegovih prvih različic, poimenovano HBA. V sklopu eksperimenta smo oba algoritma uporabili za optimizacijo desetih različnih testnih funkcij. Eksperimente smo zaganjali na različnih konfiguracijah, pri čemer smo spreminjali dimenzije problemov in velikost populacije. Ugotovili smo, da se je hibridna različica BA izkazala za uspešnejšo na vseh konfiguracijah nad vsemi testnimi funkcijami.

Za prihodnje delo bomo uporabili hibridno različico algoritma po vzoru obnašanja netopirjev v okviru realnih aplikacij, kot je na primer sestavljanje urnikov. Zanimiva bi bila tudi primerjava dobljenih rezultatov z rezultati drugih algoritmov, ki jih podpira knjižnica NiaPy.

Tabela 9: Rezultati Sphere

Alg.	Nast.	Min	Max	Mean	Median	Std
BA	D=10,	1,729	23,752	9,256	8,718	5,678
HBA	Np=20	8E-15	6E-12	6E-13	1E-13	1E-12
BA	D=10,	2,417	18,624	9,533	10,048	4,455
HBA	Np=30	1E-14	6E-12	8E-13	2E-13	1E-12
BA	D=10,	1,120	23,656	9,089	8,915	5,111
HBA	Np=50	9E-15	4E-12	7E-13	2E-13	1E-12
BA	D=20,	6,318	34,822	21,238	19,588	7,618
HBA	Np=20	2E-16	9E-14	2E-14	6E-15	3E-14
BA	D=20,	5,700	46,084	18,393	17,703	9,185
HBA	Np=30	5E-17	2E-13	5E-14	2E-14	6E-14
BA	D=20,	8,663	42,936	23,238	20,392	10,005
HBA	Np=50	1E-16	1E-12	1E-13	2E-14	3E-13
BA	D=30,	10,235	51,285	23,936	22,397	10,715
HBA	Np=20	2E-14	7E-12	1E-12	5E-13	2E-12
BA	D=30,	5,173	52,861	22,680	23,952	10,070
HBA	Np=30	1E-14	2E-11	2E-12	2E-13	4E-12
BA	D=30,	3,689	52,914	22,079	21,339	9,386
HBA	Np=50	3E-15	2E-11	2E-12	2E-13	5E-12

Tabela 10: Rezultati Whitley

Alg.	Nast.	Min	Max	Mean	Median	Std
BA	D=10,	2,8E8	1E8	1E7	2E6	2E7
HBA	Np=20	11E3	73E3	44E3	43E3	15E3
BA	D=10,	6,4E7	6E7	1E7	7E6	2E7
HBA	Np=30	19E3	62E3	44E3	46E3	11E3
BA	D=10,	1,3E7	4E7	7E6	4E6	1E7
HBA	Np=50	9E3	59E3	41E3	44E3	12E3
BA	D=20,	3,8E8	1E9	1E8	5E7	2E8
HBA	Np=20	71E3	328E3	249E3	250E3	51E3
BA	D=20,	3,9E8	5E8	9E7	6E7	1E8
HBA	Np=30	150E3	327E3	248E3	258E3	43E3
BA	D=20,	4,8E8	4E8	8E7	5E7	9E7
HBA	Np=50	124E3	343E3	240E3	248E3	52E3
BA	D=30,	3E6	2E9	3E8	2E8	3E8
HBA	Np=20	451E3	784E3	656E3	667E3	78E3
BA	D=30,	9E6	9E8	3E8	3E8	2E8
HBA	Np=30	408E3	855E3	663E3	663E3	98E3
BA	D=30,	8E6	6E8	2E8	2E8	1E8
HBA	Np=50	334E3	851E3	659E3	639E3	117E3

ZAHVALA

Raziskovalni program št. P2-0057 je sofinancirala Javna agencija za raziskovalno dejavnost Republike Slovenije iz državnega proračuna.

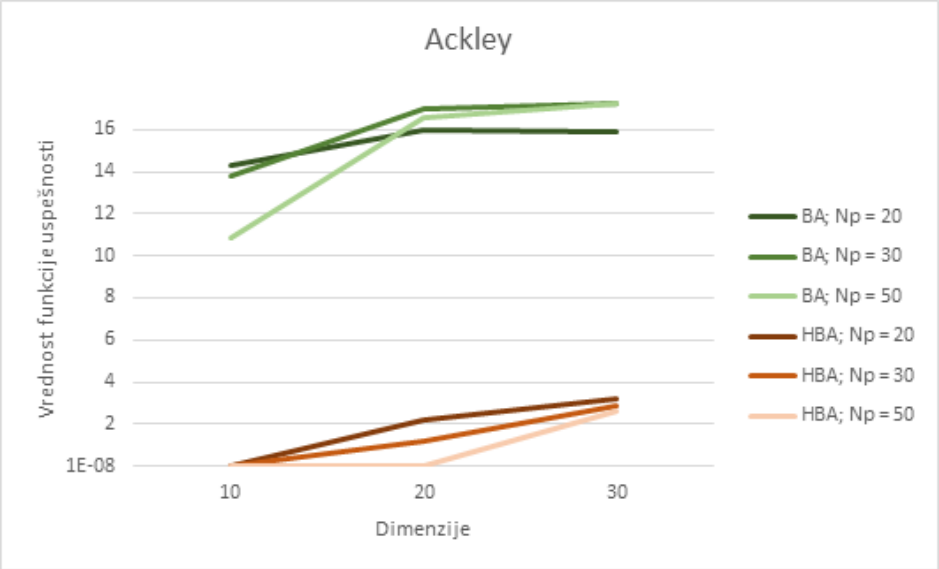
LITERATURA

[1] E. Bonabeau, D. d. R. D. F. Marco, M. Dorigo, G. Theraulaz, et al. *Swarm intelligence: from natural to artificial systems*. Number 1. Oxford university press, 1999.

[2] L. Brezočnik. Optimizacija z rojem delcev za izbiro atributov pri klasifikaciji. Master's thesis, University of Maribor, Slovenia, 2016.

[3] L. Brezočnik, I. Fister, and V. Podgorelec. Swarm intelligence algorithms for feature selection: a review. *Applied Sciences*, 8(9):1521, 2018.

[4] I. Fister Jr. *Algoritmi računske inteligence za razvoj*



Slika 1: Primerjava optimizacije testne funkcije Ackley glede na D in Np.

umetnega športnega trenerja. PhD thesis, University of Maribor, Slovenia, 2017.

[5] I. Fister Jr., D. Fister, and X.-S. Yang. A hybrid bat algorithm. *Elektrotehniški vestnik*, 80(1-2):1–7, 2013.

[6] Ž. Grajfoner. Primerjava različnih algoritmov po vzoru obnašanja netopirjev. University of Maribor, Slovenia, 2019.

[7] D. Karaboga. An idea based on honey bee swarm for numerical optimization. Technical report, TR06, Department of Computer Engineering, Engineering Faculty, Erciyes University, 2005.

[8] K. Ljubič and I. Fister Jr. Narava kot navdih računalništvu. *Življ. teh.*, 65(15):14–17, 2014.

[9] K. Ljubič and I. Fister Jr. Algoritem na osnovi obnašanja netopirjev. *Presek*, 42(3):26–28, 2015.

[10] S. Mirjalili, S. M. Mirjalili, and X.-S. Yang. Binary bat algorithm. *Neural Computing and Applications*, 25(3-4):663–681, 2014.

[11] J.-S. Pan, T.-K. Dao, M.-Y. Kuo, M.-F. Horng, et al. Hybrid bat algorithm with artificial bee colony. In *Intelligent Data analysis and its Applications, Volume II*, pages 45–55. Springer, 2014.

[12] A. K. Qin, V. L. Huang, and P. N. Suganthan. Differential evolution algorithm with strategy adaptation for global numerical optimization. *IEEE transactions on Evolutionary Computation*, 13(2):398–417, 2008.

[13] T. Sureshkumar, M. Lingaraj, B. Anand, and T. Premkumar. Hybrid firefly bat algorithm (hfba)–based network security policy enforcement for psa. *International Journal of Communication Systems*, 31(14):e3740, 2018.

[14] G. Vrbančič, L. Brezočnik, U. Mlakar, D. Fister, and I. Fister Jr. Niapy: Python microframework for building nature-inspired algorithms. *J. Open Source Softw.*, 3:613, 2018.

[15] G. Wang and L. Guo. A novel hybrid bat algorithm with harmony search for global numerical optimization. *Journal of Applied Mathematics*, 2013, 2013.

[16] X.-S. Yang. A new metaheuristic bat-inspired algorithm. In *Nature inspired cooperative strategies for optimization (NICSO 2010)*, pages 65–74. Springer, 2010.

[17] X.-S. Yang and M. Karamanoglu. Swarm intelligence and bio-inspired computation: An overview. In X.-S. Yang, Z. Cui, R. Xiao, A. H. Gandomi, and M. Karamanoglu, editors, *Swarm Intelligence and Bio-Inspired Computation*, pages 3 – 23. Elsevier, Oxford, 2013.

Nadgradnja algoritma FLORS za besednovrstno označevanje slovenskih besedil

Domen Kavran
Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko
Koroška cesta 46, 2000 Maribor, Slovenija
domen.kavran@student.um.si

Robi Novak
Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko
Koroška cesta 46, 2000 Maribor, Slovenija
robi.novak1@student.um.si

Jan Banko
Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko
Koroška cesta 46, 2000 Maribor, Slovenija
jan.banko@student.um.si

Rok Potočnik
Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko
Koroška cesta 46, 2000 Maribor, Slovenija
rok.potocnik4@student.um.si

Luka Pečnik
Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko
Koroška cesta 46, 2000 Maribor, Slovenija
luka.pecnik@student.um.si

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

POVZETEK

Besednovrstno označevanje je postopek razpoznavanja besednih vrst v besedilu. Algoritem FLORS učinkovito izvaja besednovrstno označevanje z lokalnim kontekstom posamezne besede. V članku smo algoritem FLORS nadgradili za besednovrstno označevanje slovenskega jezika. Izboljšavo smo dosegli z odstranitvijo morfoloških značilk, vezanih na angleški jezik. Uporabili smo tudi analizo poglavitnih komponent. Z opisano spremembo nabora značilk smo dosegli uspešnost 85,16 %. Ugotavljamo, da se algoritem lahko uporabi za označevanje slovenskega jezika.

Ključne besede

besednovrstno označevanje, klasifikacija, procesiranje naravnega jezika, jezikovne tehnologije

1. UVOD

Pri opazovanju samostojnih besed v besedilu naletimo na pojav večpomenskosti, katero lahko zmanjšamo s povezovanjem individualne besede v lokalni kontekst s sosednjimi besedami. Besednovrstno označevanje je postopek, katerega cilj je določitev besedne vrste besedi na podlagi konteksta, ki ga predstavljajo sosednje besede oz. značilke teh besed. Besednovrstni označevalnik je torej sistem, ki samodejno izvaja besednovrstno označevanje. Za praktično uporabo mora biti robusten, učinkovit, natančen in prenosljiv [8].

Poznamo dve vrsti besednovrstnih označevalnikov, in sicer označevalnike na osnovi pravil (angl. rule-based taggers) ter stohastične označevalnike (angl. stochastic taggers). Slednji dosegajo visoko stopnjo točnosti brez sintaktične analize vhodnega besedila in se odločajo predvsem na podlagi verjetnosti s pomočjo statističnih tabel, v katerih je znanje o besedah predstavljeno posredno, medtem ko označevalniki na osnovi pravil uporabljajo množico definiranih pravil, s pomočjo katerih določijo besedno vrsto posamezne besede. S temi pravili je znanje o besedah zapisano neposredno [10, 5, 4].

V [16] je opisan algoritem FLORS (angl. Fast, LOcal, Robust, Simple). Gre za besednovrstni označevalnik, ki zgradi kontekst in znanje o besedi, odvisno od njenega lokalnega okolja, namesto da bi iskal optimalno zaporedje besednih vrst za celotne stavke. Kontekst besede je sestavljen iz binarnih in numeričnih značilk, ki so skupaj z delovanjem samega algoritma podrobneje predstavljene v drugem poglavju.

Uspešnost algoritma FLORS nad slovenskim jezikom smo skušali izboljšati z novimi značilkami. V drugem poglavju predstavimo sorodna dela, po katerih smo se zgledovali, v tretjem poglavju pa opišemo lastno idejo in komponente, ki smo jih potrebovali za izvedbo samega eksperimenta. V zadnjem poglavju rekonstruiramo rezultate originalnega algoritma nad izbranim slovenskim korpusom in jih nato primerjamo z rezultati nadgrajene različice algoritma.

2. FLORS IN SORODNA DELA

Pristopi k označevanju besed s strojnim učenjem uporabljajo različne metode učenja. V [7] pristopajo z nevronskimi mrežami in dosežejo visoko točnost, vendar pa sta tako izračun značilk kot tudi učenje mreže časovno zelo zahtevna procesa. V drugih delih so uporabili preprostejše značilke in nekatere druge metode strojnega učenja.

Uporaba strojnega učenja po metodi s podpornimi vektorji (angl. Support Vector Machine, SVM) za besednovrstno označevanje je bila objavljena v [20]. Z omejitvijo na lokalni kontekst so avtorji dosegli hitro učenje in izvajanje modela. Na zbirki besedil Wall Street Journal (WSJ) so dosegli točnost označevanja 97,16 %. Težave so se pojavile ob aplikaciji za drugo domeno. To problematiko so reševala kasnejša dela.

V [6] so avtorji predstavili nov način za povečanje robustnosti besednovrstnega označevanja. Metoda deluje na podlagi dveh ločenih modelov (splošnega in domensko specifičnega), naučenih na istih podatkih, a z različnimi značilkami. Za učenje splošnega modela so uporabili n-grame, ki so se v be-

DOI: https://doi.org/10.26493/978-961-7055-82-5.91-99

sedilu pojavili vsaj trikrat, za učenje domensko specifičnega modela pa n-grame, ki so se pojavili najmanj enkrat. Pri dekodiranju se uporabita oba modela tako, da se izbrani model spreminja dinamično glede na trenutno vhodno poved. Pri tem je izbira modela odvisna od podobnosti vhodne povedi podatkom, ki so jih uporabili za učenje algoritma. Takšen pristop dinamične izbire modela se nato uporabi v kombinaciji z algoritmom besednovrstnega označevanja, ki deluje v enem prehodu od leve proti desni. Algoritem je bil naučen na zbirki Wall Street Journal (WSJ), za testiranje delovanja algoritma pa so avtorji uporabili korpuse iz sedmih različnih domen. Ob testiranju so dosegli povprečno točnost besednovrstnega označevanja 70,54 % [6].

Algoritem FLORS [16] se loti domenske adaptacije na drugačen način. Uči se na označenih podatkih izvorne domene, hkrati pa uporabi nekaj statističnih podatkov neoznačenega besedila iz ciljne domene. Algoritem deluje hitro, saj za učenje in izvajanje uporablja le omejeno okolico posamezne besede (v nadaljevanju: kontekst) in preproste značilke; statistični podatki iz neoznačenega besedila pa zahtevajo zgolj preštevanje pojavitev. Avtorji definirajo štiri skupine značilk (slika 1), ki jih je možno ovrednotiti z uporabo zgolj trenutne besede in vnaprej pripravljene statistike.

Prva skupina značilk se nanaša na bigrame in njihove pojavitve v učnem korpusu. Bigram predstavimo tako, da trenutni besedi določimo za predhodnika eno od 500 najpogostejših besed v korpusu. Število pojavitev posameznega bigrama logaritmično obtežimo in tako tvorimo 500 značilk. Vse manj pogoste besede obravnavamo kot identične in na enak način tvorimo še eno dodatno značilko, s čimer se izognemo ničelnemu vektorju.

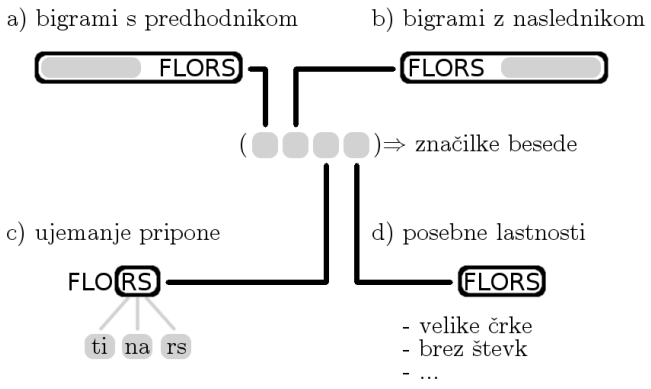
Podobno tvorimo drugo skupino značilk, le da tokrat trenutni besedi namesto predhodnika pripenjamo naslednika, kot vidimo na sliki 1(b). Značilke na osnovi pogostih sosedov doprinesejo koristno informacijo o besednih vrstah [17, 18].

Tretja skupina značilk je množica binarnih značilk za ujemanje pripone s priponami besed v neoznačenem besedilu. Drugače, kot je to storjeno v sorodnih delih [19, 14], FLORS uporabi vse pripone vseh besed. Tako se izognemo potrebi po izbiranju podmnožice pripon.

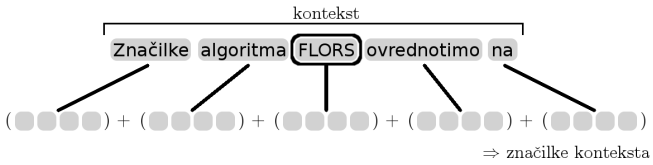
Nazadnje dodajo še značilke za 16 posebej izbranih lastnosti besede, kot so vsebovanje števk, vezajev in velikih začetnic, pa tudi ujemanje s pogostimi končnicami angleškega jezika.

Predstavljen postopek določi približno 100.000 značilk na besedo. Največji delež značilk sestavljajo značilke tretje skupine. Na sliki 2 je prikazano, kako iz značilk besed sestavimo vektor značilk konteksta. Značilke konteksta uporabimo za nadzorovano učenje klasifikacijskega modela. Avtorji uporabijo okno dolžine petih besed, torej učimo s 500.000 vhodnimi diskretnimi atributi. Model klasificira besede v razrede glede na njihovo slovnično besedno vrsto. Podobno kot v [20] in [6] tudi pri FLORS učimo po metodi *SVM* s pristopom One-Vs-All.

Na izvorni domeni so avtorji dosegli točnost 96,59 %, pri adaptaciji na različne ciljne domene pa med 89,44 % in 94,71 %.



Slika 1: Vektor značilk posamezne besede po algoritmu FLORS na primeru besede "FLORS". a) prva in b) druga skupina značilk so frekvence bigramov, ki jih sestavimo s trenutno besedo in pogostimi besedami učnega korpusa. Dodamo c) binarne značilke za ujemanje pripone in d) nekaj dodatnih značilk za posebne lastnosti besede.



Slika 2: Vektor značilk konteksta po algoritmu FLORS dobimo tako, da združimo vektorje značilk besed v lokalnem oknu. Prikazan je primer zaporedja besed "Značilke algoritma FLORS ovrednotimo na".

V našem delu prilagodimo algoritem FLORS za klasifikacijo besednih vrst v slovenskem jeziku, kar opišemo v naslednjem poglavju.

3. ALGORITEM

Nadgradnjo algoritma FLORS smo izvedli z dodatnimi značilkami, izbranimi za točnejše označevanje besednih vrst v slovenskem jeziku. Posebno pozornost smo namenili šumnikom in besednim priponam, ki so pogosto uporabljene v slovenskem jeziku, v angleščini pa niso prisotne.

Na podlagi analize besednih vrst in njihovih značilnosti v slovenščini smo izpeljali množico predpon in pripon, ki jih predstavimo v naslednjem podpoglavju.

Naša hipoteza je bila, da se bo z izpeljanimi značilkami točnost besednovrstnega označevanja nad slovenskimi besedili izboljšala.

3.1 Značilke

Obstoječim značilkam algoritma FLORS smo dodali binarne značilke, ki jih lahko razvrstimo v naslednje skupine:

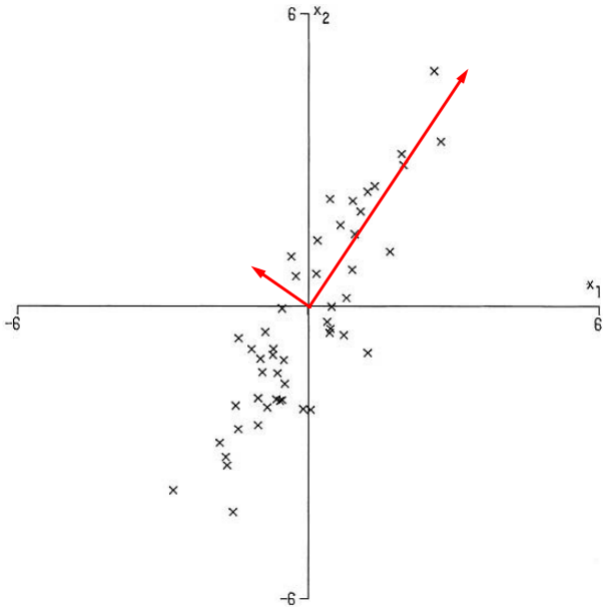
- **predponi *u* in *v*.** Predponi se lahko umestita na začetek različnih besednih vrst. Med glagoli lahko najdemo naslednje primere uporabe omenjenih predpon: u-krasti, u-pasti, v-plačati, v-pisati.
- **Predpone sestavljenk.** Predponi *od-* in *na-* se lahko uporabita na začetku sestavljenke (npr. od-dati, od-zgoraj, na-pisati, na-govoriti).
- **Pripone izpeljank.** Pripone *-lec*, *-ec*, *-arna* se lahko uporabijo kot končine sestavljenk (npr. bra-lec, pis-ec, knjig-arna). Pri izpeljavi del dvodelne podstave zamenjamo s priponskim obrazilom.
- **Končnice pridevnikov in pridevniških zaimkov.** Končnica *-ra* se pojavi pri pridevnikih (npr. dob-ra), končnica *-ja* pa se lahko pojavi pri pridevniških zaimkih (npr. mo-ja, tvo-ja).
- **Sprememba pri soglasniških premenih.** Za besede s končnico *-ci* se črka k v velelniku spremeni v črko c (npr. tekel – teci), kar imenujemo mehčanje ali palatizacija. Za besede s končnico *-zi* se črka g v velelniku spremeni v črko z (npr. vrgel – vrzi).
- **Končnice samostalniških besed.** Pri besedah ženskega spola ima samostalniška beseda končnico *-a*, pri srednjem spolu pa *-o*. Naslednji primeri so posamostaljene pridevniške besede: dobra, tista, dobro, tisto.
- **Končnice vrstnih pridevnikov.** Z končnicama *-ski* in *-ški* lahko razpoznamo vrstne pridevnike (npr. fotograf-ski, potaplja-ški).
- **Obrazilo v kombinaciji s podstavo.** Pri besedotvorju se lahko končnici *-ica* in *-ost* pojavita kot obrazilo, skupaj s podstavo pa tvorita novo besedo. Uporabili smo desna obrazila (npr. miz-ica, mlad-ost).
- **Pripone besedotvorne vrste sklop.** Končnice *-to*, *-tem*, *-deset* se lahko pojavijo kot pripone pri besedotvorni vrsti sklop, ki nastane s sklaplanjem, kjer posamezne dele večdelne podstave združimo v novo besedo (npr. na-to, po-tem, dva-in-tri-deset).
- **Naglasna, naslonska ali navezna sklonska oblika.** Končnice *-ga*, *-ne*, *-me* se pojavijo kot pripone naglasnih, naslonskih ali naveznih sklonskih oblik osebnega zaimka (npr. nje-ga, me-ne, na-me).
- **Pripone kazalnih zaimkov.** Priponi *-ta* in *-ti* se pojavita pri kazalnih zaimkih (npr. ta, tis-ti).
- **Pripone vprašalnih zaimkov.** Končnice *-aj*, *-em*, *-im*, *-en* in *-od* se pojavijo v vprašalnih zaimkih (npr. k-aj, kater-em, kater-im, kakš-en, k-od).
- **Šumniki** Posebnost slovenskega jezika so šumniki *č*, *š* in *ž*.

Binarne značilke predstavljajo prisotnost predpon, pripon in šumnikov. Opisane značilke zajemajo morfološke značilnosti slovenskega jezika in jih uvrščamo med oblikovne (angl. shape) značilke algoritma FLORS. Obdržali smo ortografski del značilk (ali beseda vsebuje številko, poševnico, veliko začetnico). Izpostaviti moramo dejstvo, da se zaradi kompleksnosti gramatike slovenskega jezika nekatere značilke pojavijo tudi pri vrstah besed, za katere niso bile v osnovi zasnovane.

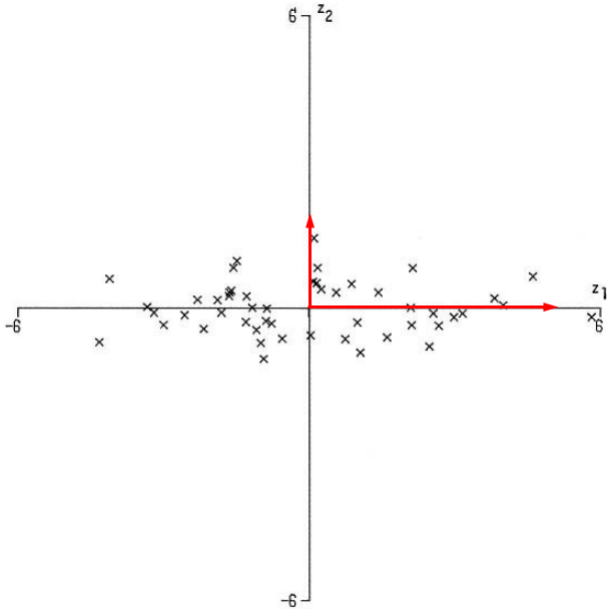
3.2 Analiza poglavitnih komponent

Osrednji namen analize poglavitnih komponent (angl. Principal Component Analysis, PCA) je zmanjšanje števila dimenzij množice podatkov, ki jih sestavlja veliko število koreliranih spremenljivk, ne da bi pri tem okrnili izraznost podatkov. Učinek je dosežen s pomočjo transformacije spremenljivk v novo množico, ki jo imenujemo množica poglavitnih komponent (angl. principal components). Poglavitne komponente med seboj niso odvisne, ohranjajo pa kar največjo stopnjo raznolikosti podatkov, ki je prisotna v originalnih spremenljivkah, s čimer zagotovijo, da je z njihovo pomočjo podatke med seboj možno ustrezno razlikovati. Tako za ceno nekaj točnosti dobimo enostavnejšo predstavitev, s čimer poenostavimo analizo in obdelavo podatkov [12].

Naj bo x vektor p naključnih spremenljivk. V procesu določanja poglavitnih komponent nas zanimajo raznolikosti in korelacije med spremenljivkami. Če vrednost p ni majhna, ni smiselno pregledovati vseh možnih povezav med spremenljivkami x, zato se raje osredotočimo na nekaj ($\ll p$) izpeljanih spremenljivk, ki vsebujejo kar največ informacij v zvezi z raznolikostjo, korelacijo in kovarianco med originalnimi spremenljivkami. PCA izpelje spremenljivke, odvisno od tipa podatkov, na podlagi kovariančne ali korelacijske matrike, ki opisujeta, na kakšen način so izvorne spremenljivke med seboj povezane. Gre za simetrični matriki, velikosti $p \times p$. Izpeljane spremenljivke so linearne kombinacije izvornih spremenljivk in so med seboj nekorelirane, torej gre za ortogonalne vektorje, ki predstavljajo smeri, v katerih se kaže največja raznolikost podatkov. Te kombinacije oz. poglavitne komponente so izpeljane in urejene na način, da se



Slika 3: Graf originalnih podatkov za spremenljivki x_1 in x_2 z označenima poglavitnima komponentama. Vir: [12]



Slika 4: Graf glede na spremenljivki z_1 in z_2 z označenima poglavitnima komponentama v prostoru poglavitnih komponent. Vir: [12]

v začetnih komponentah nahaja čim več, v kasnejših komponentah pa vedno manj informacij. Med p izpeljanimi spremenljivkami je cilj izbrati nekaj prvih ($\ll p$), ki vsebujejo največ informacij in s tem ohraniti izraznost podatkov kljub zmanjšanju števila dimenzij [12].

Za enostaven, sicer nerealen, prikaz delovanja PCA privzamemo, da je $p = 2$. Na ta način lahko prikažemo podatke v dveh dimenzijah. Slika 3 prikazuje primer grafa za dve spremenljivki x_1 in x_2, ki sta v korelaciji. Spremenljivost je prisotna tako v smeri x_1 kot tudi v smeri x_2, če pa vse skupaj pretvorimo v domeno poglavitnih komponent, dobimo spremenljivki z_1 in z_2 ter s tem graf, ki ga prikazuje slika 4 [12]. Iz slike 4 je vidno, da so spremembe v smeri z_1 velike, medtem ko je variacija v smeri z_2 majhna. Če so spremenljivke med seboj močno korelirane, bo prvih nekaj poglavitnih komponent odraz večine variacij originalnih spremenljivk, medtem ko bodo naslednje poglavitne komponente predstavljale smeri, kjer ni veliko raznolikosti. Z drugimi besedami, prve poglavitne komponente vsebujejo večino informacij [12].

Primeri apliciranja PCA na razna področja so:

- identifikacija pomembnih virov raznolikosti pri anatomskih meritvah različnih življenjskih vrst;
- analiza demografskih informacij, pridobljenih s pomočjo anketiranja starejšega prebivalstva Združenega kraljestva;
- pregled prostorskih in časovnih sprememb v atmosferskih znanostih;
- določanje pomembnih povezav med lastnostmi kemijskih spojin;
- analiza borznih cen [12].

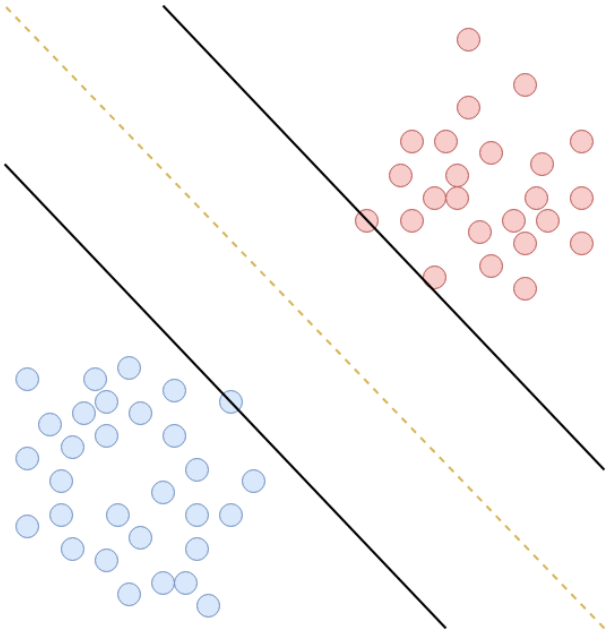
3.3 Uporabljeni klasifikatorji

Za klasificiranje besednih vrst smo uporabili tri različne klasifikatorje, in sicer One-vs-All *SVM* z linearnim jedrom, Naivni Bayes ter Random Forest. Posameznim parametrom pri vsakem izmed uporabljenih klasifikatorjev se nismo posebej posvečali, pri testiranju pa smo uporabili petkratno navzkrižno preverjanje. V tem podpoglavju bomo na kratko opisali vsakega izmed uporabljenih klasifikatorjev.

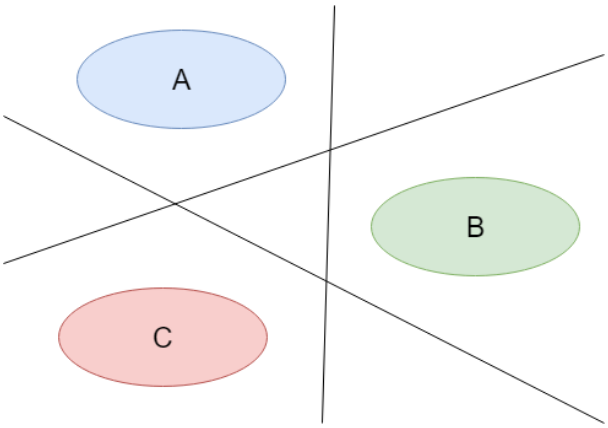
3.3.1 One-vs-All SVM

SVM je metoda strojnega učenja z uporabo nadzorovanega učenja, ki jo lahko uporabimo tako za klasifikacijo kot regresijo. Pri tej metodi vsak primerek iz učne množice predstavimo kot točko v n-dimenzionalnem prostoru, pri čemer n predstavlja število značilk primerka. Algoritem nato najde optimalno hiperravnino, ki najbolje razdeli dva razreda.

Naloga metode SVM je najti optimalno hiperravnino izmed vseh, ki ločujejo primerke na dva razreda, pri čemer primerki enega razreda ležijo pod, primerki drugega razreda pa nad hiperravnino. Primerki, ki ležijo nad hiperravnino, so pozitivni, primerki pod hiperravnino pa negativni. Najprej poiščemo v vsakem razredu primerek, ki leži najbližje delilni hiperravnini. Izbranim primerkom pravimo podporni vektorji (angl. support vectors). Skozi njiju potegnemo vzporednici, ki jima pravimo pozitivna in negativna ravnina. Razdaljo med tema dvema ravninama imenujemo rob (angl. margin). Optimalna delilna ravnina je tista, pri kateri je velikost roba maksimalna. Razred, kateremu pripada posamezen primerek, določimo glede na to, na kateri strani delilne ravnine se nahaja. Na sliki 5 lahko vidimo delilno, pozitivno ter negativno hiperravnino [13].



Slika 5: Prikaz hiperravnin pri metodi SVM.



Slika 6: Prikaz večrazredne klasifikacije SVM z metodo One-vs-All.

Če želimo SVM uporabiti za klasifikacijo v več razredov, moramo uporabiti eno izmed metod za večrazredno klasifikacijo. V naši implementaciji algoritma FLORS smo uporabili metodo One-vs-All. Metoda za vsak razred ustvari binarni klasifikator SVM. Vsak i-ti klasifikator razdeli primerke na tiste, ki spadajo v i-ti razred in tiste, ki ne. Primer večrazredne klasifikacije SVM z metodo One-vs-All je prikazan na sliki 6 [1].

3.3.2 *Naivni Bayes*

Klasifikator Naivni Bayes je del družine enostavnih verjetnostnih klasifikatorjev, ki temeljijo na uporabi Bayesovega pravila. Njegova uporaba je popularna predvsem na področju klasifikacije besedila ter zaznavanja neželjene pošte.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{1}$$

S pomočjo Bayesovega pravila, zapisanega v enačbi (1), lahko poiščemo verjetnost, da se zgodi dogodek A v primeru, da se je zgodil dogodek B.

Pri klasifikaciji s pomočjo Naivnega Bayesa se odločamo o tem, v kateri razred spada določen primerek glede na vrednosti njegovih značilk. Glavna predpostavka, na kateri temelji ta algoritem je, da so vse značilke posameznega primerka pogojno neodvisne druga od druge pri danem razredu. Prav tako predpostavimo, da ima vsaka značilka enak vpliv na odločitev klasifikatorja. Če dogodek A iz enačbe (1) sedaj zamenjamo s klasifikacijskim razredom y, ter dogodek B z vektorjem značilk X, potem lahko Bayesovo pravilo zapišemo z enačbo (2).

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)} \tag{2}$$

Ker je dogodek X presek več dogodkov oz. značilk, lahko zapišemo enačbo (3).

$$P(y|x_1, x_2, ..., x_n) = \frac{P(x_1|y)P(x_2|y)...P(x_n|y)P(y)}{P(x_1)P(x_2)...P(x_3)} \tag{3}$$

Vrednost izraza v imenovalcu je enaka za vse primerke v množici, zato se lahko imenovalca znebimo in uvedemo odvisnost. Tako dobimo enačbo (4).

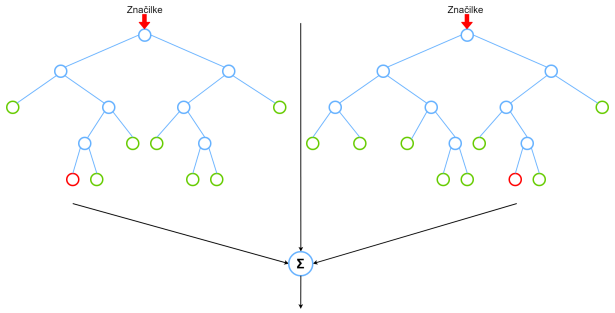
$$P(y|x_1, x_2, ..., x_n) \propto P(y)\prod_{i=1}^{n} P(x_i|y) \tag{4}$$

Primerek uvrstimo v razred y, dobljen z enačbo (5), ki vrne razred z najvišjo verjetnostjo [11, 3].

$$y = argmax_y P(y)\prod_{i=1}^{n} P(x_i|y) \tag{5}$$

3.3.3 *Random Forest*

Klasifikator Random Forest je metoda nadzorovanega strojnega učenja. Spada v t.i. ansambelske metode, katerih značilnost je, da združujejo več klasifikatorjev enakih ali različnih vrst v en skupen klasifikator. Ideja ansambelskih metod je, da več slabših klasifikatorjev skupaj deluje bolje kot en dober klasifikator. Klasifikator Random Forest je sestavljen iz več različnih odločitvenih dreves, ki jih naučimo na naključni podmnožici primerkov iz učne množice z naključno izbrano podmnožico značilk. Končno klasifikacijo dobimo tako, da združimo odločitve posameznih odločitvenih dreves. Generalna shema algoritma je prikazana na sliki 7 [9, 15].



Slika 7: Shema algoritma Random Forest.

4. REZULTATI

Algoritem smo preizkusili nad korpusom jos100k[1]. Z izvornim algoritmom FLORS smo nad slovenskim besedilom dosegli uspešnost 83,05 % po metriki F1. Po odstranitvi morfoloških značilk, vezanih na angleški jezik, ter izvajanju analize poglavitnih komponent smo dosegli 85,16 %. Algoritem PCA je vedno izboljšal rezultate, saj z zmanjšanjem dimenzije iskalnega prostora dosežemo boljšo generalizacijo in s tem višjo točnost klasifikacije prej nevidenih podatkov. Najboljše rezultate smo uspeli doseči z algoritmom SVM, katerega rezultati pri različnem naboru značilk so vidni na sliki 8. Z dodajanjem značilk, vezanih na pogoste predpone in končnice v slovenskem jeziku, nam uspešnosti klasifikacije ni uspelo izboljšati.

Klasifikacijo besednih vrst smo izvedli s tremi različnimi klasifikacijskimi algoritmi. Prej omenjeno točnost smo dosegli z metodo podpornih vektorjev, ki ga uporablja tudi izvorni algoritem FLORS. Modela Naivni Bayes in Random Forest sta bila manj uspešna kot SVM. Na slikah 9 in 10 vidimo klasifikacijske rezultate modelov nad testno množico podatkov.

Rezultate najuspešnejšega modela nad individualnimi besednimi vrstami si ogledamo z matriko napak na sliki 11. Napake so pogoste pri klasifikaciji razreda ”medmet”in ”neuvrščeno”. Napačno uvrstitev medmetov obrazložimo z dejstvom, da medmeti ne upoštevajo slovničnih pravil in se lahko pojavijo v mnogih oblikah, na primer z različnim številom zaporednih pojavitev iste črke. Poleg tega je število medmetov v učnem korpusu majhno. Razred ”neuvrščeno”je slabo definiran, saj vsaka beseda pripada določeni besedni vrsti. Razred ”neuvrščeno”predstavlja zgolj manjkajoče podatke v učni množici. Model bo takšno besedo uvrstil v drug razred, ki je z veliko verjetnostjo celo pravilen.

5. ZAKLJUČEK

V sklopu dela smo implementirali algoritem besedovrstnega označevanja FLORS [16] in na različne načine skušali izboljšati njegovo delovanje na slovenskem jeziku. FLORS definira nabor značilk, ki se uporabijo za učenje modela linearni SVM za označevanje angleškega jezika. Odstranili smo značilke, vezane izključno na angleščino, s čimer smo dvignili uspešnost označevanja. Poskusili smo dodati nove značilke na osnovi morfoloških značilnosti slovenščine, vendar smo opazili, da je to znižalo točnost rezultatov. Preizkusili smo učna modela Naivni Bayes in *Random Forest*, vendar sta se izkazala kot manj uspešna v primerjavi s SVM. Nazadnje smo uspeli uspešnost še dodatno povečati z uporabo algoritma za analizo poglavitnih komponent (PCA). Najvišja uspešnost, ki smo jo s tem dosegli, je 85,16 % po metriki F1.

Na podlagi rezultatov sklepamo, da uspešnost algoritma FLORS na slovenščini ni primerljiva z uspešnostjo drugih metod besednovrstnega označevanja [2]. Algoritem FLORS je preprost in hiter, a s preprostimi značilkami ne uspe zajeti kompleksnosti slovenskega jezika. Predpona in končnica ne nudita koristnih informacij o besedni vrsti, ker ima beseda preveč oblik, v katerih se lahko pojavi. Dodatne značilke, ki ne pripomorejo k razlikovanju med besednimi vrstami, povzročijo, da se dimenzija iskalnega prostora poveča in uspešnost klasifikacijskega modela zmanjša.

Točnost lahko povečamo z izključitvijo tistih obstoječih značilk, ki so nekoristne za označevanje besednih vrst slovenskega jezika. Odstranitev morfoloških značilk angleškega jezika je prvi korak k poenostavitvi algoritma in izboljšanju klasifikacijskih rezultatov. Transformacija PCA dodatno zmanjša dimenzijo prostora značilk, kar pri vsakem naboru značilk poveča točnost klasifikatorja linearni SVM.

V nadaljnjem delu se bomo osredotočili na slovnične operacije, s katerimi bi lažje obvladovali kompleksnost slovenskega jezika in izpeljali značilke, ki bi povečale točnost besednovrstnega označevanja. Smiselno bi bilo uporabiti tudi metode s področja lematizacije, saj bi z osnovnimi oblikami besed lažje določili njihove besedne vrste.

[1]Dostopno na http://nl.ijs.si/jos

6. VIRI

[1] B. Aisen. A comparison of multiclass svm methods, 2006.

[2] P. Belej, M. Robnik-v Sikonja, and S. Krek. Character-level part-of-speech tagger of slovene language, 2019. Slovenian language resource repository CLARIN.SI.

[3] M. Bozhinova. *NAIVNI BAYESOV KLASIFIKATOR*. PhD thesis, Univerza v Mariboru, Fakulteta za elektrotehniko, računalništvo in informatiko, 2015.

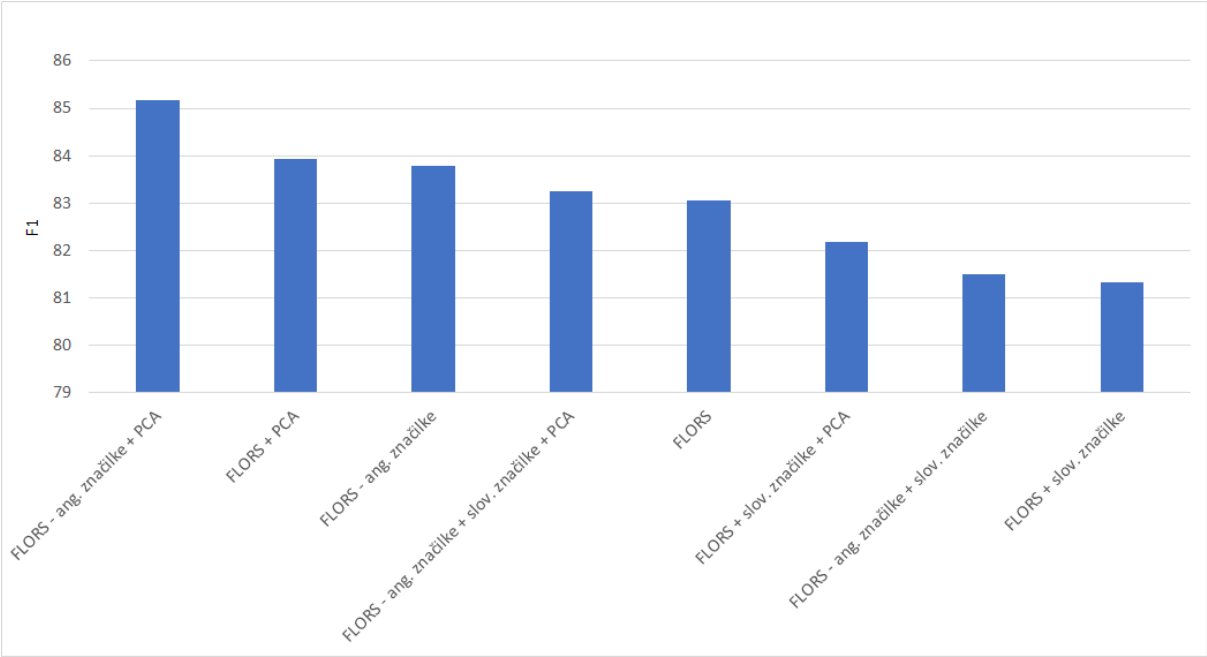
[4] E. Brill. A simple rule-based part of speech tagger. In *Proceedings of the Third Conference on Applied Natural Language Processing*, ANLC ’92, pages 152–155, Stroudsburg, PA, USA, 1992. Association for Computational Linguistics.

[5] E. Brill. Some advances in transformation-based part of speech tagging. *CoRR*, abs/cmp-lg/9406010, 1994.

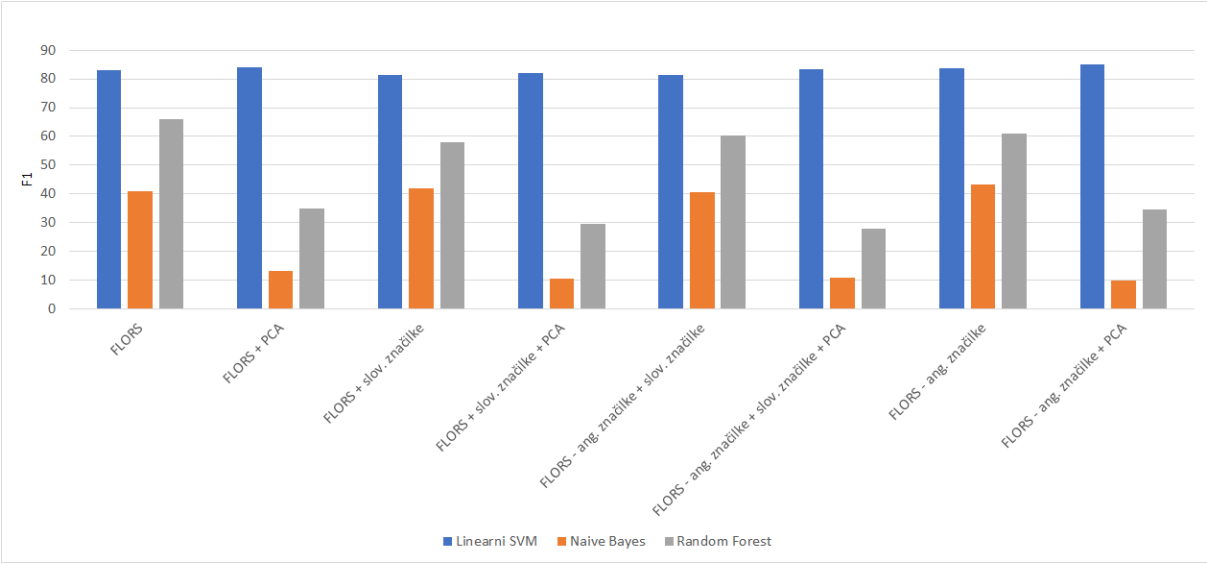
[6] J. Choi and M. Palmer. Fast and robust part-of-speech tagging using dynamic model selection. volume 2, pages 363–367, 07 2012.

[7] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *Computing Research Repository - CORR*, 12, 03 2011.

[8] D. Cutting, J. Kupiec, J. Pedersen, and P. Sibun. A practical part-of-speech tagger. In *Proceedings of the Third Conference on Applied Natural Language Processing*, ANLC ’92, pages 133–140, Stroudsburg, PA, USA, 1992. Association for Computational



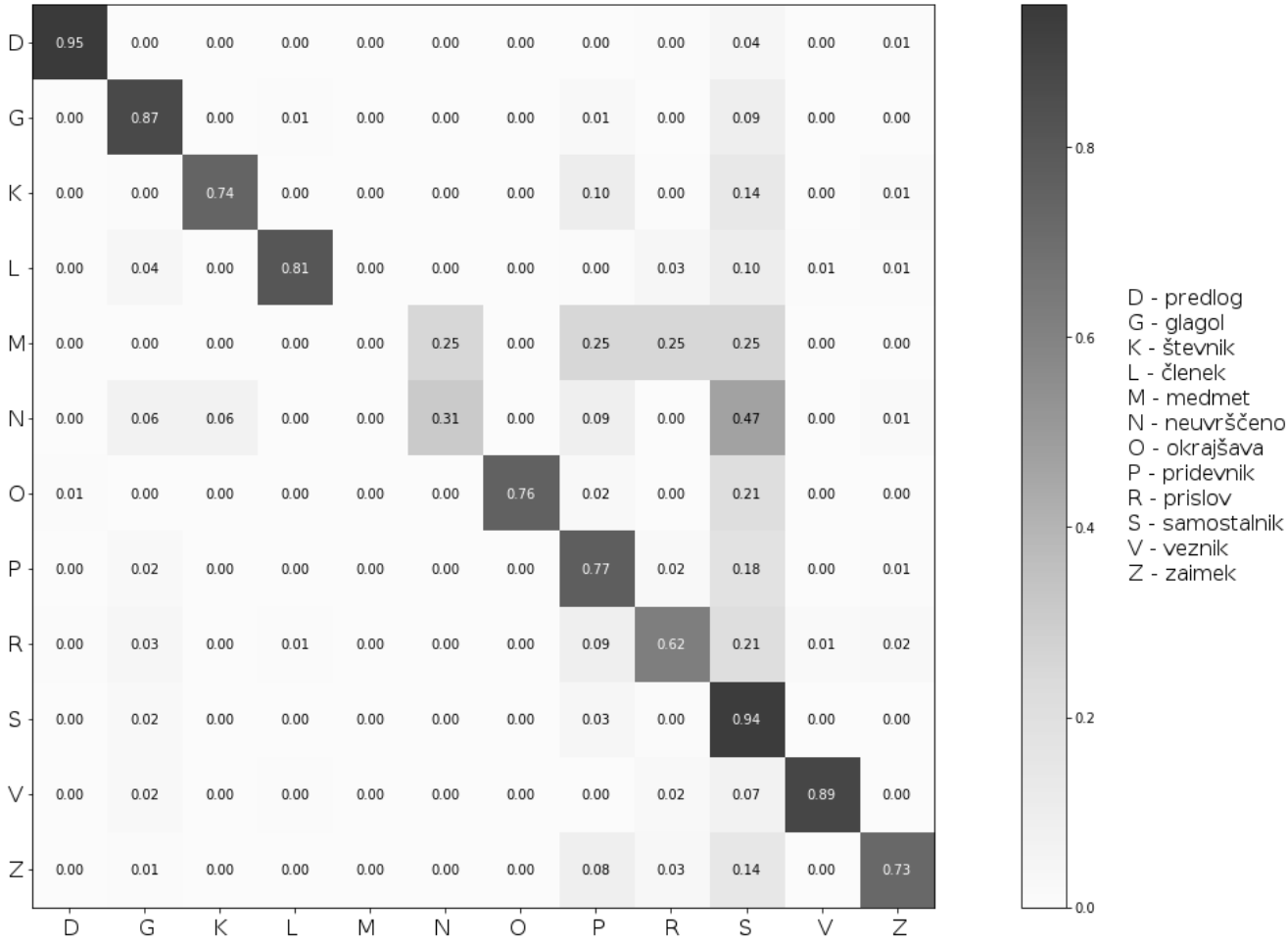
Slika 8: Urejeni rezultati metrike F1 pri klasifikacijskem modelu SVM za dodane slovenske morfološke značilke, odstranjene angleške morfološke značilke in apliciranje transformacije PCA.



Slika 9: Rezultati metrike F1 za modele SVM, Naivni Bayes in Random Forest.

Slika 10: Vrednosti metrike F1 klasifikacijskih modelov SVM, učenih z različnimi nabori značilk: dodane slovenske morfološke značilke, odstranjene angleške morfološke značilke in apliciranje transformacije PCA.

	Linearni SVM	Naivni Bayes	Random Forest
FLORS	83.05 %	40.81 %	65.83 %
FLORS + PCA	83.94 %	13.03 %	34.77 %
FLORS + slov. značilke	81.32 %	42.03 %	57.96 %
FLORS + slov. značilke + PCA	82.19 %	10.49 %	29.64 %
FLORS - ang. značilke + slov. značilke	81.51 %	40.63 %	60.22 %
FLORS - ang. značilke + slov. značilke + PCA	83.24 %	10.83 %	27.81 %
FLORS - ang. značilke	83.79 %	43.16 %	61.05 %
FLORS - ang. značilke + PCA	85.16 %	9.65 %	34.59 %



Slika 11: Matrika zmot algoritma FLORS po metriki F1 z modelom SVM z odstranjenimi angleškimi morfološkimi značilkami in po uporabi PCA. Prikazana je točnost klasifikacije posamezne besedne vrste. Stolpec predstavlja razred, v katerega smo besedo uvrstili. Vrstica predstavlja dejanski razred besede.

Linguistics.
[9] N. Donges. The random forest algorith, 2018.
[10] P. F. Brown, V. Dellapietra, P. V. de Souza, J. Lai, and R. Mercer. Class-based n-gram models of natural language. *Computational Linguistics*, 18:467–479, 01 1992.
[11] R. Gandhi. Naive bayes classifier, 2018.
[12] I. Jolliffe. *Principal Component Analysis*, pages 1094–1096. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.
[13] D. Kumar. Demystifying support vector machines, 2019.
[14] J. Miller, M. Torii, and K. Vijay-Shanker. Building domain-specific taggers without annotated (domain) data. pages 1103–1111, 01 2007.
[15] S. Patel. Chapter 5: Random forest classifier, 2017.
[16] T. Schnabel and H. Schütze. Flors: Fast and simple domain adaptation for part-of-speech tagging. *Transactions of the Association for Computational Linguistics*, 2:15–26, 12 2014.
[17] H. Schütze. Part-of-speech induction from scratch. pages 251–258, 01 1993.
[18] H. Schütze. Distributional part-of-speech tagging. page 141, 03 1995.
[19] K. Toutanova, D. Klein, C. Manning, and Y. Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology—NAACL '03*, 1, 03 2004.
[20] J. Us Gim Enez and L. S M Arquez. Svmtool: A general pos tagger generator based on support vector machines. 07 2004.

Analiza igralnih strategij v iterativni zaporniški dilemi

Klemen Kac
Laboratorij za heterogene računalniške sisteme
Koroška cesta 46
Maribor, Slovenija
klemen.kac@um.si

Bor Praznik
Laboratorij za heterogene računalniške sisteme
Koroška cesta 46
Maribor, Slovenija
bor.praznik@um.si

POVZETEK

V pričujoči raziskavi smo analizirali igralne strategije v iterativni zaporniški dilemi. Strateške igre so sestavljene iz množice igralcev, množice strategij za vsakega igralca in vektorja izplačil, ki vsaki strategiji priredi določeno vrednost izplačila. Cilj strateške igre je maksimizirati vrednost izplačila za posameznega igralca. Igralne strategije v članku smo med seboj primerjali v dveh vrstah turnirjev, kjer se več posameznikov pomeri v več tekmah. Pri prvi vrsti turnirja, se vsak igralec pomeri z vsemi ostalimi, pri drugi pa smo uporabili turnirsko selekcijo, kjer po vsakem krogu izpade najslabši igralec. Implementirali smo tudi lastno igralno strategijo, ki dosega rezultate primerljive z obstoječimi strategijami. Ugotovili smo, da je učinkovitost posamezne igralne strategije zelo odvisna od vrste turnirja, s katerim strategijo ocenjujemo.

Ključne besede

zaporniška dilema, turnirska selekcija, igralne strategije

1. UVOD

Strateška igra je urejena trojka $G = \langle N, S, u \rangle$, kjer pomenijo:

- $N = \{1, ..., n\}$ je končna množica igralcev,
- $S = \{S_i\}$ je množica strategij S_i, za vsakega izmed igralcev $i = \{1, ..., n\}$,
- $u_i : S \rightarrow \mathbb{R}$ je funkcija koristnosti, ki vsaki množici strategij S priredi izplačilo za i-tega igralca $u_i(S)$.

Cilj teorije strateških iger je svetovati, katero strategijo bodo nasprotniki igrali z večjo verjetnostjo, oz. priporočiti igralcem, katero strategijo igrati, da je izplačilo največje.

V splošnem poznamo tri koncepte strateških iger: dominantnost, stabilnost in maksimalna družbena korist. Glede na določeno strategijo i-tega igralca obstaja več možnih izidov.

Pravimo, da je strategija S_i dominantna za igralca i, če ne glede na to, katero strategijo S_j igra igralec j, igralec i dobi več, kot bi dobil, če bi igral katerokoli drugo strategijo. Igranje dominantne strategije zagotavlja agentu najboljši izid. Najbolj važen koncept v teoriji iger je stabilnost. Dve strategiji S_1 in S_2 sta stabilni pod pogojem, da:

- če igralec i odigra strategijo S_1, igralec j ne more odigrati boljše poteze kot S_2 in
- če igralec j odigra strategijo S_2, igralec i ne more odigrati boljše poteze kot S_1.

Maksimalna družbena korist ne gleda na izplačilo posameznega agenta, ampak vseh agentov skupaj.

Analiza igralnih strategij v zaporniški dilemi je že precej dobro obdelano področje v teoriji iger. V članku [1] avtorji ugotovljajo, da dominantne strategije v posameznem krogu niso nujno dominantne v več krogih. Axelrod v članku [2] opisuje evolucijsko razvijanje igralnih strategij v iterativni zaporniški dilemi. Isti avtor v članku [3] predlaga tudi, kako učinkovito igrati iterativno zaporniško dilemo.

V naši raziskavi smo analizirali že omenjene igralne strategije v iterativni zaporniški dilemi in poskušali izmeriti uspešnost posameznih strategij v določeni vrsti turnirja. Definirali smo dve vrsti turnirja. Pri prvi vrsti turnirja vsak posameznik tekmuje z vsemi ostalimi. Zmagovalna strategija je tista, ki ima skupno najboljši rezultat. Pri drugi vrsti po vsakem krogu izločimo najslabšo strategijo, ostale strategije pa igrajo še enkrat v naslednjem krogu. V takšnem turnirju je zmagovalna strategija tista, ki ne izgubi z nobeno izmed preostalih strategij v turnirju.

Struktura članka v nadaljevanju je naslednja: v drugem poglavju podrobneje predstavimo zaporniško dilemo, v tretjem povzamemo igralne strategije in opišemo obe vrsti turnirja, s katerima smo analizirali uspešnosti strategij ter podrobneje opišemo lastno igralno strategijo "PRAKAC", četrto poglavje je namenjeno predstavitvi poskusov in rezultatov, v zadnjem poglavju pa povzamemo opravljeno delo ter predstavimo naše ugotovitve.

2. ZAPORNIŠKA DILEMA

Zaporniška dilema je igra, v kateri nastopata dva igralca, ki se pretvarjata, da sta zapornika. Zaprta sta vsak v svoji

DOI: https://doi.org/10.26493/978-961-7055-82-5.101-106

ločeni celici. Med seboj se ne smeta pogovarjati. Policija želi vsaj enega izmed njiju zapreti za dalj časa [4]. Zapornik lahko svoj zločin prizna ali ne. Odločitev posameznega zapornika se ovrednoti s številom let, ki jih mora preživeti v zaporu. Cilj obeh zapornikov je, da sta zaprta čim manj časa.

Analiza izidov klasične zaporniške dileme pokaže, da je za zapornika najbolje, če izda sojetnika, saj v primeru, da drugi zapornik ne prizna sodelovanja v zločinu, prvi "namoči" drugega, ki mora odgovarjati za svoje nesodelovanje s policijo, in sicer z maksimalno kaznijo, prvi pa bo oproščen vsakršne krivde. Ta igra v večagentnih sistemih predstavlja dilemo, saj je v nasprotju s tezo, da je sodelovanje med agenti tista strategija, ki zagotavlja ključ do uspeha.

Vendar pa se izkaže, da sodelovanje postane zmagovalna strategija, če igro ponavljamo večkrat. V tem primeru imamo opravka s t.i. iterativno zaporniško dilemo, kjer se zapornika lahko odločata za igranje poljubne strategije. Lahko se na primer odločita, da bosta vedno izdajala svojega nasprotnika ali pa, da bosta vedno sodelovala. Število let zapora se pri iterativni zaporniški dilemi skozi kroge seštevа. S ponavljanjem igre dosežemo, da postane sodelovanje racionalna odločitev. Z igranjem strategije sodelovanja vzpodbujamo, da tudi nasprotnik igra isto, kar postane na dolgi rok najbolj profitabilna odločitev za oba igralca.

Igralca (A in B), morata spoštovati naslednja pravila [5]:

- Če igralec A izda igralca B, ta pa njega ne, potem gre igralec B v zapor za 3 leta, igralec A pa je izpuščen.
- Če zločin priznata oba, gresta oba v zapor za 2 leti.
- Če zločina ne prizna nobeden, bosta oba obsojena na 1 leto zapora.

V Tabeli 1 vidimo prikaz razdelitve točk med igralcema A in B v primeru sodelovanja oziroma izdaje. Opazimo, da je za oba igralca dominantna strategija izdajanje (nesodelovanje), kajti ne glede na nasprotnikovo odločitev, igralec največ pridobi z izdajanjem. Stabilnost dosežemo, če oba igralca hkrati izdajata drug drugega. Takrat sta odločitvi druga drugi najboljši odgovor. Maksimalna družbena korist je v primeru, ko oba igralca sodelujeta. Takrat je vsota let, ki jih oba igralca morata preživeti v zaporu najmanjša.

Tabela 1: Točkovanje

A \ B	Sodelovanje	Izdaja
Sodelovanje	A in B dobita po 1 leto zapora (-1 tč.)	A: 3 leta (-3 tč.) B: izpuščen (0 tč.)
Izdaja	A: izpuščen (0 tč.) B: 3 leta (-3 tč.)	A in B dobita po 2 leti zapora (-2tč.)

Vsak igralec se mora odločiti, katero igralno strategijo bo igral. Igralec A lahko sodeluje in pri tem upa, da bo sodeloval tudi njegov nasprotnik. S tem bi oba igralca dobila po 1 leto zapora. V primeru, da igralec A izda igralca B, je igralec A izpuščen, igralec B pa dobi 3 leta zapora. Če se katerikoli izmed igralcev odloči za izdajanje, lahko v najslabšem primeru dobi 2 leti zapora. Seveda pa igralec, ki izdaja upa, da bo nasprotnik sodeloval. Tako bi se sam izognil kazni, nasprotnik pa bi dobil 3 leta zapora. S ponavljanjem igre želimo dokazati, da kljub temu, da je za oba igralca dominantna strategija izdajanje, se na dolgi rok splača sodelovanje.

3. IGRALNO OKOLJE

Igro iterativna zaporniška dilema smo implementirali z uporabo označevalnega jezika HTML in programskega jezika JavaScript kot spletno aplikacijo[1]. Grafični prikaz igre lahko vidimo na Sliki 2. Možne igralne strategije so opisane v naslednjem podpoglavju. Znotraj aplikacije lahko določimo: (1) koliko igralcev bo igralo na turnirju, (2) koliko iteracij bodo igrali in (3) katero vrsto turnirja bodo uporabljali. Končne rezultate nato prikažemo v tabeli, ki je sortirana po najnižji količini kazni, ki jo je posameznik z določeno strategijo pridobil in tako razglasimo zmagovalca. Zmagovalna strategija je tista, ki je kaznovana z najmanjšim številom let zapora.

3.1 Opis igralnih strategij

Strategije, ki smo jih izbrali, so ene izmed najbolj znanih v zaporniški dilemi. Nekatere so enostavnejše, druge pa bolj sofisticirane. V naši študiji smo uporabili naslednje:

- "ALL-D" - v tej strategiji igralec nikoli ne sodeluje, temveč vedno izdaja. Igralec torej v najboljšem primeru ne gre v zapor, v najslabšem pa gre v zapor za dve leti [6].
- "RANDOM" - igralec se naključno odloči ali bo sodeloval ali izdajal svojega soigralca [7].
- "TIT-FOR-TAT" - v prvem krogu igralec sodeluje, v vseh ostalih pa ponovi zadnjo soigralčevo odločitev [8].
- "TESTER" - strategija izkorišča sisteme, ki neprestano sodelujejo; v prvem krogu igralec izdaja, nato pa ponavlja zaporedje "sodelujem, sodelujem, izdajam". Če tudi soigralec medtem kdaj ne sodeluje, nadaljujemo z igranjem strategije "TIT-FOR-TAT" [9].
- "JOSS" - strategija je namenjena izkoriščanju "šibkih" nasprotnikov; zelo podobna je strategiji "TIT-FOR-TAT", razlika je le v tem, da v 10% primerov namesto sodelovanja izbere izdajanje [10].

Poleg že znanih strategij smo razvili tudi lastno strategijo "PRAKAC", ki jo predstavljamo v nadaljevanju poglavja.

3.2 Predlagana strategija "PRAKAC"

Na podlagi delovanja prej opisanih strategij smo implementirali lastno strategijo in z njo poskušali dobiti čim boljše rezultate. Naša strategija se imenuje "PRAKAC" in temelji na maščevanju. Pri tej strategiji v prvem krogu izdamo nasprotnika. S to potezo upamo, da strategija pridobi proti

[1] http://betrayal-simulator.praznikbor.now.sh/

Slika 1: Nastavitve spletne aplikacije

strategijam, ki v prvem krogu sodelujejo. V nadaljnih potezah spremljamo zadnjo nasprotnikovo odločitev. V primeru, da v zadnji potezi nasprotnik sodeluje, v naslednji potezi sodelujemo tudi mi. To storimo z upanjem, da bo tudi v naslednji potezi nasprotnik sodeloval, ker je to za oba najboljša poteza. Če nas nasprotnik kadarkoli izda, ga mi v naslednjih dveh potezah izdamo in tako upamo, da pridobimo nazaj izgubljene točke. Strategija je najbolj uspešna proti tistim strategijam, ki izmenično sodelujejo in izdajajo, ima pa tudi dobre rezultate proti bolj sofisticiranim strategijam.

Algoritem 1 Strategija PRAKAC

1: $N \Leftarrow$ current turn
2: $X \Leftarrow cooperate$
3: **if** $N = 1$ or $(N - 1$ or $N - 2) = betrayed$ **then**
4: $\quad X \Leftarrow betray$
5: **end if**
6: return X

V Algoritmu 1 vidimo postopek izbire odločitve pri strategiji "PRAKAC". Najprej inicializiramo spremenljiko N, v katero shranjujemo nasprotnikove odločitve. Vrednost spremenljivke X hrani našo trenutno odločitev. Privzeto jo nastavimo na sodelovanje. Potem s pogojnim stavkom pogledamo, če je bila nasprotnikova zadnja ali predzadnja odločitev izdajanje. Če pogojni stavek drži, nastavimo vrednost spremenljivke X na izdajanje. Kot rezultat algoritma vrnemo vrednost spremenljivke X.

3.3 Turnir s krožnim dodeljevanjem

Pri turnirju s krožnim dodeljevanjem vsak posameznik tekmuje z vsemi ostalimi [11]. V našem primeru smo to vrsto turnirja uporabili za testiranje učinkovitosti posameznih strategij. V turnir smo vključili poljubno število igralcev, ki igrajo eno od opisanih strategij. Igralec je nato tekmoval z ostalimi in glede na uspešnost pridobival leta, ki jih mora preživeti v zaporu. Na koncu smo pridobili rezultat v obliki števila let, ki jih dobi igralec, če uporablja določeno strategijo. Kot najbolj uspešna strategija se smatra tista, ki igralcu pridobi najnižjo število let v zaporu [12].

V to vrsto turnirja smo dodali še dodatno pravilo izločanja, ki na koncu vsakega turnirja izloči najslabšega igralca (tistega, ki bi moral v zaporu preživeti največje število let) in ponovili celotni turnir s preostalimi igralci. Kadar izločimo strategije, ki jih je preprosto izkoriščati, lahko pri tem ugotovimo, katere strategije so v povprečju najboljše.

4. POSKUSI IN REZULTATI

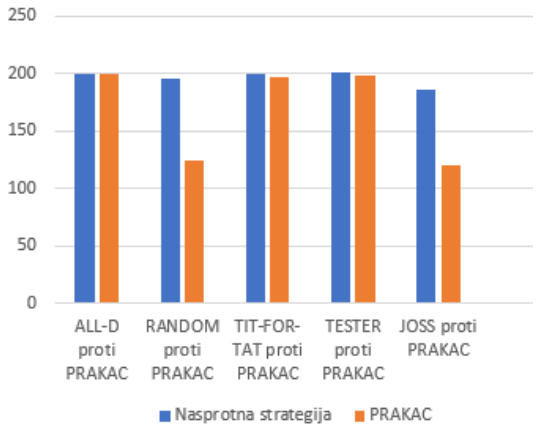
Strategije smo med sabo primerjali z uporabo implementirane spletne aplikacije, na kateri lahko nastavimo število iteracij, število tekmovalcev, ki predstavljajo določeno strategijo ter način izvajanja turnirja (z izločanjem ali brez). Za primerjavo strategij smo v obeh vrstah turnirja izpisovali rezultate za 10, 100, 500 in 1000 iteracij. Tako lahko ugotovimo, katere strategije so boljše na krajši in katere na daljši igralni rok. Število igralcev, ki predstavljajo določeno strategijo smo nastavili najprej na 1, nato na 5 in na koncu še na 10. Če pri turnirju z izločanjem vsako strategijo predstavlja samo 1 igralec, se lahko zgodi, da bo strategija, ki v povprečju dosega zelo dobre rezultate, že v začetku turnirja igrala z močnejšo strategijo in bo izpadla zelo hitro. Z večjim številom igralcev na vsako strategijo ta problem omilimo. Za verodostojne rezultate smo vsem strategijam vedno nastavili enako število igralcev.

Za demonstracijo smo našo strategijo primerjali z ostalimi implementiranimi strategijami v igri s stotimi iteracijami. V Tabeli 2 vidimo rezultate lastne strategije v igri ena proti ena, z že obstoječimi igralnimi strategijami. Za dvoboj smo dodelili 100 iteracij, z namenom da imajo strategije ki temeljijo na odzivanju možnost blesteti. Opazimo, da strategija "PRAKAC" premaga vse strategije in pridobi najmanjšo število let v zaporu, z izjemo "ALL-D", s katero odigra izenačen rezultat.

Na Sliki 2 vidimo grafično predstavitev rezultatov lastne strategije v igri ena na ena proti ostalim strategijam. Opazimo, da je rezultat v primerjavi s strategijami "ALL-D", "TIT-FOR-TAT" in "TESTER" precej izenačen, medtem ko

Tabela 2: Rezultati strategije PRAKAC 1 proti 1 s 100 iteracijami.

	PRAKAC
ALL-D	200 / 200
RANDOM	124 / 196
TIT-FOR-TAT	197 / 200
TESTER	198 / 201
JOSS	120 / 186



Slika 2: Grafična predstavitev strategije PRAKAC v primerjavi z ostalimi strategijami.

v je rezultat v primerjavi s strategijami "RANDOM" in "JOSS" občutno boljši v prid strategije "PRAKAC".

4.1 Turnir s krožnim dodeljevanjem brez izločanja

Najprej smo v turnir vključili po enega igralca za vsako strategijo, nato po pet igralcev za vsako strategijo in na koncu še po deset igralcev za vsako strategijo (Tabela 3). Pri turnirju s krožnim dodeljevanjem brez izločanja, se pri enemu igralcu na vsako strategijo izkaže, da prevladuje strategija "ALL-D", ki je odlična za izkoriščanje naivnih strategij, katerih je pri naši postavitvi vedno dovolj, da strategija nadvladuje.

Na drugem mestu se vedno pojavi naša strategija "PRAKAC", ki je pravtako dobra v izkoriščanju naivnih strategij. Tretje mesto se pri enemu igralcu na strategijo spreminja. Pri desetih iteracijah vidimo na tretjem mestu celo strategijo "RANDOM", ki s svojim kaotičnim pristopom pretenta bolj sofisticirane metode, preden jo imajo le-te možnost premagati. Pri stotih iteracijah se na tretjem mestu pojavi strategija "TESTER", kadar pa povečamo iteracije na petsto in več, pa tretje mesto okupira strategija "TIT-FOR-TAT".

Ko imamo pri vsaki strategiji več igralcev, ki zastopajo določeno strategijo, začnejo prevladovati igralci strategije "ALL-D". Slednja strategija se je izkazala za najboljšo v tej vrsti turnirja.

4.2 Turnir s krožnim dodeljevanjem in izločanjem

Enak turnir ponovimo še enkrat, le da tokrat v vsakem krogu izločimo strategijo, ki ima najslabši rezultat. Turnir izvajamo tako dolgo, dokler ne dobimo skupnega zmagovalca. Pri turnirju, kjer imamo samo enega predstavnika vsake strategije, dobimo več zmagovalcev. Pri desetih iteracijah sta zmagovalca strategiji "ALL-D" in "PRAKAC", pri večjem številu iteracij pa strategiji "TIT-FOR-TAT" in "TESTER" (Tabela 4). Ugotovimo, da pri enem igralcu na strategijo prevladuje "ALL-D", pri večih igralcih na strategijo pa "TIT-FOR-TAT". Iz tega lahko razberemo, da je strategija "ALL-D" odlična za izkoriščanje simplističnih strategij, ki ne poskušajo predvidevati nasprotnikove poteze (npr. "JOSS" in "RANDOM"). Ko te strategije izpadejo iz turnirja, pa začnejo prevladovati bolj sofisticirane strategije, kot je "TIT-FOR-TAT".

Pod drobnogled smo vzeli variantni turnir s krožnim dodeljevanjem z enim igralcem na strategijo. V Tabeli 5 lahko vidimo, kdaj je katera strategija izpadla pri določenem številu iteracij. Opazimo, da je, glede na povprečno mesto izpada, najboljša strategija "TIT-FOR-TAT". Precej podobne rezultete je pridobila strategija "TESTER", vendar ima slednja slabše povprečje, zaradi rezultata pri igri z desetimi iteracijami (izpade kot druga strategija). Naša strategija "PRAKAC" v tej vrsti tekmovanja zasede sprejemljivo tretje mesto. Najslabše se izkaže strategija "JOSS". Presenetljivo jo premaga tudi naivna strategija "RANDOM", ki jo premaga s pomočjo sreče, kajti "RANDOM" naključno izbira svoje odločitve. Zato lahko v določenih primerih doseže zelo dobre rezultate, v drugih primerih pa porazne.

4.3 Diskusija

Z izvajanjem raznih turnirjev smo ugotovili, da s prirejenim turnirjem s krožnim dodeljevanjem pridobimo zanimivejše rezultate. Medtem ko v turnirju brez izločanja vedno prevladuje strategija, ki izkorišča naivne metode, v turnirju z izločanjem prevladajo bolj sofisticirane strategije (Tabela 3).

V Tabeli 5 lahko opazimo, da naivne strategije vedno izpadejo v prvih krogih, nato izpadejo strategije, ki so preživele le z izkoriščanjem le-teh, na koncu pa imamo samo "pametne" strategije. Strategije, ki prevladujejo v turnirju z izločanjem, so takšne, ki temeljijo na sodelovanju, ampak ne brezpogojnem.

5. ZAKLJUČEK

V raziskavi smo analizirali pet obstoječih igralnih strategij v zaporniški dilemi. Razvili smo tudi lastno igralno strategijo in jo primerjali z ostalimi. Strategije smo med seboj primerjali v turnirju s krožnim dodeljevanjem brez izločanja in turnirju z izločanjem.

Ugotovili smo, da so rezultati strategij zelo odvisni od vrste turnirja, ki ga uporabimo. Določena strategija lahko doseže v nekem turnirju zelo dober rezultat, v drugem pa pogori. Zanimivo je dejstvo, da naša strategija "PRAKAC"

Tabela 3: Turnir s krožnim dodeljevanjem brez izločanja.

	1 igralec na strategijo	5 igralcev na strategijo	10 igralcev na strategijo
10 iteracij	1. ALL-D 2. PRAKAC 3. RANDOM	1. ALL-D 2. ALL-D 3. ALL-D	1. ALL-D 2. ALL-D 3. ALL-D
100 iteracij	1. ALL-D 2. PRAKAC 3. TESTER	1. ALL-D 2. ALL-D 3. ALL-D	1. ALL-D 2. ALL-D 3. ALL-D
500 iteracij	1. ALL-D 2. PRAKAC 3. TIT-FOR-TAT	1. ALL-D 2. ALL-D 3. ALL-D	1. ALL-D 2. ALL-D 3. ALL-D
1000 iteracij	1. ALL-D 2. PRAKAC 3. TIT-FOR-TAT	1. ALL-D 2. ALL-D 3. ALL-D	1. ALL-D 2. ALL-D 3. ALL-D

Tabela 4: Turnir s krožnim dodeljevanjem in izločanjem.

	1 igralec na strategijo	5 igralcev na strategijo	10 igralcev na strategijo
10 iteracij	ALL-D, PRAKAC	TIT-FOR-TAT	TIT-FOR-TAT
100 iteracij	TIT-FOR-TAT, TESTER	TIT-FOR-TAT	TIT-FOR-TAT
500 iteracij	TIT-FOR-TAT, TESTER	TIT-FOR-TAT	TIT-FOR-TAT
1000 iteracij	TIT-FOR-TAT, TESTER	TIT-FOR-TAT	TIT-FOR-TAT

Tabela 5: Turnir s krožnim dodeljevanjem in izločanjem, 1 igralec na strategijo, skupno mesto.

	10 iteracij	100 iteracij	500 iteracij	1000 iteracij	POVPREČJE
TIT-FOR-TAT	3.	1.	1.	1.	1,5
TESTER	5.	1.	1.	1.	2
PRAKAC	1.	3.	3.	3.	2,5
ALL-D	1.	4.	4.	4.	3,25
RANDOM	4.	6.	5.	5.	5
JOSS	6.	5.	6.	6.	5,75

dosega najboljši rezultat v igri ena na ena, na implementiranih turnirjih pa dosega povprečne rezultate. Vidimo tudi, da je naša strategija najbolj efektivna v manjšem številu soočanj, torej strategija v nasprotju z strategijo "TIT-FOR-TAT" gleda na kratkoročni dobiček.

Rezultati so odvisni tudi od točkovanja, ki ga uporabimo med tekmovanjem. Med raziskovanjem tematike smo opazili, da raziskovalci igre ne uporabljajo fiksnega točkovanja, temveč si ga določajo po svoje. Pri turnirju s krožnim dodeljevanjem brez izločanja smo ugotovili, da je najboljša strategija "ALL-D". Pri turnirju z izločanjem je najboljša strategija "TIT-FOR-TAT", ki je med bolj sofisticiranimi strategijami in je sposobna izkoriščati ostale pametne strategije. Strategijo, ki bi dosegala solidne rezultate ne glede na vrsto turnirja, je zelo težko implementirati, saj v enem turnirju predvladujejo strategije, ki so namenjene izkoriščanju naivnih pristopov, v drugem turnirju pa prevladajo bolj sofisticirane strategije.

Čeprav obstaja že ogromno strategij, menimo, da je na voljo še veliko maneverskega prostora za izboljšave obstoječih strategij in implementacijo novih. V prihodnje bi lahko v spletno aplikacijo, s katero smo delali analizo, vključili še več igralnih strategij in več vrst turnirjev. S tem bi lahko naredili raziskavo še obsežnejšo.

6. LITERATURA

[1] J. Roberts R. Wilson D. M. Kreps, P. Milgrom. Rational cooperation in the finitely repeated prisoners' dilemma. In *Journal of Economic Theory*, pages 245–252, 1982.

[2] R. Axelrod. The evolution of strategies in the iterated prisoner's dilemma. In *The Dynamics Of Norms*, 1997.

[3] R. Axelrod. Effective choice in the prisoner's dilemma. In *Journal of Conflict Resolution*, 1980.

[4] C. Tőke G. Szabó. Evolutionary prisoner's dilemma game on a square lattice. In *American Physical Society*, pages 58–69, 1998.

[5] WD. Hamilton R. Axelrod. The evolution of cooperation. In *Science*, pages 58–69, 1998.

[6] J. R. Stevens D. W. Stephens, C. M. McLinn. Discounting and reciprocity in an iterated prisoner's dilemma. In *Science*, pages 2216–2218, 2002.

[7] R. Axelrod. More effective choice in the prisoner's dilemma. In *Sage journals*, 1980.

[8] K. Sigmund M. A. Novak. Tit for tat in heterogeneous populations. In *Nature*, 1992.

[9] P. Mathieu J. Delahaye. Complex strategies in the iterated prisoner's dilemma. In *Semantic Scholar*, 1994.

[10] R. Axelrod. Effective choice in the prisoner's dilemma. In *Sage journals*, 1980.

[11] L. Moser F. Harary. The theory of round robin tournaments. In *The American Mathematical Monthly*, 2018.

[12] X. Yao P. J. Darwen. On evolving robust strategies for iterated prisoner's dilemma. In *EvoWorkshops*, 1994.

Napovedovanje nogometnega zmagovalca z rekurentno nevronsko mrežo LSTM

Nejc Planer
Fakulteta za elektrotehniko, računalništvo in informatiko
Koroška cesta 46, 2000 Maribor
nejc.planer@student.um.si

Mladen Borovič
Fakulteta za elektrotehniko, računalništvo in informatiko
Koroška cesta 46, 2000 Maribor
mladen.borovic@um.si

POVZETEK

Prispevek predstavlja napovedovanje nogometnega zmagovalca in nekaterih drugih statistik z rekurentno nevronsko mrežo, ki uporablja celico LSTM. Opisano je tudi kje lahko pridobimo podatke za posamezne lige in tekmovanja, kako te podatke potem preoblikujemo in njihov končen izgled. Podamo tudi uporabljeno nevronsko mrežo, ter rezultate. To so ali pade več kot 1,5 gola na tekmo, ali pade več kot 2,5 gola na tekmo in ali obe ekipi zadeneta. Na koncu podamo še nekaj idej za možne izboljšave uspešnosti napovedovanja.

Kjučne besede

napovedovanje, rekurentne nevornske mreže, celica LSTM, nogomet, umetna inteligenca

1. UVOD

Stave so velik del športa in pritegnejo veliko ljudi, saj so privlačne zaradi možnega hitrega zaslužka. Problem je, da je šport lahko zelo nepredvidljiv in je težko ugotoviti končen izid. Zato si pri stavljanju velikokrat pomagamo s statistiko prejšnjih tekem, oziroma prejšnjih let. To naredi nevronske mreže zelo zanimive, saj lahko pošljemo to statistiko kot vhod v mrežo in potem ta napove rezultate za naprej.

Namen dela je ugotoviti, ali lahko z nogometno statistiko in nevronskimi mrežami uspešno napovemo, kaj se bo zgodilo v prihodnjih tekmah. Za to je potrebno pridobljene podatke ustrezno preoblikovati in dodati nove, da si izboljšamo natančnost. Tudi oblika mreže je pomembna, saj vse ne dosegajo iste natančnosti.

Avtor Korpič je v diplomskem delu [13] uporabil različne algoritme strojnega učenja za napovedovanje zmagovalca nogometne tekme. Uspešnost napovedovanja je bila med 30 in 50 odstotki.

Tudi v raziskovalnem delu [9] so uporabili različne algoritme strojnega učenja za napovedovanje nizozemske nogometne lige. Najboljše rezultati so bili v povprečju okoli 60 odstotkov.

Microsoft ima svoj sistem za napovedovanje športnih rezultatov. Leta 2014 so napovedali vse tekme izločevalnega dela svetovnega prvenstva pravilno [11]. Njihov matematični model vzame v račun več faktorjev, od števila zmag, porazov in neodločenih izidov vse do tipa podlage na igrišču in vremenskih razmer [10].

Za napovedovanje bomo uporabili rekurentne nevronske mreže, ki še na tem področju niso veliko uporabljene, in sicer da vidimo ali lahko z njimi dosežemo boljše rezultate kot z ostalimi metodami umetne inteligence.

V naslednjem poglavju opišemo pridobivanje podatkov, ki jih potem tudi podrobneje razložimo in povemo njihovo strukturo. Sledi njihova preobdelava preden jih pošljemo v nevronsko mrežo. Zatem v tretjem poglavju opišemo arhitekturo nevronske mreže, ki jo uporabljamo. V četrtem poglavju so predstavljeni rezultati. V petem zaključnem poglavju pa podamo smernice za nadaljnje delo.

2. PODATKI

2.1 Pridobivanje podatkov

Nekatere zgodovinske podatke o nogometu smo našli na spletu, druge pa smo si morali zgenerirati sami. Za največje svetovne nogometne lige že obstajajo datoteke CSV za sezone od začetka 90. let naprej [4]. Najdemo lahko tudi datoteke CSV za vsa svetovna prvenstva [3].

Za slovensko nogometno ligo smo si morali napisati svoj program [6]. Ta je iz podatkov iz wikipedije [7] in uradne strani Prve slovenske nogometne lige [5] razbral prave informacije in jih shranil v datoteko CSV. Največja težava pri tem je bila, da so se imena klubov skozi leta veliko spreminjala in je bilo potrebno kar nekaj ročnega dela, da smo povezali skupaj prava imena klubov.

Prav tako smo morali napisati program za pridobivanje podatkov tekmovanja Copa America [1]. Ta je iz podatkov iz wikipedije [2] ustvaril datoteko CSV, vendar smo morali podatke za določene tekme vnesti sami, saj ima wikipedija nekonsistentno kodo.

2.2 Opis podatkov

Določene datoteke CSV so imele veliko število podatkov, zato smo vzeli le ključne za nas. Prvi podatek je kvote iz

DOI: https://doi.org/10.26493/978-961-7055-82-5.107-110

stavnice Bet365. Pogledali smo 3 kvote, in sicer za zmago domače ekipe, za neodločen izid in za zmago tuje ekipe. Tega podatka za slovensko ligo žal ni bilo mogoče najti. Naslednji podatek je bil ime ekipe. Sledilo je število zadetkov obeh ekip in pa kdo je zmagal, torej domača ekipa, neodločen izid ali pa tuja ekipa.

2.3 Preoblikovanje podatkov

Za vsako tekmo smo v upoštev vzeli vse prejšnje tekme te sezone. Tako smo si najprej izračunali zadete in prejete gole do trenutne tekme v sezoni. Potem je sledil izračun števila točk obeh ekip pred tekmo. Shranili smo si tudi število zmag, remijev in porazov obeh ekip pred trenutno tekmo. Za zadnjih 5 tekem obeh ekip smo beležili tudi formo. Če je ekipa zmagala je bil rezultat v podatkih označen z W, remi z D in poraz z L. Posebej smo z M označili tudi, če še tekma ni bila odigrana, torej v prvih petih krogih. Iz teh podatkov smo ugotovili tudi ali ima ekipa zaporedni niz zmag ali porazov. Pogledali smo za nize treh ali petih tekem. Naslednji podatek je bil, kateri zaporedni teden sezone je. Izračunali smo tudi razliko v golih in točkah med obema ekipama. Zabeležili smo tudi razliko med končnima položajema ekip v prejšnji sezoni. Če je katera izmed ekip bila prejšnjo sezono v nižji ligi, se ji je dodelila pozicija 30, da je se je videla razlika od ekip, ki so igrale v tej ligi. Kratek pregled podatkov lahko vidimo tudi v Tabeli 1. Preden so se podatki poslali v mrežo, se je za vse tekstovne podatke naredilo t.i. one-hot kodiranje, ki je vsak tekst enolično določilo.

Tabela 1: Vhodni podatki

Vhodni podatek	Razlaga
Kvote stavnice Bet365	trije podatki: kvota za zmago domače ekipe, kvota za neodločen izid in kvota za zmago tuje ekipe
Število zadetih golov	Koliko golov je zadela vsaka ekipa do tega trenutka v sezoni
Število prejetih golov	Koliko golov je prejela vsaka ekipa do tega trenutka v sezoni
Točke	Koliko točk imata obe ekipi v tem trenutku sezone
Pregled statistike tekem	Število zmag, remijev in porazov obeh ekip do tega trenutka v sezoni
Forma zadnjih 5 tekem	Za vsako izmed zadnjih 5 tekem ali je bila zmaga (W), remi (D), poraz (L) ali pa še ni bila odigrana (M)
Igralen teden	Kateri zaporedni teden sezone je
Niz zmag	Ali je ekipa zadnje 3 ali 5 tekem zmagala
Niz porazov	Ali je ekipa zadnje 3 ali 5 tekem izgubila
Razlika v golih	Število zadetih golov - število prejetiih golov
Razlika v točkah	Točke domače ekipe - točke tuje ekipe
Razlika v lanskih položajih na lestivic	Lanski končni položaj domače ekipe - lanski končni položaj tuje ekipe (če katera od ekip v prejšnji sezoni ni igrala v tej ligi, se ji dodeli pozicija 30, tako da je razlika od ostalih)
Ekipa	Ime ekipe
Rezultat	H - zmaga domače ekipe, NH - neodločeno ali poraz domače ekipe

3. ARHITEKTURA NEVRONSKE MREžE

Rekurentna nevronska mreža za razliko od navadnih, kot vhod ne dobi le vhodnega vektorja, temveč tudi stanje iz preteklosti. To je smiselno, kadar se neka informacija lahko skriva v samem zaporedju dogodkov. Najpogosteje uporabljen nevron pri teh tipih mreže pa je celica LSTM (Long short-term memory).

Uporabljena rekurentna nevronska mreža ima 4 plasti. Prve tri plasti imajo po 32 nevronov, zadnja pa 2 nevrona, saj imamo tudi toliko izhodov. Implementirali smo jo v programskem jeziku Python [8] in uporabili njegovo knjižnico za umetno inteligenco Keras [12]. V kodi samo ustvarimo sekvenčni model in dodamo število željenih plasti ter jim določimo število nevronov. Na koncu vsake plasti smo dodali še aktivacijsko funkcijo sigmoid, ki nam še dodatno normalizira vrednosti. Za prvimi tremi plastmi lahko vidimo tudi funkcjio izpuščanja (ang. Dropout function), ki poskrbi da se določen delež naučenega pozabi in tako ne prihaja do prekomernega prileganja.

4. REZULTATI

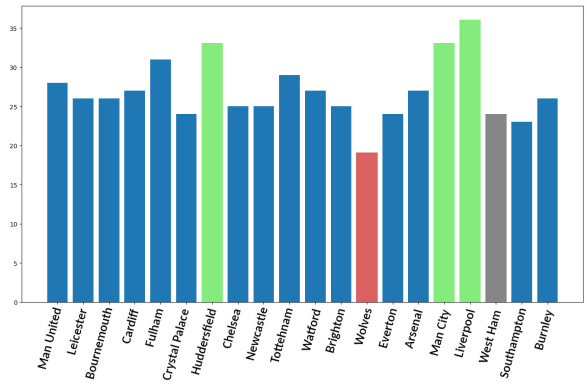
Rezultate razdelimo na dva dela. Najprej podamo napovedovanja tekem ligaških sezon, nato pa nogometnih turnirjev. Model naučimo vse do zadnje sezone oziroma turnirja, zadnjo pa uporabimo kot testno množico.

Za zaganjanje uporabljamo računalnik s procesorjem Intel i5-4750 3,6GHz, grafično kartico NVIDIA GeForce GTX 960 in 8 GB RAM pomnilnika.

4.1 Napovedovanje tekem ligaške sezone

Tabela 2: Uspešnost napovedovanja zmagovalca tekme glede na različno število plasti

	Mreža	
Liga	**2 plasti**	**4 plasti**
Italijanska liga	70,97%	**71,84%**
Angleška liga	70,53%	**70,79%**
Nemška liga	67,65%	**69,93%**
Španska liga	65,26%	**66,05%**
Francoska liga	**65,53%**	63,68%
Slovenska liga	61,11%	**61,67%**



Slika 1: Uspešnost napovedovanja za posamezno ekipo znotraj zadnje sezone angleške lige

Lige se načeloma med seboj razlikujejo le po številu ekip, edino v slovenski ligi je drugačen sistem igranja, saj imajo ekipe med seboj štiri tekme namesto dveh.

V Tabeli 2 vidimo primerjavo napovedovanja zmagovalca med mrežo z dvema plastema in štirimi plastmi nevronov LSTM. Vidimo, da se nevronska mreža z manj plastmi slabše odnese, ampak je pa načeloma manj častovno zahtevna in pride običajno do rezultata v polovičnem času v primerjavi z mrežo s štirimi plastmi. Ker za slovensko ligo nimamo podatkov za kvote na stavnicah in vidimo slabšo napovedljivost, lahko sklepamo, da so kvote pomemben faktor pri učenju nevronske mreže. Ostale lige pa imajo iste podatke na vhodu, kar pomeni, da sta italijanska in angleška liga manj naključni, kot pa francoska in španska, nemška pa je nekje vmes.

Če podrobneje pogledamo rezultate, lahko vidimo, da prihaja tudi do razlik med ekipami, saj se rezultate določene ekipe lažje napove kot druge, kar lahko za angleško ligo vidimo na Sliki 1. Najbolje se napoveduje Liverpool (36 od 38 pravilnih), Manchester City (33 od 38 pravilnih) in pa Huddersfield (33 od 38 pravilnih), ki so označeni z zeleno. Če pogledamo lestvico, so to prva, druga in pa zadnja ekipa v ligi, torej ji večje probleme delajo ekipe iz sredine lestvice, kar pa je tudi logično, saj so te majn konsistentne. Najslabše napovedljiv je Wolves, ki je označen z rdečo, in sicer le 19 od 38 tekem pravilnih. Če pogledamo ekipo iz sredine lestvice (10 mesto), West Ham, ki je označen s sivo in je pravilno napovedanih 24 od 38 tekem.

Tabela 3: Uspešnost napovedovanja, ali pade več kot 1,5 gola na tekmo

Liga	Uspešnost napovedi (v odstotkih)
Italijanska liga	74,74%
Angleška liga	79,74%
Nemška liga	84,64%
Španska liga	73,95%
Francoska liga	71,23%
Slovenska liga	81,67%

Tabela 4: Uspešnost napovedovanja, ali pade več kot 2,5 gola na tekmo

Liga	Uspešnost napovedi (v odstotkih)
Italijanska liga	53,68%
Angleška liga	57,63%
Nemška liga	63,73%
Španska liga	55,00%
Francoska liga	60,00%
Slovenska liga	62,78%

V Tabeli 3 je uspešnost napovedovanja, ali na tekmo pade več kot 1,5 gola. Napovedljivost tega izgleda precej boljša kot pa napovedljivost zmagovalca. Če podrobneje preučimo napovedovanje, lahko vidimo, da mreža v trenutku pride do lokalnega minimuma iz katerega se kasneje ne spremeni na bolje. Če pa pogledamo dejanske napovedi, pa lahko vidimo, da v večini primerov pride do spoznanja, da skoraj vedno pade več kot 1,5 gola na tekmo. Torej so te natančnosti bolj ali manj enake dejanskemu odstotkovnemu številu tekem, ko pade več kot 1,5 gola na tekmo.

V Tabeli 4 je uspešnost napovedovanja, ali na tekmo pade več kot 2,5 gola. Kot lahko vidimo je napovedljivost slaba. Ni enak problem kot pri več kot 1,5 gola na tekmo, saj tu ne pride do rešitve, da bi naj to bilo vedno res, ampak iz teh vhodnih podatkov ne moremo razbrati nekega vzorca, kdaj bi temu bilo tako.

V Tabeli 5 je uspešnost napovedovanja, ali obe ekipi na tekmi zadeneta vsaj en gol. Kot vidimo so rezultati podobno slabi kot pri napovedovanju, ali pade več kot 2,5 gola na tekmo, torej je uspešnost napovedovanja kar slaba.

4.2 Napovedovanje tekem nogometnih turnirjev

Malo drugačen sistem igranja pa se pojavlja na večjih nogometnih turnirjih kot je recimo svetovno prvenstvo, saj se

Tabela 5: Uspešnost napovedovanja, ali obe ekipi na tekmi zadeneta vsaj en gol

Liga	Uspešnost napovedi (v odstotkih)
Italijanska liga	62,37%
Angleška liga	54,74%
Nemška liga	59,48%
Španska liga	57,89%
Francoska liga	55,79%
Slovenska liga	66,11%

Tabela 6: Uspešnost napovedovanja zmagovalca tekme

Tekmovanje	Natančnost napovedovanja zmagovalca
Copa America	65,62%
Svetovno prvenstvo	65,28%

Tabela 7: Uspešnost napovedovanja skupinskega in izločevalnega dela tekmovanja

Tekmovanje	Uspešnost napovedi skupinskega dela	Uspešnost napovedi izločevalnega dela
Copa America	58,3%	87,5%
Svetovno prvenstvo	58,3%	72,9%

igranje deli na dva dela, to sta skupinski in izločevalni del.

V Tabeli 6 vidimo uspešnost napovedovanja zmagovalca tekme na svetovnem prvenstvu in pa turnirju Copa America. Obe tekmovanji sta približno isto napovedljivi. Imata tudi podoben trend, in sicer da ima mreža večje probleme z napovedovanjem skupinskega dela turnirja (pod 60 odstotkov), medtem ko izločilni del predvideva dobro, od 70 do 90 odstotkov natančno, kar lahko vidimo v Tabeli 7.

5. ZAKLJUčEK

Prispevek predstavlja napovedovanje določenih statistik nogometne tekme z rekurentno nevronsko mrežo LSTM. Ta se na preteklih rezultatih nauči določenih vzorcev in poskuša napovedati prihodnje tekme. Za to smo morali zbrati dovolj veliko podatkov in jih ustrezno predelati. Pomembna je bila tudi struktura nevronske mreže, torej število skritih plasti in število nevronov v teh plasteh.

Rekurentna nevronska mreža se uporabi na primeru napovedovanja zmagovalca nogometne tekme, ali pade več kot 1,5 ali 2,5 gola na tekmo in ali obe ekipi zadeneta. Uporabljene so angleška, francoska, italijanska, nemška, španska in slovenska liga ter tekmovanji Copa America in svetovno prvenstvo. Napovedljivost zmagovalca ligaških tekem je od 61 do 72 odstotkov, zmagovalca nogometnih turnirjev pa od 65 do 66 odstotkov. Napovedljivost, ali pade več kot 1,5 gola je od 71 do 85 odstotkov, ali pade več kot 2,5 pa od 53 do 64 odstotkov, podobni rezultati so tudi pri napovedovanju, ali obe ekipi zadeneta, in sicer od 54 do 67 odstotkov.

Nadaljne delo bi zajemalo pridobivanje večjega števila podatkov. Za nogometne lige imamo podatke od začetka 90. let naprej, kar za Slovenijo sicer predstavlja 100 odstotkov podatkov, ampak ostale lige pa segajo dalje v preteklost in bi ti podatki mogoče lahko pomagali. Še ena izboljšava bi bila drugačno preoblikovanje podatkov, oziroma uporaba drugih podatkov nogometne statistike (npr. število kotov ali število kartonov). Zanimivo bi bilo tudi preizkusiti drugačno arhitekturo nevronske mreže v smislu različnih plasti rekurentne nevronske mreže ali pa kombinacijo različnih tipov nevronskih mrež.

6. LITERATURA

[1] Copa america web scraper. https://github.com/planeer/CopaAmericaWebScraper, 19. 6. 2019.

[2] Copa américa. https://en.wikipedia.org/wiki/Copa_Am%C3%A9rica, 19. 6. 2019.

[3] Fifa world cup. https://www.kaggle.com/abecklas/fifa-world-cup, 19. 6. 2019.

[4] Football betting | football results | football bets | football odds. http://www.football-data.co.uk/, 19. 6. 2019.

[5] Plts - prva liga telekoma slovenije. https://www.prvaliga.si/prvaliga/default.asp, 19. 6. 2019.

[6] Prva liga web scraper. https://github.com/planeer/PrvaLigaWebScraper, 19. 6. 2019.

[7] Slovenian prvaliga. https://en.wikipedia.org/wiki/Slovenian_PrvaLiga, 19. 6. 2019.

[8] Python. https://www.python.org/, 2. 7. 2019.

[9] Dutch football prediction using machine learning classifiers. https://pdfs.semanticscholar.org/b347/e38d5c61a139115884fbff352221c4f7bfe1.pdf, 22. 8. 2019.

[10] Microsoft has perfectly predicted this stage of the world cup and it thinks brazil is finished. https://qz.com/231583/microsoft-world-cup-predictions-brazil-germany/, 22. 8. 2019.

[11] With germany's win microsoft perfectly predicted the world cup's knockout round. https://qz.com/233830/world-cup-germany-argentina-predictions-microsoft/, 22. 8. 2019.

[12] Keras. https://keras.io/, 29. 6. 2019.

[13] Žan Korpar. *Predikcija športnih rezultatov z uporabo strojnega učenja*. Fakulteta za elektrotehniko, računalništvo in informatiko Univerze v Mariboru, Maribor, 2018.

Izboljšanje zaznave sovražnega in zlonamernega govora s pomočjo slovarja besed

Sašo Kolac
Fakulteta za elektrotehniko, računalništvo in informatiko
Inštitut za računalništvo
Koroška cesta 46,
2000 Maribor, Slovenija
saso.kolac@student.um.si

Aljaž Soderžnik
Fakulteta za elektrotehniko, računalništvo in informatiko
Inštitut za računalništvo
Koroška cesta 46,
2000 Maribor, Slovenija
aljaz.soderznik@student.um.si

Simon Slemenšek
Fakulteta za elektrotehniko, računalništvo in informatiko
Inštitut za računalništvo
Koroška cesta 46,
2000 Maribor, Slovenija
simon.slemensek1@student.um.si

Borko Bošković
Fakulteta za elektrotehniko, računalništvo in informatiko
Inštitut za računalništvo
Koroška cesta 46,
2000 Maribor, Slovenija
borko.boskovic@um.si

POVZETEK

V članku je predstavljena metoda, ki temelji na optimizaciji predprocesiranja besedil z namenom izboljšati natančnost klasifikacije sovražnega govora z uporabo algoritmov strojnega učenja. Žaljive kratice in zaznamke nadomestimo z žetonom <curseword>, s čimer algoritmi lažje klasificirajo sovražni in zlonamerni govor. V članku so primerjani rezultati klasifikacij z in brez naše metode algoritmov naivni bayes, logistične regresije, podporni vektorji, naključni gozdovi, gradiento pospeševanje regresijskih gozdov, nevronske mreže in "Bagging"klasifikator. Testna besedila smo dobili iz socialnega omrežja Twitter.

Ključne besede

jezikovne tehnologije, klasifikacija, sovražni govor, strojno učenje, zlonamerni govor

1. UVOD

Svoboda izražanja je temeljna človekova pravica in predpogoj za obstoj demokratične družbe. Kot vse pravice, tudi svobode izražanja ni dopustno izrabljati na škodo drugih in je zamejena s človekovim dostojanstvom in z načelom varovanja javnega reda in miru. Nekatere oblike izražanja so zato zakonsko prepovedane ali veljajo za družbeno nesprejemljive, na primer grožnje, žalitve, komunikacija z namenom preslepitve in sovražni govor.

Sovražni in zlonamerni govor se navezujeta na besedila, ki so do posameznikov ali skupine ljudi žaljiva, prestrašujoča, poniževalna ali takšna, da spodbujajo nasilje [9]. Njegov cilj je razčlovečiti tiste, proti katerim je namenjen. Prepoznavanje takšnih besedil na spletu je dandanes zelo pomembno, saj imajo socialni mediji [6, 8] zelo velik vpliv na človeško psiho.

Družbena omrežja in razne platforme za objavljanje spletnih videoposnetkov v svojih pravilih prepovedujejo širjenje sovraštva proti družbenim skupinam, posameznicam in posameznikom. Prepovedujejo tudi grožnje in nadlegovanje ter omejujejo objavljanje vsebin, ki niso primerne za mladoletne. Facebook, Twitter, YouTube, Microsoft, Google+, Instagram in Snapchat so se s podpisom posebnega kodeksa zavezali, da bodo večino upravičenih prijav nezakonitega sovražnega govora pregledali v roku 24 ur in onemogočili dostop do teh vsebin ob upoštevanju lokalne in evropske zakonodaje.

Problem na katerega naletimo je preveliko število besedil oz. objav, da bi lahko ljudje ročno preverjali, če se besedilo dejansko uvršča pod sovražni govor. Eden izmed načinov za soočanje s tem problemom so algoritmi strojnega učenja [1, 7, 10], ki omogočajo dokaj uspešno zaznavo sovražnega govora.

V tem delu smo se osredotočili na predprocesiranje besedila v tvitih, tako da smo s pomočjo slovarja žaljivih besed zamenjali vse žaljive besede, ki so se pojavile v besedilih tvitoh z žetonom <curseword>. Nad predprocesiranim besedilom smo nato uporabili algoritme različnih vej strojnega učenja v programskem jeziku Python in medsebojno primerjali rezultate z in brez modifikacije pri predprocesiranju. Podatkovno bazo s približno 100.000 tviti smo pridobili iz repozitorija [3].

V pričujočem poglavju bomo predstavili ugotovitve sorodnih del. Sledi poglavje, ki opisuje predstavljeno metodo, ter potek eksperimenta z analizo rezultatov. Temu poglavju sledi poglavje, ki govori o naši razlagi rezultatov. Nazadnje sledi še zaključek, ki povzema ključne ugotovitve našega dela.

DOI: https://doi.org/10.26493/978-961-7055-82-5.111-114

2. SORODNA DELA

Podobno delo so opravili v [5], kjer so avtorji preizkusili pet modelov tradicionalnega strojnega učenja in nekaj modelov temelječih na nevronskih mrežah. Uporabili so naslednje modele:

1. Tradicionalni modeli strojnega učenja:
   - **Naivni Bayes (NB)**: z aditivno konstanto glajenja 1,
   - **Logistična regresija (LR)**: Linearna z L2 regularizacijsko konstanto 1 in z BFGS optimizacijo za omejen spomin,
   - **Metoda podpirnih vektorjev (SVM)**: Linearna z L2 regularizacijsko konstanto 1 in s funkcijo logistične izgube,
   - **Metoda naključnih gozdov (RF)**: Porazdelitev verjetnosti napovedi 10 naključnih dreves odločanja,
   - **Metoda gradientnega pospeševanja regresijskih gozdov (GBT)**: S konstanto učenja 1 in s funkcijo logistične izgube.
2. Modeli temelječi na nevronskih mrežah:
   - **Konvolucijska nevronska mreža (CNN)**: Modeli uporabljajo križno entropijo s softmax, kot funkcijo izgube in Adam kot optimizator,
   - **Ponavljajoče se dvosmerna nevronska mreža (RNN)**: Modeli uporabljajo križno entropijo s sigmoid, kot funkcijo izgube in Adam kot optimizator. Uporabljen je tudi vratni mehanizem GRU,
   - Variacije zgornjih dveh modelov.

Modele so preizkusili na sovražnih in zlonamernih besedilih iz socialnega omrežja Twitter. Predstavili so možnost izboljšave klasifikatorjev z uporabo dodatnih lastnosti in kontekstnih podatkov. Iz eksperimentov so ugotovili, da je najbolj natančen model temelječ na dvosmerni nevronski mreži z GRU, naučen na besednih lastnostih z metodo modulov za grozdenje latentnih tem. Izmerjena vrednost mere F1 za model je bila 80.5 %.

Članek [2] opisuje metodo za zaznavanje neprimernega obnašanja uporabnikov na Twitterju. Predstavili so tudi robustno metodologijo za ekstrakcijo besedil, uporabniških in omrežno temelječih atributov, preučevanja lastnostni agresivnih in ustrahujočih uporabnikov ter kakšne lastnosti jih ločijo od navadnih uporabnikov. Ugotovili so, da uporabniki, ki ustrahujejo, manj objavljajo na splet, so deležni manj socialnih skupin in so manj popularni od navadnih uporabnikov. Agresivni ljudje so relativno popularni in imajo več negativnosti v svojih objavah. Dokazali so, da lahko algoritmi strojnega učenja zaznajo agresivno in ustrahojoče vedenje uporabnikov z več kot 90 % AUC (angl. Area Under the Curve).

Delo [4] opisuje dvokoračno metodo klasifikacije zlonamernega jezika na twitterju in nato nadaljno delenje teh klasifikacij v specifične tipe. To metodo primerjajo z enokoračno, ki opravi samo eno več razredno klasifikacijo v delenje tipov za zaznavanje rasističnih in seksističnih besedil. Za enokoračno metodo so ugotovili, da je najboljša rešitev nevronska mreža HybridCNN z vrednostjo mere F1 82.7 %, za dvokoračno pa tradicionalni model logistične regresije z vrednostjo mere F1 82.4 %.

Članek [7] opisuje klasifikacijo besedil na twitterju v kategorije seksistično, rasistično ali nič od tega. Preizkusili so veliko različnih arhitektur globokega učenja in z eksperimenti pokazali, da so metode temelječe na semantični analizi besed boljše od metod temelječih na znakovnih in besednih n-gramih za približno 18 % vrednosti mere F1.

Članek [1] opisuje metodo za avtomatizirano zaznavo sovražnega govora na twitterju z ekstrakcijo lastnosti besedil na različnih konceptualnih nivojih in apliciranjem več razredne klasifikacije nad njimi. Sistem izkorišča variacije statističnih modelov in vzorce temelječe na pravilih. Obstaja tudi pomožni repozitorij z uteženimi vzorcei, ki izboljšajo natančnost tako, da povežejo besedilo z njegovim ocenjenim vnosom.

3. PREDSTAVITEV METODE IN EKSPERIMENTA

3.1 Metoda

Predstavljena metoda temelji na optimizaciji predprocesiranja besedil z namenom izboljšati natančnost klasifikacije sovražnega govora z uporabo algoritmov strojnega učenja.

Ključna razlika med predstavljeno metodo in metodami drugih avtorjev se skriva v koraku predprocesiranja. Pri predstavljeni metodi v tem koraku poleg vseh ostalih tipičnih elementov predprocesiranja, vse besede iz vsebine sporočila, ki so v slovarju žaljivih besed, zamenjamo z žetonom <curseword>, s čimer algoritmi lažje klasificirajo sovražni in zlonamerni govor.

Delovanje predstavljene metode je podrobno opisano v naslednjem podpoglavju, ko opisujemo izvedbo eksperimenta.

3.2 Eksperiment

Za implementacijo smo uporabili programski jezik Python. Ene izmed pomembnejših knjižnic, ki smo jih uporabili v implementaciji so Numpy, katera nudi hitre in učinkovite operacije na poljih. Programski jezik razširi v visoko nivojni jezik za manipulacijo numeričnih podatkov podobno jeziku Matlab. Sklearn smo uporabili za orodja, ki jih nudi za rudarjenje in analizo podatkov. Uporabili smo še knjižnico Pandas, ki nudi dobre podatkovne strukture in orodja za analizo podatkov in NLTK, ki je knjižnica za podporo programom, ki se ukvarjajo z naravnim jezikom ter Tweepy, ki poenostavi dostop do API-jev socialnega omrežja Twitter.

3.2.1 Pridobivanje korpusa

Za pridobivanje podatkov smo najprej morali na socialnem omrežju Twitter zaprositi za spletni račun razvijalca. S tem smo pridobili dostop do Twitterjeve spletne storitve v katero smo pošiljali IDje tvitov iz repozitorija [3], storitev pa nam je vrnila vsebino tvita. Zaradi omejitve števila klicov spletne storitve je pridobivanje vseh tvitov iz repozitorija trajalo približno 72 ur. Od 99799 tvitov, katere smo pridobili iz repozitorija [3] smo jih s pomočjo spletne storitve Tiwtter uspešno prenesli 79768, saj so Twitterjevi moderatorji nekaj

žaljivih in zlonamernih tvitov od takrat, ko je bila baza z IDji tvitov objavljena že uspešno odstranili. Izvajanje učnih algoritmov je implementirano v programskem jeziku Python po zgledu članka [5].

3.2.2 *Predprocesiranje*

Gre za klasično predprocesiranje besedila, kot je odstranitev posebnih znakov, "hash tags", spletne povezav in podobnih elementov z uporabo regularnih izrazov. Podatke razdelimo na 10 naključnih prečnih preverjanj (ang. folds). Z našo metodo, še v objavah iz Twitterja zamenjamo žaljive besede z žetonom <curseword> za nadaljno boljše učenje algoritmov. Kot žaljive besede smo smatrali vse besede, ki so se pojavile na seznamu Googlovih prepovedanih besed (vir: https://www.freewebheaders.com/full-list-of-bad-words-banned-by-google/).

3.2.3 *Implementiranje dodatnih učnih algoritmov*

Zraven naštetim učnim algoritmom članka [5] v poglavju 2 smo še implementirali nevronsko mrežo MLP in tradicionalni model "Bagging"klasifikator. Nevronsko mrežo smo izbrali zaradi dobrih rezultatov v sorodnih člankih. Za izbran tradicionalni model pa je znano, da dobro deluje kadar nimajo besede zelo podobnih pomenov.

3.2.4 *Učenje in evaluacija algoritmov*

Po predprocesiranju smo algoritme učili 2 krat. Enkrat z našo metodo optimizacije in enkrat brez. Po končanem učenjo smo še naredili evulacijo pridobljenih učnih modelov. Največjo pomembnost smo namenili meri F1 - enačba (1), ki predstavlja harmonično povprečje med senzitivnstjo in preciznostjo. Najboljšo vrednost doseže pri 1 (popolna preciznost in senzitivnost), najslabšo pa pri številu 0.

$$F_1 = 2 \cdot \frac{preciznost \cdot senzitivnost}{preciznost + senzitivnost} \tag{1}$$

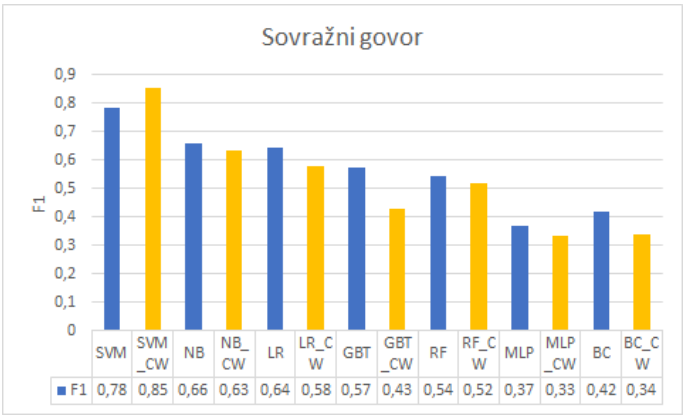
3.3 Analiza rezultatov

Pri vsaki metodi smo naredili deset poskusov z naključno izbranimi podatki. Uporabljali smo mero F1, preciznost, senzitivnost, mikro, makro ter uteženo povprečje. Primerjava rezultatov je narejena glede na povprečje vseh poskusov z uporabo mere F1, kot je razvidno na slikah 1 in 2.

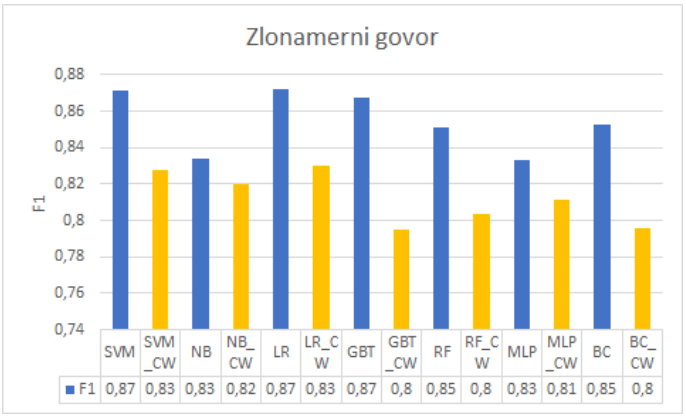
Slika 1 prikazuje mero F1 posameznih metod brez in z našo modifikacijo pri predprocesiranju besedila v tvitih. Pri sovražnem govoru nam je uspelo metodo SVM v povprečju izboljšati za 7 %, medtem ko so metode NB, LR, RF in MLP ostale primerljivo dobre. Občutno poslabšali pa sta se metodi GBT in BC (za 9 % oziroma 8 %).

Kot je razvidno iz slike 2 nam pri zaznavanju zlonamernega govora z modifikacijo pri predprocesiranju ni uspelo doseči boljše mere F1 pri nobeni metodi . Najslabše rezultate smo dosegli pri metodah GBT in BC (7 % oziroma 5 % poslabšanje s predprocesiranjem). Zakaj menimo, da je temu tako, smo opisali v naslednjem poglavju.

Pri zaznavi sovražnega govora sta nam metodi, ki smo ju dodatno implementirali (MLP in BC) prinesli najslabše rezultate mere F1 (37 % in 42 %) od vseh sedmih preizkušenih



Slika 1: Primerjava uspešnosti metod za zaznavo sovražnega govora



Slika 2: Primerjava uspešnosti metod za zaznavo zlonamernega govora

metod. Z modificiranim predprocesiranjem se je mera F1 še dodatno poslabšala na 33 % (MLP) in 34 % (BC).

Pri zaznavi zlonamernega govora sta se metodi MLP in BC odrezali malce bolje, še vedno pa nista bili med najnatančnejšimi. Brez uporabe slovarja pri predprocesiranju si je metoda BC delila četrto mesto z metodo RF, obe sta imeli mero F1 85 %. Medtem, ko si je metoda MLP delila zadnje mesto z metodo NB z mero F1 83 %. Ko smo vključili še predprocesiranje s slovarjem je metoda BC skupaj z metodama GBT in RF celo kazala najslabšo mero F1 izmed vseh metod 80 %, metoda MLP pa je bila le mesto nad njimi z mero F1 81 %.

Pri vseh metodah smo uporabili optimalne parametre, kateri so bili izračunani s pomožno funkcijo. Tako smo dobili za SVM parameter Alpha vrednost 0.0001, izguba je bila logaritmična s kaznijo 12, L1 razmerje 0.15, ter moč T parametra 0.5. Pri metodi NB smo dobili parameter Alpha z vrednostjo 1 in omogočili učenje z zgodovino. Pri LR metodi smo uporabili logaritem Lbgfs s kaznijo 12 ter omejili iteracije na 100. Za GBT metodo smo omejili globino na 1 in uporabili število približkov na 100 ter stopnjo učenja 0.1. Metoda RF je bila najmanj omejena, saj nismo omejevali globine in širine, uporabili pa smo kriterijsko metodo

Gini. Pri metodi MLP smo uporabili skrite nivoje velikosti 100, parameter Alpha z vrednostjo 0.0001, število iteracij smo omejili na 200 in uporabljali algoritem Adam. Za BC metodo smo uporabili 5 približkov z neomejenim številom opravil, vsi približki so imeli tudi namestnike po funkciji Bootstrap.

4. DISKUSIJA

S pomočjo predlaganega predprocesiranja smo poenotili žaljive besede. To pa je algoritmu SVM omogočilo, da je dosegel boljše rezultate. Razlog temu bi lahko bil ta, da so se vse žaljive besede preslikale v isti vektor. To pa je algoritmu omogočilo lažje določanje hiperravnin zaradi manjšega šuma v podatkih.

Pri predprocesiranju smo v slovarju imeli tako zlonamerne, kot tudi sovražne besede. Ko smo te besede zamenjali z enakim žetonom, smo posplošili klasifikacijo in zmanjšali razlike med vrstami besedil.

V prihodnosti bi lahko predprocesiranje izboljšali z večjim in boljšim slovarjem. Lahko bi ločili besede slovarja za vsak razred klasifikacije. Lahko bi v slovar vključili besedne zveze in fraze.

Natančnost klasifikacij, bi lahko izboljšali z bolj podrobnim deljenjem žaljivih besed na več različnih žetonov (v tem poskusu je samo 1 tip žetona). Nadalje bi jo izboljšali z razširjanjem slovarja žaljivih besed.

Lahko bi uporabili različni korpus za zlonamerni in sovražni govor. S tem bi dosegli bolj robustno in podrobno zaznavo žaljivega govora. Po primerjanju rezultatov, bi še lahko poskusili zgraditi nove hibridne in amsambelske metode učenja iz najboljših testiranih algoritmov.

5. ZAKLJUČEK

Na spletu je veliko različnih primerkov sovražnega in zlonamernega govora. Pri prepoznavi je eden izmed problemov možne variacije žaljivih besed, ki imajo podoben pomen. Rezultati našega dela so pokazali, da lahko naredimo majhno izboljšavo klasifikacij sovražnega govora takih primerov pri algoritmu SVM, če zmanjšamo raznolikost žaljivih besed s slovarjem žaljvih besed, ki te besede zamenja z žetonom.

Uspešnost naših metod smo ocenili z mero F1. Strokovnjaki so dosegli z njihovo najboljšo izkazano metodo SVM pri sovražnem govoru mero F1 z vrednostjo 78 %, z našo metodo slovarja žaljivih besed pa smo dosegli vrednost 85 %. Za zlonamerni govor so s to metodo dosegli vrednost mere F1 87 %, z našo metodo pa smo dosegli le 83 %. Pri ostalih rezultatih smo glede na mero F1 za sovražni in zlonamerni govor dosegli slabše rezultate pri metodah naivni bayes, linearna regresija, gradient boosting dreves in naključni gozd.

Prav tako je do razlik v rezultatih enakih testiranih učnih metod kot v članku [5], prišlo zaradi tega, ker nismo mogli pridobiti vseh tvitov iz baze, ker so upravljalci Twitterja med tem že izbrisali nekatere zlonamerne in sovražne tvite.

S pomočjo predlaganega predprocesiranja smo poenotili žaljive besede. To pa je algoritmu SVM omogočilo, da je dosegel boljše rezultate. Razog temu bi lahko bil to, da so se vse žaljive besede preslikale v enak vektor. To pa je algoritmu omogočilo lažje določanje hiperravnine, zaradi manjšega šuma v podatkih.

Literatura

[1] Sasha Uritsky Stan Matwin Amir H. Razavi, Diana Inkpen. Offensive language detection using multi-level classification. *Canadian AI 2010: Advances in Artificial Intelligence, Berlin, Heidelberg*, 2010.

[2] Jeremy Blackburn Emiliano De Cristofaro Gianluca Stringhini Athena Vakali Despoina Chatzakou, Nicolas Kourtellis. Mean birds: Detecting aggression and bullying on twitter. *WebSci '17, Troy, NY, USA*, 2017.

[3] Antigoni-Maria Founta, Constantinos Djouvas, Despoina Chatzakou, Ilias Leontiadis, Jeremy Blackburn, Gianluca Stringhini, Athena Vakali, Michael Sirivianos, and Nicolas Kourtellis. Large scale crowdsourcing and characterization of twitter abusive behavior. In *11th International Conference on Web and Social Media, ICWSM 2018*. AAAI Press, 2018.

[4] Pascale Fung Ji Ho Park. One-step and twostep classification for abusive language detection on twitter. *Proceedings of the First Workshop on Abusive Language Online, Vancouver, BC, Canada*, 2017.

[5] Younghun Lee, Seunghyun Yoon, and Kyomin Jung. Comparative studies of detecting abusive language on twitter. *Proceedings of the Second Workshop on Abusive Language Online (ALW2), Brussels, Belgium*, 2018.

[6] Chikashi Nobata, Joel Tetreault, Achint Thomas, Yashar Mehdad, and Yi Chang. Abusive language detection in online user content. *WWW '16 Proceedings of the 25th International Conference on World Wide Web, Montréal, Québec, Canada*, 2016.

[7] Manish Gupta Vasudeva Varma Pinkesh Badjatiya, Shashank Gupta. Deep learning for hate speech detection in tweets. *WWW '17 Companion Proceedings of the 26th International Conference on World Wide Web Companion, Perth, Australia*, 2017.

[8] Sara Sood, Judd Antin, and Elizabeth Churchill. Profanity use in online communities. *CHI '12 Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, Austin, Texas, USA*, 2012.

[9] William Warner and Julia Hirschberg. Detecting hate speech on the world wide web. *Proceedings of the Second Workshop on Language in Social Media, Montréal, Canada*, 2012.

[10] Guang Xiang, Bin Fan, Ling Wang, Jason Hong, and Carolyn Rose. Detecting offensive tweets via topical feature discovery over a large scale twitter corpus. *CIKM '12 Proceedings of the 21st ACM international conference on Information and knowledge management, Maui, Hawaii, USA*, 2012.

Investigating patterns using cellular automata

László Tóth
University of Szeged
13 Dugonics square
Szeged, Hungary
tothl@jgypk.u-szeged.hu

ABSTRACT

Complex systems based on a simple rule generates random, chaotic patterns. Examples of such models include cellular automata (CA). Over fifty years of CA research have been applied and analyzed in several CA fields, but their network topology study is still not significant.

How a cell can influence its environment is very important in CA models. The chaotic behaviors cause, that the information carried by the system can be of great importance for a pattern that will be passed down through generations.

During our research it was implemented a framework that can determine a communication network formed by Langton's CAs cells. This allows us to analyze factors that can influence the spread of information carried by cells. It is possible to simulate and analyze dynamically changing patterns with the proper parameterization of the developed system. These patterns created by cell cultures are grown in neurobiological labs.

Keywords

cellular automata, information spreading, cellular communication

Acknowledgments

The author was supported by the EU-funded Hungarian grant EFOP-3.6.2-16-2017-00015.

DOI: https://doi.org/10.26493/978-961-7055-82-5.115

StuCoSReC

University of Primorska Press
www.hippocampus.si
ISBN 978-961-7055-82-5