StuCoSReC

# Proceedings of the 2017 4th Student Computer Science Research Conference

University of Primorska Press

StuCoSReC
Proceedings of the 2017 4th Student Computer Science Research Conference

*Edited by*
Iztok Fister jr. and Andrej Brodnik

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

*Editor-in-Chief*
Jonatan Vinkler
*Managing Editor*
Alen Ježovnik

Koper, 2017

ISBN 978-961-7023-40-4 (pdf)
www.hippocampus.si/ISBN/978-961-7023-40-4.pdf
ISBN 978-961-7023-41-1 (html)
www.hippocampus.si/ISBN/978-961-7023-41-1/index.html

DOI: https://doi.org/10.26493/978-961-7023-40-4

Kataložni zapis o publikaciji (CIP) pripravili v Narodni in univerzitetni knjižnici v Ljubljani
COBISS.SI-ID=291995392
ISBN 978-961-7023-40-4 (pdf)
ISBN 978-961-7023-41-1 (html)

## Preface

Computer science is one of the quicker developing areas of Science. This area captures a lot of disciplines that have been developed independently. Thus, new disciplines have been arisen. On the other hand, the computer science starts connecting with the other natural sciences in order to draw an inspiration for solving their problems. Nowadays, incorporating principles from biology (e.g., Darwinian evolution or behavior of social living insects and animals) into the computer algorithms have been observed. In general, this allows enough material for a computer science conference. This proceeding contains papers as presented in the fourth Student Computer Science Research Conference 2017 (StuCoSRec) held in Maribor under the cover of the 21th Multi-Conference on Information Society in Ljubljana and ACM Slovenia.

The 4th Student Computer Science Research Conference is an answer to the fact that modern PhD. and MSc. programs foster early research activity among the students of computer science. The prime goal of the conference is to become a place for students to present their research work and hence further encourage students for an early research. Besides the conference, it also wants to establish an environment, where students from different institutions meet, let know each other, exchange the ideas, and nonetheless make friends and research colleagues. In line with this, we would like to exhibit research efforts of students in four Slovenian institutions, i.e., University of Maribor, University of Ljubljana, University of Primorska and Jožef Stefan international post-graduate school, neighboring countries, i.e., Austria, Croatia, Hungary and Italy, and the other countries in the world (e.g., one of the papers on out conference comes from India). At last but not least, the conference is also meant to serve as meeting place for students with senior researchers from different institutions.

Eleven papers addressed this conference, covering several topics of the computer science. All the papers were reviewed by two international reviewers and accepted for the oral presentation. This fact confirms a good work with authors in their research institutions. The content of the papers will be presented in three sections covering different areas of computer science and even robotics. The conference is dedicated to graduate and under-graduate students of computer science and is therefore free of charge. In line with this, the organizing committee would like to thank to the University of Maribor, i.e., the Faculty of Electrical Engineering and Computer Science (FERI) for the support. Especially, we are grateful to the UP IAM and prof. Andrej Brodnik for payment of conference costs arising during the organization. At the end, a special thank goes to the dean of UM FERI, prof. Borut Žalik for his unselfish support.

## Contents

# Parameterless Harmony Search for image Multi-thresholding

Krishna Gopal Dhal
Midnapore College (Autonomous),Dept. of Computer Sc. & Application
Midnapore (West)-721101, India
krishnacse42@gmail.com

Iztok Fister Jr.
Faculty of Electrical Engineering and Computer Science, University of Maribor
Smetanova 17, 2000 Maribor
iztok.fister1@um.si

Sanjay Das
University of Kalyani, Dept. of Eng. & Technological Studies
Kalyani-741235, India
dassanjay0810@hotmail.com

## ABSTRACT

The Harmony Search (HS) Algorithm is one of the efficient nature-inspired optimization algorithms which exhibits interesting search capability within less computational overhead. However, empirical studies showed that the main problem of this kind of algorithms is the proper setting of the associated parameters. HS associated with a few parameters and to find out the proper combination of the parameter values is time consuming. That's why a parameterless variant has been proposed here, which does not need the tuning over control parameters. The effect of different population size and stopping criterion has been considered in the experiment. The efficiency of the proposed HS is measured in Shannon's entropy based image multi-thresholding field.

## Keywords

Harmony Search, Control parameters, multi-thresholding, optimization.

## 1. INTRODUCTION

Recently, several nature-inspired optimization algorithms have been developed which mimic the behavior of natural and biological systems [5]. These algorithms are very powerful and effective for solving the real world optimization problems within a reasonable time [12]. In this study, the Harmony Search (HS) Algorithm [8] has been taken into consideration, and its extension to a parameterless variant. HS proves its effective performance in different optimization fields. But, the efficiency of the original HS depends on the proper tuning of the associated three control parameters. The proper setting of the values of these three parameters is very difficult for different kinds of problems. In order to overcome that problem, one parameterless variant of HS (PLHS) is developed here. In literature, parameterless variants of some algorithms, such as the Bat Algorithm (BA) [4, 3], Genetic Algorithm [7] and Differential Evolution (DE) [6], have been developed and proved their significant performance over a mathematical optimization field. One parameterless variant of HS is reported in literature where the associated parameters initialized by constant values, including population size [11]. In [11], an experiment with population size and stopping criterion was not performed. In our research paper, these experiments have been performed, inspired by methodologies the same as in [4, 3]. The proposed PLHS has been employed in a multi-thresholding based image segmentation domain, which is one of the significant pre-processing steps in computer vision application. Shannon entropy is used here as an objective function that maximizes the entropy of different regions in the image. Therefore, the organization of this paper is as follows. Section 2 presents the discussion about the HS and the associated control parameters. In section 3, a parameterless variant of HS has been presented and Shannon entropy based multi-thresholding is also explained. Experimental results are discussed in section 4. The paper is concluded in section 5.

## 2. HARMONY SEARCH (HS) ALGORITHM

In the Harmony Search (HS) Algorithm [8], the individual algorithms are called a "harmony" and they are represented by a real vector whose dimension is $n$. Let $X_i = \{x_i(1), x_i(2), \ldots, x_i(n)\}$ represent $i^{th}$ randomly generated harmony vector: $x_i(j) = l(j)+(u(j)-l(j))\times rand(0,1)\ for\ j = 1, 2, \ldots, n\ and\ i = 1, 2.., HMS$, where $l(j)$ and $u(j)$ denotes the upper bound and lower bound of the search space respectively and rand(0,1) is a uniform random number between 0 and 1. The HM memory is filled by the HMS harmony vector as follows:

$$HM = \begin{bmatrix} \mathbf{X_1} \\ \mathbf{X_2} \\ \vdots \\ \mathbf{X_{HMS}} \end{bmatrix} \quad (1)$$

### 2.1 Control Parameters in the HS Algorithm

The values of the control parameters affect the efficiency of the algorithm under experiment significantly.To control these parameters is the same as the controlling the exploration and exploitation efficiency of the considered algorithm. Therefore, the parameter tuning and control become an essential area in the nature-inspired optimization algorithms based research field. But, the setting of the control

**Algorithm 1** Harmony Search.

```
 1: Set the parameters HMS, HMCR, PAR, BW and
 2: NI or MAX_FE, which are discussed in sections 2.1 and 3.
 3: Initialize the HM and calculate the objective function value
 4: of each harmony vector.
 5: Improvise a New Harmony $X_{new}$ as follows:
 6: for $j = 1$ to $n$ do
 7:     if $r_1 < HMCR$ then
 8:         $x_{new} = x_a(j)$ where $a \in (1, 2, \ldots, HMS)$
 9:         if $r_2 < PAR$ then
10:             $x_{new} = x_{new}(j) \pm r_3 \times BW$
11:             where $r_1, r_2, r_3 \in rand(0, 1)$
12:         end if
13:         if $x_{new}(j) < l(j)$ then
14:             $x_{new}(j) = l(j)$
15:         end if
16:         if $x_{new}(j) > u(j)$ then
17:             $x_{new}(j) = u(j)$
18:         end if
19:     else
20:         $x_{new}(j) = l(j) + r \times (u(j) - l(j)$, where
21:         $r \in rand(0, 1)$
22:     end if
23: end for
24: Update the HM as $X_w = X_{new}$ if $f(X_{new}) < f(X_w)$,
25: where $f(\cdot)$ represents objective function value.
26: If stopping criterion is completed, the best harmony
27: vector $X_b$ in the HM is returned; Otherwise go back to line 6.
```

parameters depends crucially on the type of problems. HS is guided by five parameters which are as follows:

a. Population size or Harmony Memory size (HM)

b. Harmony-Memory Consideration Rate (HMCR)

c. Pitch Adjusting Rate (PAR)

d. Distance Bandwidth (BW)

e. Number of iterations (NI)

In order to evade the tuning process, a parameterless variant of the HS has been proposed which will be introduced in the next section.

## 3. DESIGN OF A PARAMETERLESS HARMONY SEARCH ALGORITHM

The performance of HS is influenced strongly by the values assigned to parameters, i.e. HM, HMCR, PAR, BW and NI. In order to develop a new parameterless HS (PLHS), the influence of these algorithm dependent parameters was studied, which demonstrates that some algorithm parameters such as, in this case, Number of Iterations (NI) could be set wisely, while the optimal setting of other parameters are not so easy. In the memory consideration step (i.e. line no. 7), depending on the HMCR new solution, ($x_{new}(j)$) is generated by selecting randomly from a value in the current existing HM i.e. from the set of $\{x_1(j), x_2(j), \ldots, x_{HMS}(j)\}$. For this operation, one random number $r_1$ is generated within the range of [0,1] from uniform distribution. If $r_1$ is less than HMCR, the decision variable $x_{new}(j)$ is generated from memory consideration, otherwise, it is generated from random initialization between $[l(j), u(j)]$ (i.e. line no 20), which are the search boundaries. Therefore, HMCR controls the global search or exploration capability of the HS. Equation no. (2) represents the action of HMCR. Every component which is obtained by memory consideration is checked further to determine whether it should be pitch adjusted or not. The Pitch Adjustment Rate (PAR) is defined as assignment of the frequency adjustment and the bandwidth factor (BW) to control the local search of the HM. The pitch-adjustment decision is calculated by equation no. (3).

Table 1: Parameters' Setting

| Alg. | NI | HMCR | PAR | BW | HM |
|---|---|---|---|---|---|
| HS | 5000 | 0.75 | 0.5 | 0.5 | 100 |
| PLHS | 5000 | 0.8 | 0.5 | 0.55 | [10, 1280] |

$$x_{new}(j) = \begin{cases} x_i(j) \in \{x_1(j), x_2(j), \ldots, x_{HMS}(j)\} \\ \text{with probability HMCR,} \\ l(j) + (u(j) - l(j) \times rand(0,1) \\ \text{with probability 1-HMCR.} \end{cases} \tag{2}$$

$$x_{new}(j) = \begin{cases} x_{new}(j) = x_{new}(j) = x_{new}(j) \pm rand(0,1) \times BW \\ \text{with - probability PAR,} \\ x_{new}(j) \\ \text{with probability (1-PAR).} \end{cases} \tag{3}$$

Finding the Stopping Criterion is very crucial for different optimization algorithms. In the experiments, two Stopping Criteria (SC) have been considered, which are as follows:

**$1^{st}$ Stopping Criterion (SC1):** First is the number of times in which the best fitness values remain unchanged. Therefore, if the fitness value for the best harmony remains the same in 10% of the total Number of Iterations (NI), then the HS is stopped.
**$2^{nd}$ Stopping Criterion (SC2):** The other Stopping Criterion is the number of Fitness Evaluations (FEs), and the maximum number of FEs (i.e. $MAX\_FE$) has been taken as 10,000.

The values of the parameters of the traditional HS are same as [8], but the parameters' values of the proposed PLHS are set from the experience that is given as Table 1.

The population size is also a crucial parameter in HS. It is also reported that an appropriate population size (i.e. HM) is significant to both run-time efficiency and effectiveness [10, 1]. A lower population size may suffer from lack of diversity, whereas the higher population size may affect the convergence speed. In traditional HS, it is set as 100, but, in PLHS, it is varied in the interval $HM \in [10, 1280]$ such that each population size is multiplied by two in each run starting with HM=10. Therefore, eight instances of PLHS have been executed (i.e. PL-1, ..., PL-8) and the best one is considered by the user.

## 3.1 Multi-Level Shannon Entropy

Let $P = (p_1, p_2, p_3, \ldots, p_n)\ in\delta_n$, where $\delta_n\{(p_1, p_2, \ldots, p_n) \mid p_i \geq 0,\ i = 1, 2, \ldots, n,\ n \geq 2,\ \sum_{i=1}^{n} p_i = 1\}$ is a set of discrete finite $n$-ary probability distributions. Then entropy of the total image can be defined as [9]:

$$H(P) = -\sum_{i=1}^{n} p_i log_2 p_i \tag{4}$$

$I$ denotes an 8 bit gray level digital image of dimension $M \times N$. $P$ is the normalized histogram for image with $L = 256$ gray levels. Now, if there are $n-1$ thresholds $(t)$, partitioning the normalized histogram into $n$ classes, then the entropy for each class may be computed as:

$$\begin{aligned} H_1(t) &= -\sum_{i=0}^{t_1} \frac{p_i}{P_1} ln \frac{p_i}{P_1}, \\ H_2(t) &= -\sum_{i=t_1+1}^{t_2} \frac{p_i}{P_2} ln \frac{p_i}{P_2}, \\ H_n(t) &= -\sum_{i=t_{n-1}+1}^{L-1} \frac{p_i}{P_n} ln \frac{p_i}{P_n}, \end{aligned} \tag{5}$$

where

$$P_1(t) = \sum_{i=0}^{t_1} p_i,\ P_2(t) = \sum_{i=t_1+1}^{t_2} p_i, \ldots,\ P_n(t) = \sum_{i=t_{n-1}+1}^{L-1} p_i, \tag{6}$$

and for ease of computation, two dummy thresholds $t_0 = 0$, and $t_n = L-1$ are introduced with $t_0 < t_1 < \ldots < t_{n-1} < t_n$. Then the optimum threshold value can be found by:

$$\varphi(t_1, t_2, \ldots, t_n) = Arg\ max([H_1(t) + H_2(t) + \ldots + H_n(t)]) \tag{7}$$

## 4. EXPERIMENTAL RESULTS

The experiment has been performed over 50 benchmark images with MatlabR2009b with Windows-7 OS, x32-based PC, Intel(R) Pentium (R)-CPU, 2.20 GHz with 2 GB RAM. The purpose of our experiment is to prove how much the efficiency of the HS is affected by employing different population sizes (i.e. HM) and stopping criteria. In line with this, the traditional HA with 100 numbers of individuals is also compared with these eight PLHS. The traditional HS and PLHSs are run to solve the Shannon's entropy based multi-thresholding problem where optimal threshold values are found by solving equation no. 7. Both HS and PLHSs are stochastic in nature, and that's why each algorithm is run 30 times for each image. The number of thresholds used in this experiment is 2, 3, 4 and 5-level thresholding. The efficiency and consistency of the algorithms are evaluated and compared in terms of Computational Time (CT), Mean Fitness value (Fit$_m$) and Standard Deviation (Fit$_{std}$) for each problem. On the other hand, the image quality assessment metric, known as Peak-Signal to Noise Ratio (PSNR) [2] is computed to assess the similarity of the segmented image against the original image. It is actually a distortion metric,which depends crucially on Mean-Squared Error (MSE), which is defined as:

$$MSE(f, G) = \frac{\sum_{i=0}^{N-1} \sum_{j=0}^{M-1} [f(i,j) - G(i,j)]^2}{M \times N}, \tag{8}$$

where $f$ and $G$ and are the inputs and output image respectively. $M$ and $N$ are the numbers of rows and columns of the image. The PSNR is calculated as follows:

$$PSNR(f, G) = 10 log_{10} \frac{(L-1)^2}{MSE(f,G)}. \tag{9}$$

Greater values of PSNR represent better segmentation.

Table 2: Comparison and ranking based on Computational Time (CT), Mean Fitness value (Fit$_m$), Standard Deviation (Fit$_{std}$) and PSNR for 2-level multi-thresholding.

| Alg. | CT | Fit$_m$ | Fit$_{std}$ | PSNR |
|---|---|---|---|---|
| HS | 1.92 (5) | **18.8027 (1)** | **0 (1)** | **14.60 (1)** |
| PL-1 | 1.90 (4) | **18.8027 (1)** | **0 (1)** | **14.60 (1)** |
| PL-2 | 2.22 (8) | **18.8027 (1)** | **0 (1)** | **14.60 (1)** |
| PL-3 | 2.31 (9) | **18.8027 (1)** | **0 (1)** | **14.60 (1)** |
| PL-4 | 1.73 (3) | **18.8027 (1)** | **0 (1)** | **14.60 (1)** |
| PL-5 | 2.14 (7) | **18.8027 (1)** | **0 (1)** | **14.60 (1)** |
| PL-6 | 2.10 (6) | 18.8010 (7) | 1.1024e-13 (9) | 14.38 (7) |
| PL-7 | 1.68 (2) | **18.7820 (9)** | 1.0669e-13 (8) | 14.31 (9) |
| PL-8 | **1.02 (1)** | 18.7985 (8) | 5.3334e-16 (7) | 14.38 (7) |

## 4.1 Result section for 1st Stopping Criterion (SC1)

Tables numbers 2 to 5 represent the experimental results using the first Stopping Criterion (SC1), and the Tables demonstrate clearly that population size has a great impact over the performance of the HS. According to the Mean Fitness value (Fit$_m$) and Standard Deviation (Fit$_{std}$), less population size produces better output, and that's why PLHS with HM=10 outperforms others. But in terms of computational time, PLHSs with larger population size are better, but they fail to produce the best solution in terms of the objective function. Therefore, it could be said that the larger population performs premature convergence of the HS. Stability (i.e. Fit$_{std}$) also decreases when the size of the population increases. But, experimental study also indicates that when the number of threshold levels increases, stability of the PLHS with lower population size also decreases, but the values of the Fit$_{std}$ remains same for PLHS with larger population size. Therefore, larger population size may help to solve the more complex problems. To get an average performance of HS and PLHSs over different threshold levels, the sum of the algorithms' rankings of each problem has been presented in Table 10, and general ranking is also done based on the sum of the rankings. Table 10 also demonstrates that PLHS with HM=10 and 20 are the best variants in terms of Fit$_m$ and PLHS with HM=10 is the best variant depending on Fit$_{std}$ and PSNR. But, PLHS with HM=1280 is the best when considering the Computational Time (CT). It could be concluded that the average performance of the traditional HS is good as it gets middle ranks in Table 10 by considering all efficiency assessment metrics. Fig. 1 represents the thresholded images and histograms for PL-1, whereas Fig. 2 represents the convergence curves of PL-1 (HM=10) for 2, 3, 4 and 5 level thresholdings of Fig. 1(j).

Table 3: Comparison and ranking based on Computational Time (CT), Mean Fitness value ($\text{Fit}_m$), Standard Deviation ($\text{Fit}_{std}$) and PSNR for 3-level multi-thresholding.

| Alg. | CT | $\text{Fit}_m$ | $\text{Fit}_{std}$ | PSNR |
|---|---|---|---|---|
| HS | 2.33 (5) | **23.4286 (1)** | **0 (1)** | 16.88 (2) |
| PL-1 | 2.17 (4) | **23.4286 (1)** | **0 (1)** | 16.88 (2) |
| PL-2 | 2.58 (7) | **23.4286 (1)** | **0 (1)** | 16.88 (2) |
| PL-3 | 2.49 (6) | **23.4286 (1)** | **0 (1)** | 16.88 (2) |
| PL-4 | 2.97 (8) | **23.4286 (1)** | **0 (1)** | 16.88 (2) |
| PL-5 | 3.01 (9) | **23.4286 (1)** | **0 (1)** | 16.88 (2) |
| PL-6 | 1.70 (3) | 23.4065 (7) | 1.0092e-14 (8) | **16.92 (1)** |
| PL-7 | 1.15 (2) | 23.4011 (8) | 1.0262e-15 (7) | 16.80 (9) |
| PL-8 | **1.06 (1)** | 23.3893 (9) | 2.0334e-14 (9) | 16.84 (8) |

Table 4: Comparison and ranking based on Computational Time (CT), Mean Fitness value ($\text{Fit}_m$), Standard Deviation ($\text{Fit}_{std}$) and PSNR for 4-level multi-thresholding.

| Alg. | CT | $\text{Fit}_m$ | $\text{Fit}_{std}$ | PSNR |
|---|---|---|---|---|
| HS | 4.57 (5) | 27.7252 (6) | 2.0021e-15 (5) | 19.03 (4) |
| PL-1 | 6.45 (9) | **27.7275 (1)** | **0 (1)** | **19.05 (1)** |
| PL-2 | 6.07 (8) | **27.7275 (1)** | **0 (1)** | **19.05 (1)** |
| PL-3 | 5.68 (6) | **27.7275 (1)** | **0 (1)** | **19.05 (1)** |
| PL-4 | 4.51 (4) | 27.7272 (4) | 2.0001e-15 (4) | 19.02 (5) |
| PL-5 | 4.89 (7) | 27.7256 (5) | 1.0121e-14 (8) | 18.88 (6) |
| PL-6 | 1.98 (3) | 27.6518 (9) | 1.0093e-14 (7) | 18.61 (9) |
| PL-7 | 1.35 (2) | 27.6692 (7) | 1.0342e-14 (9) | 18.80 (7) |
| PL-8 | **1.16 (1)** | 27.6687 (8) | 2.0302e-15 (6) | 18.78 (8) |

Table 5: Comparison and ranking based on Computational Time (CT), Mean Fitness value ($\text{Fit}_m$), Standard Deviation ($\text{Fit}_{std}$) and PSNR for 5-level multi-thresholding.

| Alg. | CT | $\text{Fit}_m$ | $\text{Fit}_{std}$ | PSNR |
|---|---|---|---|---|
| HS | 5.07 (6) | 31.6959 (4) | 3.0225e-13 (5) | 20.33 (4) |
| PL-1 | 8.23 (9) | **31.6975 (1)** | **2.0543e-14 (1)** | **20.40 (1)** |
| PL-2 | 7.89 (8) | **31.6975 (1)** | 2.0888e-14 (2) | 20.38 (2) |
| PL-3 | 6.03 (7) | 31.6959 (4) | 2.0786e-13 (4) | 20.34 (3) |
| PL-4 | 4.97 (5) | 31.6961 (3) | 2.0031e-12 (8) | 20.32 (5) |
| PL-5 | 4.94 (4) | 31.6804 (6) | 1.9021e-12 (7) | 20.16 (6) |
| PL-6 | 2.78 (3) | 31.6260 (7) | 1.8763e-12 (6) | 20.14 (7) |
| PL-7 | 1.75 (2) | 31.5988 (9) | 2.0042e-12 (9) | 20.10 (8) |
| PL-8 | **1.18 (1)** | 31.6055 (8) | 1.0030e-13 (3) | 20.11 (9) |

Table 6: Comparison and ranking based on Computational Time (CT), Mean Fitness value ($\text{Fit}_m$), Standard Deviation ($\text{Fit}_{std}$) and PSNR for 2-level multi-thresholding.

| Alg. | CT | $\text{Fit}_m$ | $\text{Fit}_{std}$ | PSNR |
|---|---|---|---|---|
| PL-1 | 19.06 (6) | **18.8027 (1)** | **0 (1)** | **14.60 (1)** |
| PL-2 | 19.58 (8) | **18.8027 (1)** | **0 (1)** | **14.60 (1)** |
| PL-3 | 18.94 (3) | **18.8027 (1)** | **0 (1)** | **14.60 (1)** |
| PL-4 | 18.85 (2) | **18.8027 (1)** | **0 (1)** | **14.60 (1)** |
| PL-5 | **18.43 (1)** | **18.8027 (1)** | **0 (1)** | **14.60 (1)** |
| PL-6 | 19.07 (7) | **18.8027 (1)** | **0 (1)** | **14.60 (1)** |
| PL-7 | 19.04 (5) | **18.8027 (1)** | **0 (1)** | **14.60 (1)** |
| PL-8 | 19.03 (4) | **18.8027 (1)** | **0 (1)** | **14.60 (1)** |

Table 7: Comparison and ranking based on Computational Time (CT), Mean Fitness value ($\text{Fit}_m$), Standard Deviation ($\text{Fit}_{std}$) and PSNR for 3-level multi-thresholding.

| Alg. | CT | $\text{Fit}_m$ | $\text{Fit}_{std}$ | PSNR |
|---|---|---|---|---|
| PL-1 | 20.70 (7) | **23.4286 (1)** | **0 (1)** | **16.88 (1)** |
| PL-2 | 20.31 (4) | **23.4286 (1)** | **0 (1)** | **16.88 (1)** |
| PL-3 | 20.86 (8) | **23.4286 (1)** | **0 (1)** | **16.88 (1)** |
| PL-4 | 20.47 (5) | **23.4286 (1)** | **0 (1)** | **16.88 (1)** |
| PL-5 | 20.63 (6) | **23.4286 (1)** | **0 (1)** | **16.88 (1)** |
| PL-6 | 19.08 (3) | **23.4286 (1)** | **0 (1)** | **16.88 (1)** |
| PL-7 | 19.04 (2) | **23.4286 (1)** | **0 (1)** | **16.88 (1)** |
| PL-8 | **19.03 (1)** | **23.4286 (1)** | **0 (1)** | **16.88 (1)** |

### 4.2 Result section for 2nd Stopping Criterion (SC2)

Tables 6- 9 demonstrate the results of the PLHSs using $MAX\_FE$ as the stopping criterion. From the analysis of the experimental results, it can be said easily that, when population size resides within 40 and 160, then the HS gives the best result i.e. PL-3, 4 and 5 are the best among all the PLHSs. According to $\text{Fit}_m$ and $\text{Fit}_{std}$, PL-1, PL-2 and PL-3 are better than others. Large population size (i.e. 360, 640, 1280) are efficient in terms of CT only. Stability ($\text{Fit}_{std}$) and $\text{Fit}_m$ decreases when population size resides within [360, 1280]. $MAX\_FE$ based Stopping Criterion helps to reduce the stability issue compared to the NI based Stopping Criterion, which could be verified easily from the values of the corresponding tables. But, $MAX\_FE$ increases the CT rapidly. In Table 10, average efficiency has been computed by summing the ranking over different levels of thresholding and again, ranking is done based on the total ranking. PL-1 gives the best average result by considering , and PSNR, whereas, PL-8 takes less time to converge compare to others. But the convergence may be premature convergence according to the values of . Fig. 3 represents the convergence curves of PL-1 using the $MAX\_FE$ based Stopping Criterion.

## 5. CONCLUSION

Efficiency of the population-based nature-inspired optimization algorithms are significantly depends on the proper tuning of algorithm's control parameters. But finding the proper combination of the values of these parameters is very tedious work and problem specific. In order to overcome that one parameterless variant of HS (PLHS) has been developed. The most of the parameters are set from the experimental study. But the population size has been varied in the interval $\in$ [10, 1280] to evaluate the effect of the different population size over the efficiency of the HS. Two stopping

Table 8: Comparison and ranking based on Computational Time (CT), Mean Fitness value ($\text{Fit}_m$), Standard Deviation (($\text{Fit}_{std}$) and PSNR for 4-level multi-thresholding.

| Alg. | CT | $\text{Fit}_m$ | $\text{Fit}_{std}$ | PSNR |
|---|---|---|---|---|
| PL-1 | 21.22 (7) | **27.7275(1)** | **0 (1)** | **19.05 (1)** |
| PL-2 | 21.41 (8) | **27.7275(1)** | **0 (1)** | **19.05 (1)** |
| PL-3 | 21.02 (5) | **27.7275(1)** | **0 (1)** | **19.05 (1)** |
| PL-4 | 20.82 (4) | **27.7275(1)** | **0 (1)** | **19.05 (1)** |
| PL-5 | 21.04 (6) | 27.7256(6) | 2.0100e-12 (6) | 18.85 (6) |
| PL-6 | 20.56 (3) | 27.7259(5) | 2.0093e-12 (5) | 18.87 (5) |
| PL-7 | 20.54 (2) | 27.7239(7) | 3.0303e-12 (7) | 18.83 (7) |
| PL-8 | **19.91 (1)** | 27.7194(8) | 1.0001e-11 (8) | 18.83 (7) |

Table 9: Comparison and ranking based on Computational Time (CT), Mean Fitness value ($\text{Fit}_m$), Standard Deviation ($\text{Fit}_{std}$) and PSNR for 5-level multi-thresholding.

| Alg. | CT | $\text{Fit}_m$ | $\text{Fit}_{std}$ | PSNR |
|---|---|---|---|---|
| PL-1 | 21.93 (7) | **31.6975 (1)** | **0 (1)** | **20.40 (1)** |
| PL-2 | 20.88 (2) | **31.6975 (1)** | **0 (1)** | **20.40 (1)** |
| PL-3 | 21.73 (6) | 31.6959 (3) | 2.1044e-15 (3) | 20.34 (3) |
| PL-4 | 21.55 (5) | 31.6957 (4) | 1.0030e-13 (5) | 20.34 (3) |
| PL-5 | 21.46 (4) | 31.6952 (5) | 1.9001e-13 (7) | 20.29 (5) |
| PL-6 | 21.02 (3) | 31.6937 (6) | 1.7703e-13 (6) | 20.24 (6) |
| PL-7 | 21.94 (8) | 31.6930 (7) | 2.0030e-13 (8) | 20.23 (7) |
| PL-8 | **20.45 (1)** | 31.6862 (8) | 2.0011e-14 (4) | 20.20 (8) |

criteria have been used here for analysis the efficiency of the PLHSs. Analysis of the experimental results prove that PLHSs with lower population size are better for maximizing the Shannon's entropy based objective function with less standard deviation but with more computational time when Iteration based stopping criterion is used. Larger population size helps to reduce the computational time, but may performs premature convergence. In the case of $MAX\_FE$ based stopping criterion, HS with population size $\in$ [40, 160] gives best and consistent output. Here large population size also affect the stability issue. But, $MAX\_FE$ based stopping condition assists to reduce the stability issue with large computational time compare to iteration based termination condition. Therefore, development of robust adaptive nature-inspired optimization algorithms algorithms where all parameters including population size and stopping criterion for a set of problems are automatically adapted is still a big problem in this optimization field. In the future, an extensive study and systematic analysis of the parameters of different nature-inspired optimization algorithms are needed over a different set of problems.

## 6. REFERENCES

[1] Thomas Back, David B. Fogel, and Zbigniew Michalewicz, editors. *Handbook of Evolutionary Computation*. IOP Publishing Ltd., Bristol, UK, UK, 1st edition, 1997.

[2] Ashish Kumar Bhandari, Anil Kumar, S Chaudhary, and Girish Kumar Singh. A novel color image multilevel thresholding based segmentation using nature inspired optimization algorithms. *Expert Systems with Applications*, 63:112–133, 2016.

[3] Iztok Fister Jr, Iztok Fister, and Xin-She Yang. Towards the development of a parameter-free bat algorithm. In *StuCoSReC: Proceedings of the 2015 2nd Student Computer Science Research Conference*, pages 31–34, 2015.

[4] Iztok Fister Jr, Uroš Mlakar, Xin-She Yang, and Iztok Fister. Parameterless bat algorithm and its performance study. In *Nature-Inspired Computation in Engineering*, pages 267–276. Springer, 2016.

[5] Iztok Fister Jr., Xin-She Yang, Iztok Fister, Janez Brest, and Dušan Fister. A brief review of nature-inspired algorithms for optimization. *Elektrotehniški vestnik*, 80(3):116–122, 2013.

[6] Teo J. and M.Y. Hamid. A parameterless differential evolution optimizer. In *5th International Conference on Systems Theory and Scientific Computation (ISTASC'05), Malta*, pages 330–335, 2005.

[7] Fernando G Lobo and David E Goldberg. An overview of the parameter-less genetic algorithm. *Urbana*, 51:61801, 2003.

[8] Diego Oliva, Erik Cuevas, Gonzalo Pajares, Daniel Zaldivar, and Marco Perez-Cisneros. Multilevel thresholding segmentation based on harmony search optimization. *Journal of Applied Mathematics*, 2013, 2013.

[9] Soham Sarkar, Sujoy Paul, Ritambhar Burman, Swagatam Das, and Sheli Sinha Chaudhuri. A fuzzy entropy based multi-level image thresholding using differential evolution. In *International Conference on Swarm, Evolutionary, and Memetic Computing*, pages 386–395. Springer, 2014.

[10] Ruhul Amin Sarker and MF Azam Kazi. Population size, search space and quality of solution: An experimental study. In *Evolutionary Computation, 2003. CEC'03. The 2003 Congress on*, volume 3, pages 2011–2018. IEEE, 2003.

[11] LA Silva, PB Ribeiro, GH Rosa, KAP Costa, and João Paulo Papa. Parameter setting-free harmony search optimization of restricted boltzmann machines and its applications to spam detection. In *12th International Conference Applied Computing*, pages 142–150, 2015.

[12] Xin-She Yang. *Nature-inspired optimization algorithms*. Elsevier, 2014.

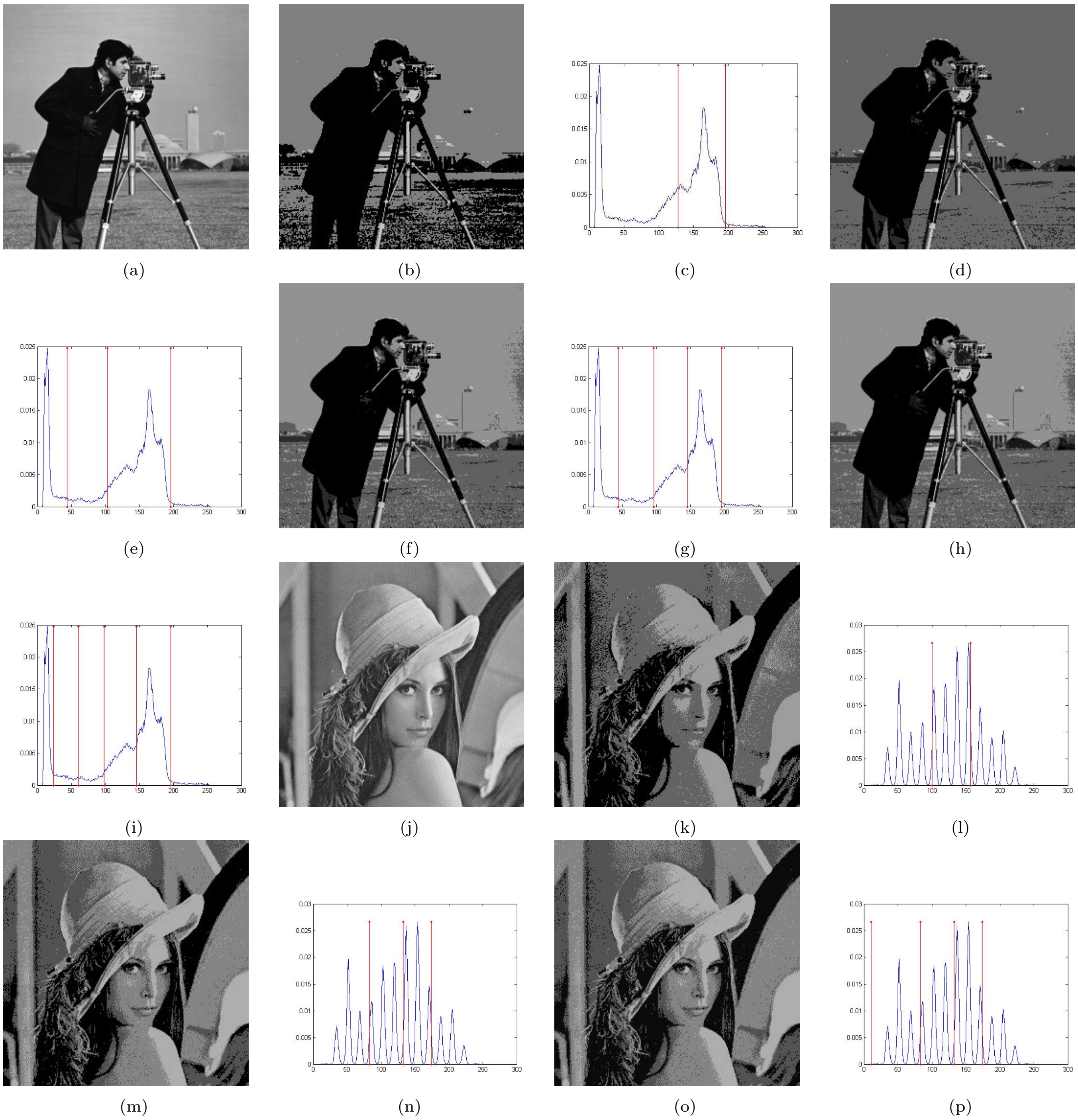
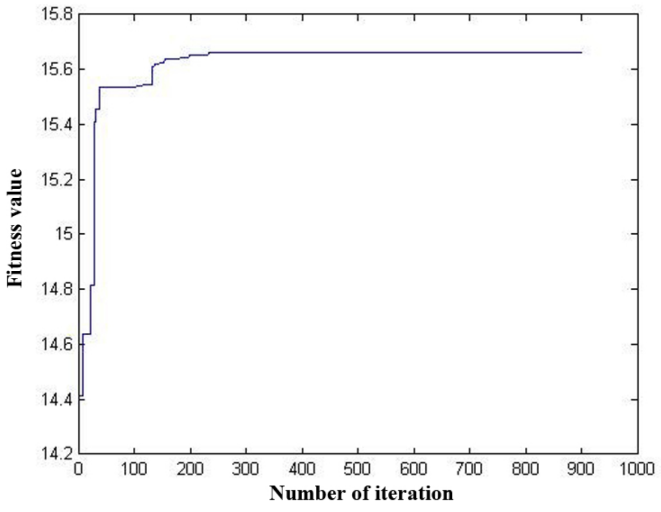


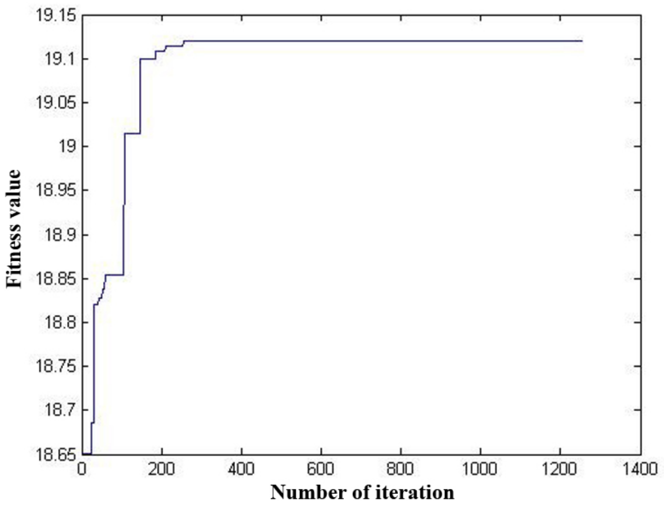
Figure 1: Results of PL-1 using SC1. (a) & (j) Original image; (b), (c) & (k), (l) are the result of 2-level thresholding; (d), (e) & (m), (n) are the result of 3-level thresholding; (f), (g) & (o), (p) are the result of 4-level thresholding; (h), (i) & (q), (r) are the result of 5-level thresholding. (Red lines point the thrershold values).

Table 10: The sum of ranking of each algorithm based on CT, Fit$_m$, Fit$_{std}$, PSNR and general ranking based on the total ranking.

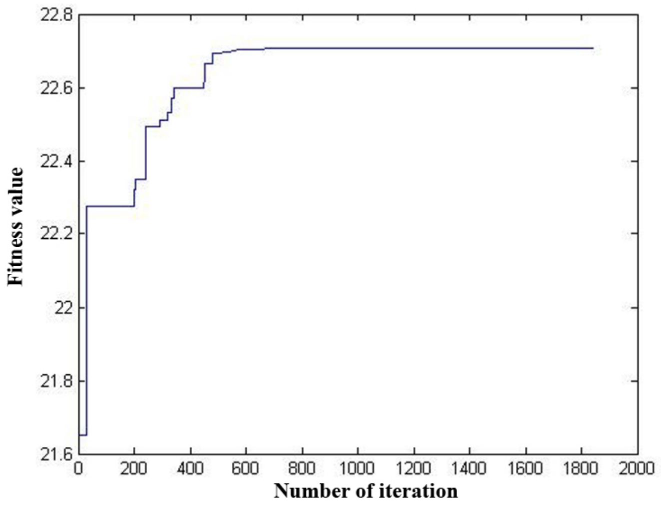
| Algo. | CT(SC1) | | CT(SC2) | | Fit$_m$(SC1) | | Fit$_m$(SC2) | | Fit$_{std}$(SC1) | | Fit$_{std}$(SC2) | | PSNR(SC1) | | PSNR(SC2) | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | T.R | G.R | T.R | G.R | T.R | G.R | T.R | G.R | T.R | G.R | T.R | G.R | T.R | G.R | T.R | G.R |
| HS | 21 | 5 | - | - | 12 | 5 | - | - | 12 | 4 | - | - | 11 | 4 | - | - |
| PL-1 | 26 | 6 | 27 | 8 | 4 | 1 | 4 | 1 | 4 | 1 | 4 | 1 | 5 | 1 | 4 | 1 |
| PL-2 | 31 | 9 | 22 | 6 | 4 | 1 | 4 | 1 | 5 | 2 | 4 | 1 | 6 | 2 | 4 | 1 |
| PL-3 | 28 | 8 | 22 | 6 | 7 | 3 | 6 | 3 | 7 | 3 | 6 | 3 | 7 | 3 | 6 | 3 |
| PL-4 | 20 | 4 | 16 | 3 | 9 | 4 | 7 | 4 | 14 | 5 | 8 | 4 | 13 | 5 | 6 | 3 |
| PL-5 | 27 | 7 | 17 | 4 | 13 | 6 | 13 | 5 | 17 | 6 | 15 | 7 | 15 | 6 | 13 | 5 |
| PL-6 | 15 | 3 | 16 | 2 | 30 | 7 | 13 | 5 | 30 | 8 | 13 | 5 | 24 | 7 | 13 | 5 |
| PL-7 | 8 | 2 | 17 | 4 | 33 | 8 | 16 | 7 | 33 | 9 | 17 | 8 | 33 | 9 | 16 | 7 |
| PL-8 | 4 | 1 | 7 | 1 | 33 | 8 | 18 | 8 | 25 | 7 | 14 | 6 | 32 | 8 | 17 | 8 |



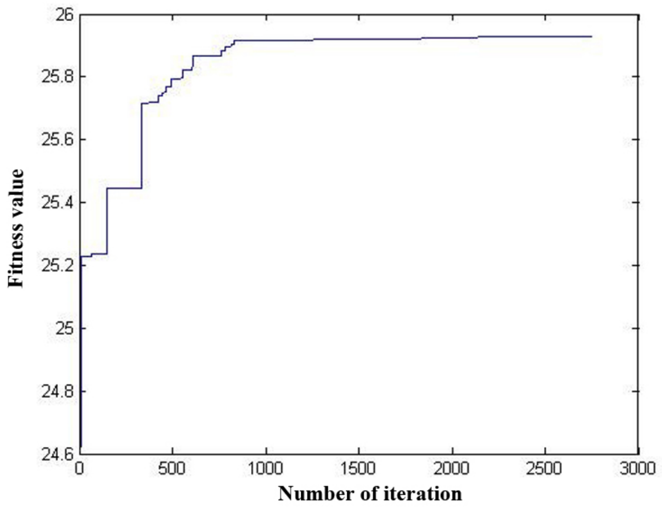
(a)



(b)



(c)



(d)

Figure 2: Convergence curves of PL-1 (HM-10) for Fig. 1(j) using SC1 (a) for 2-level thresholding (b) for 3-level thresholding (c) for 4-level thresholding (d) for 5-level thresholding.

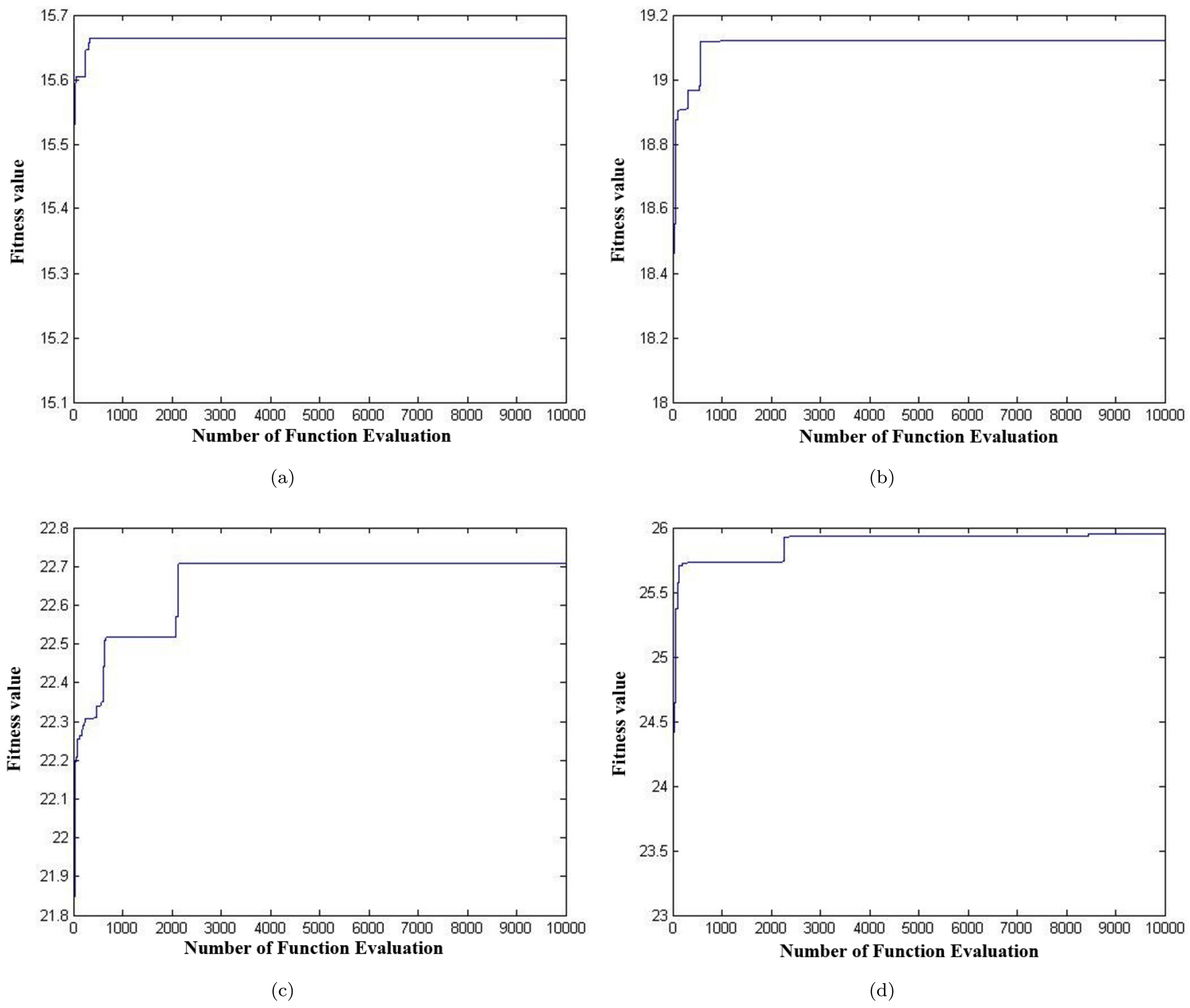


Figure 3: Convergence curves of PL-1 (HM-10) for Fig. 1(j) using SC1 (a) for 2-level thresholding (b) for 3-level thresholding (c) for 4-level thresholding (d) for 5-level thresholding.

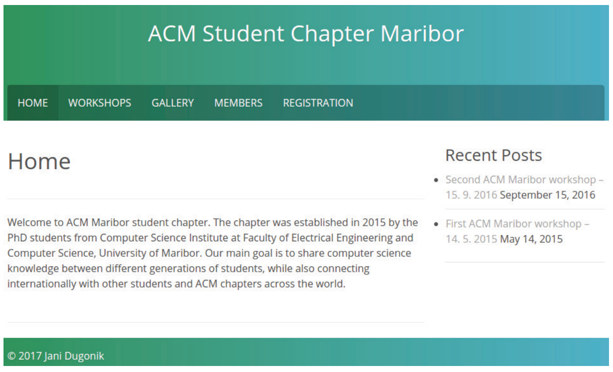
# First three years of the ACM Student Chapter Maribor

Iztok Fister Jr.
University of Maribor
Faculty of Electrical Engineering and Computer Science
Smetanova 17, 2000 Maribor
Slovenia
iztok.fister1@um.si

Niko Lukač
University of Maribor
Faculty of Electrical Engineering and Computer Science
Smetanova 17, 2000 Maribor
Slovenia
niko.lukac@um.si

## Background

The Association for Computing Machinery (ACM) is one of the largest associations for the Computer Science field in the world. In recent years, it has enabled the establishment of local student chapters, at a university or community level, besides already growing professional chapters. The Student Chapter ACM Maribor was officially founded in 2015, as the first such Chapter in Slovenia. The idea was born when Iztok Fister Jr. and Niko Lukač, who were graduate students, discussed the problem that undergraduate and graduate students of Computer Science programs did not collaborate enough. This was even more apparent due to the research fields of various laboratories at the Faculty. Furthermore, there was a lack of student workshops and conferences, which would connect students in Maribor. The possibility of meeting domestic students and students from other Slovenian universities, as well as participation in joint student conferences, was a big reason for the creation of this Chapter. Moreover, the ACM Student Chapter's members gain great benefits, such as complimentary subscription to ACM communications, a free acm.org email address, a full-year electronic subscription for the ACM's student magazine XRDS, and access to quarterly editions of the ACM Student Quick Takes (SQT) newsletter. The Chapter was created when 10 future members met together in order to elect the first Committee and set the goals for the Chapter in the inaugural year. In that meeting, the members discussed activities, where most of the words were dedicated to the first internal workshop that was planned to be organized in May, 2015. During the first meeting, members also discussed other aspects of the Chapter, such as financing, establishment of an internal mailing list, supporting and advertisement of the Chapter through all channels, especially a website (screen shoot is shown in Fig. 1). Table 1 presents the first elected student members in 2015. After a period of one year, new Chapter members were elected (Fig. 2), making the following appointments (Table 2). Future officials and members agreed that one year of being an official is not enough to implement greater ideas in the Chapter. The officials' period has, therefore, been extended to two years (by March, 2018).



Home

Welcome to ACM Maribor student chapter. The chapter was established in 2015 by the PhD students from Computer Science Institute at Faculty of Electrical Engineering and Computer Science, University of Maribor. Our main goal is to share computer science knowledge between different generations of students, while also connecting internationally with other students and ACM chapters across the world.

Recent Posts

- Second ACM Maribor workshop – 15. 9. 2016 September 15, 2016
- First ACM Maribor workshop – 14. 5. 2015 May 14, 2015

© 2017 Jani Dugonik

**Figure 1: Website of ACM Student Chapter Maribor.**

| Position | Person |
|---|---|
| Chair | Iztok Fister Jr. |
| Vice Chair | Niko Lukač |
| Secretary | Denis Kolednik |
| Treasurer | Danijel Žlaus |
| Membership Chair | Štefan Kohek |
| Web Masters | Jani Dugonik, Uroš Mlakar |

**Table 1: First officials of the ACM Student Chapter Maribor (From March, 2015 to March, 2016).**

| Position | Person |
|---|---|
| Chair | Uroš Mlakar |
| Vice Chair | Štefan Kohek |
| Secretary | Denis Kolednik |
| Treasurer | Danijel Žlaus |
| Membership chair | Dušan Fister |
| Web Master | Jani Dugonik |

**Table 2: Second officials of the ACM Student Chapter Maribor (From March, 2016 to March, 2018).**

**Figure 2: Member meeting in 2016 - From left to right: Miha Ravber, David Jesenko, Marko Bizjak, Matej Brumen, Denis Horvat, Robi Cvirn, Uroš Mlakar, Dušan Fister, Niko Lukač, Iztok Fister Jr., Štefan Kohek.**

## Chapter activities

Chapter activities can be grouped into five branches:

- Organization of internal workshops: The aim of these workshops is to organize a platform where members gather together and present their research work to other members or other Computer Science students interested in research. Until now, we have organized two workshops successfully (Fig. 3 presents members at the second workshop) where most of the members presented their research work. Presentations involved discussions, where new ideas were born for improving presented methods, or to provide an opportunity for research-based collaboration. Currently, the third workshop is in preparation and will be held in the second half of this year.

- StuCoSReC organization and promotion: Some members were participating in an annual StuCoSReC conference organization team. Chapter members were involved in the submitted papers' reviewing and editing process, as well as in the event's promotion.

- Recruiting activities: A major task of all members is to look for additional members that could join a Chapter and participate in Chapter activities. This was done in part with collaboration with the Institute of Computer Science, where the Chapter's banners were advertised on the Institutes' website (https://cs.feri.um.si/).

- Social events: The current idea of organizing them monthly does not bear fruit and, therefore, needs plenty of improvement. Especially, agreeing a proper time is one of the problems, since some members are employed in laboratories and occupied with project related research work. In a nutshell, social events should at least encompass some educational trips and invitation of ACM's distinguished speakers to deliver a talk. Currently, these activities are still in the planning process.

- Helping fellow students: In the first meeting, members concluded that all members (especially graduate students) should be able to help younger students. Mostly, this part was very successful, since some student members organized educational workshops and prepared younger students to participate in programming contests, such as ACM-ICPC, the International Collegiate Programming Contest. Some experienced members also offered help for a conference paper preparation, i.e. for StuCoSReC.

**Figure 3: Second ACM Maribor workshop - From left to right: Tilen Škrinjar, Uroš Mlakar, Dušan Fister, Denis Kolednik, Miha Ravber, Robi Cvirn, Niko Lukač, Iztok Fister Jr., Primož Bencak.**

## Conclusion

There are many tasks for the future development of this Student Chapter. The most important work that waits for the members is recruitment related activities and increase of social events, especially from the other Faculties (e.g. Faculty of Economics, Faculty of Natural Sciences and Mathematics). Additionally, in order to organize more social activities, finding sponsors is a big priority.

## Acknowledgements

We are very thankful to the Faculty of Electrical Engineering and Computer Science for their enthusiastically given support and the possibility to host our Chapter activities within the Faculty facilities. Especially, we much appreciate the support of Dean, Prof. Dr. Borut Žalik, who was also a driving force in the first baby steps of the ACM Chapter. We are also grateful to Prof. Dr. Marjan Mernik, the Head of the Institute of Computer Science, for helping with the promotional activities.

# Reševanje problema LABS s pomočjo porazdeljenega računalniškega sistema in spletnega brskalnika

Niko Kovačič
Univerza v Mariboru
Fakulteta za elektrotehniko, računalništvo in informatiko
Koroška cesta 46, SI-2000 Maribor, Slovenija
niko@fsresitve.si

Janez Brest
Univerza v Mariboru
Fakulteta za elektrotehniko, računalništvo in informatiko
Koroška cesta 46, SI-2000 Maribor, Slovenija
janez.brest@um.si

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

## POVZETEK

V članku smo predstavili porazdeljen sistem, ki s pomočjo spletnega brskalnika in programskega jezika JavaScript rešuje problem LABS. Za reševanje problema smo uporabili algoritem *lssOrel*. Ta algoritem je trenutno najboljši za reševanje tega problema. Zaustavitveni pogoj za reševanje določene velikosti problema smo načrtovali glede na najslabše dosežene rezultate med 100 neodvisnimi zagoni. Iz dobljenih rezultatov smo ugotovili, da spletni brskalnik lahko učinkovito uporabimo znotraj porazdeljenega sistema. Program se znotraj tega sistema izvaja nekoliko počasneje kot v primeru programa, ki je implementiran v programskem jeziku C++. Ker pa imamo na voljo veliko število brskalnikov oz. odjemalec, lahko zaženemo več nalog programa hkrati. Če imamo na voljo dovolj odjemalcev, lahko dosežemo tudi boljše rezultate. To smo potrdili v prototipnem sistemu, kjer smo potrdili vse najboljše rešitve problema LABS do velikosti $L = 149$.

## Kjučne besede

Porazdeljeni sistem, problem LABS, spletni brskalnik, JavaScript

## 1. UVOD

V današnjem povezanem svetu so računalniški viri dostopni vsakomur, ki najde preprost način za njihovo pridobivanje. Porazdeljeni sistemi po načelu odjemalec/strežnik večinoma uporabljajo posebne namizne programe, s katerimi si lahko delijo računalniški viri. Tako se lahko ustvari omrežje, ki je sposobno procesirati veliko količino podatkov in preiskovati ogromne iskalne prostore brez uporabe superračunalnikov.

Ključna značilnost takšnega omrežja je preprostost njegove uporabe, saj lahko le tako dosežemo dovolj veliko število uporabnikov in s tem veliko procesorsko moč. Veliko lažje bi bilo, da uporabniki ne bi prenašali nobene programske opreme več, kot je to potrebno pri klasičnih porazdeljenih računalniških sistemih. To predstavlja preveliko pregrado za uporabnika, ki navadno prej obupa in opusti nameščanje potrebne programske opreme. Zato smo raziskali programsko opremo, ki jo ima uporabnik že nameščeno na svojih napravah ter kako bi jo lahko uporabili kot odjemalca.

Za najbolj primernega se je izkazal spletni brskalnik, saj je prednaložen na večini operacijskih sistemov, tudi na mobilnih napravah, kot so pametni telefoni in tablice. Vsi spletni brskalniki podpirajo interpretirani programski jezik JavaScript, s katerim razvijemo spletne aplikacije, ki lahko dostopajo do procesorskih virov. Hitrost izvajanja programov, implementiranih v jeziku JavaScript, se je v zadnjih letih precej izboljšala. To je omogočilo razvoj porazdeljenih sistemov za izvajanje algoritmov v spletnem brskalniku. Tako bomo tudi mi v našem sistemu uporabili jezik JavaScript znotraj spletnega brskalnika za reševanje problema LABS.

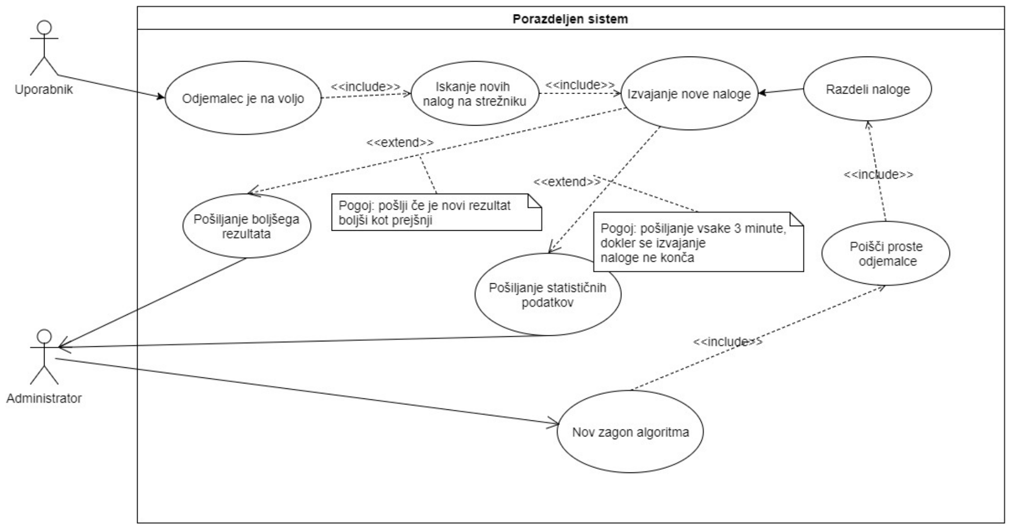
Iskanje binarnih sekvenc z največjim faktorjem F (angl. merit factor), znanega tudi kot *low autocorrelation binary sequence (LABS) problem*, predstavlja velik računski izziv. Za zmanjšanje računskih omejitev problema smo upoštevali algoritme, ki obravnavajo sekvence lihih velikosti ($L$). Posledično nekatere rešitve ne bodo optimalne. Za algoritem smo izbrali *lssOrel* [8], saj je prikazal najboljšo zmogljivost.

Članek je organiziran na naslednji način. V 2. poglavju predstavimo sorodna dela. Nato implementacijo porazdeljenega sistema opišemo v 3. poglavju. Sledijo rezultati v 4. poglavju in članek zaključimo v 5. poglavju.

## 2. SORODNA DELA

Do sedaj je bilo že kar nekaj uspešnih porazdeljenih računalniških sistemov, ki so uspeli privabiti veliko število odjemalcev za reševanje računsko zahtevnih problemov. Eden izmed bolj znanih je SETI@home [6], katerega namen je analizirati radijske signale, v katerih bi lahko bili znaki zunajzemeljske inteligence. Uporablja programsko platformo BOINC, katero gostujejo na kalifornijski univerzi Berkeley.

Med prvimi porazdeljenimi sistemi, ki se uporabljajo v znanstvene namene, je Great Internet Mersenne Prime Search (GIMPS) [3], katerega cilj je iskanje Mersennovih praštevil. GIMPS se opira na Lucas-Lehmerov preizkus praštevil ter uporablja fazo poizkusnega deljenja, ki hitro eliminira


Porazdeljen sistem
Uporabnik
Odjemalec je na voljo
<<include>>
Iskanje novih nalog na strežniku
<<include>>
Izvajanje nove naloge
Razdeli naloge
<<extend>>
<<extend>>
<<include>>
Pošiljanje boljšega rezultata
Pogoj: pošlji če je novi rezultat boljši kot prejšnji
Pogoj: pošiljanje vsake 3 minute, dokler se izvajanje naloge ne konča
Poišči proste odjemalce
Pošiljanje statističnih podatkov
<<include>>
Administrator
Nov zagon algoritma

**Slika 1: Diagram primerov uporabe porazdeljenega sistema.**

Mersennova števila z manjšimi faktorji, ki predstavljajo velik delež kandidatov. S pomočjo sistem GIMPS so do sedaj našli 15 Mersennovih praštevil.

S prihodom kriptovalut je postalo razdeljevanje nagrad za opravljeno računsko delo veliko lažje. Z njihovo pomočjo bo projekt Golem [2] poskušal postati decentraliziran superračunalnik in s tem globalni trg računalniških virov. Naročnik bo s povezovanjem računalnikov v porazdeljen sistem lahko najel vire od drugih uporabnikov, ki se nato uporabljajo za izvedbo nalog. Plačila se izvajajo na transakcijskem sistemu, ki je osnovan na Ethereum ter omogoča neposredne transakcije med uporabniki in naročniki.

V članku [9] so pokazali, da je možno uporabljati spletne brskalnike za izvajanje porazdeljenih algoritmov, kar nam je dalo dodatni zagon pri načrtovanju in implementaciji porazdeljenega sistema, opisanega v tem članku.

## 3. LABS

Problem neperiodičnih binarnih zaporedij z nizkimi avtokorelacijami lahko opišemo z binarno sekvenco dolžine $L$:

$$S = s_1 s_2 ... s_L, s_i \in \{+1, -1\}. \tag{1}$$

Energijo sekvence in faktor $F$ izračunamo z uporabo naslednjih enačb:

$$E(S) = \sum_{k=1}^{L-1} C_k^2(S), \tag{2}$$

$$C_k(S) = \sum_{k=1}^{L-1} s_i s_{i+k}. \tag{3}$$

$$F(S) = L^2/(2E(S)) \tag{4}$$

Cilj reševanja problema je minimizirati energijo oz. maksimirati vrednost faktorja $F$. Iskanje binarnih sekvenc z najboljšim faktorjem $F$ ima pomembno vlogo pri reševanju tehničnih problemov na področju komunikacij.

## 4. APLIKACIJA

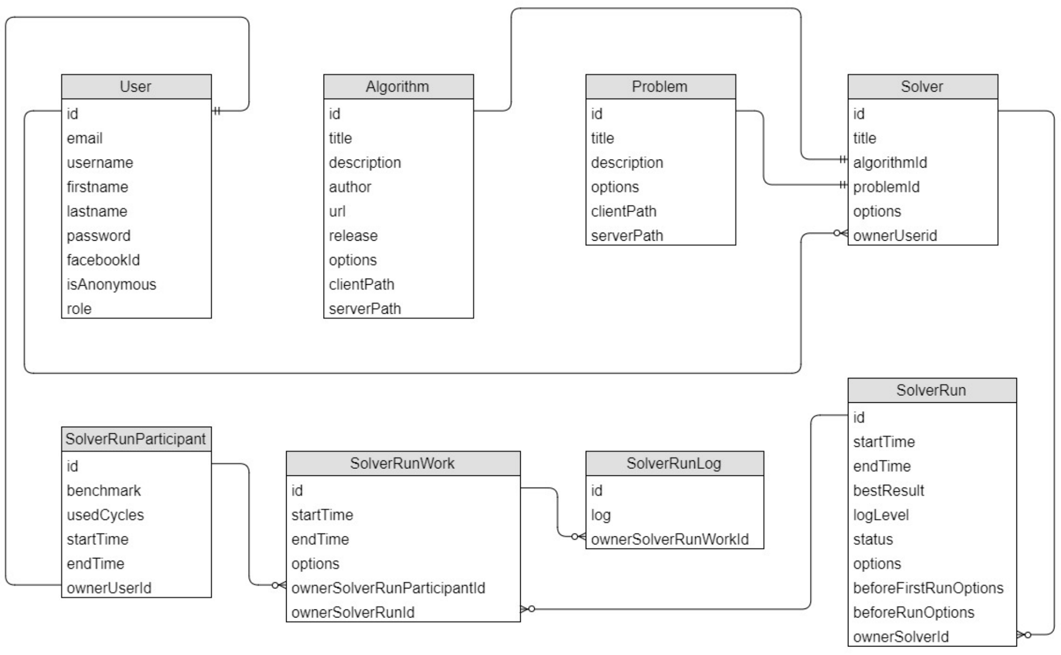
Implementacijo porazdeljenega sistema smo načrtovali tako, da bo podpiral naslednje funkcionalnosti:

- Anonimna uporaba in hitra registracija uporabnika.
- Izvajanje algoritmov v varnem okolju (angl. sandbox) znotraj JavaScript okolja v brskalniku.
- Prenašanje rezultatov iz odjemalca na strežnik.
- Dodeljevanje nalog iz strežnika na odjemalce.
- Prikaz izvajanja algoritma uporabniku na intuitiven način.

Postopek implementacije smo razdelili v več faz:

- Načrtovanje podatkovnega modela.
- Izbira ogrodij za razvoj spletnih aplikacij.
- Implementacija zalednega sistema (angl. backend).
- Implementacija algoritma *lssOrel* v jeziku JavaScript.
- Implementacija uporabniškega vmesnika.
- Implementacija nadzorne plošče.

Porazdeljen računalniški sistem začne nov zagon algoritma po ukazu administratorja na nadzorni plošči. Vsak nov odjemalec pošlje povpraševanje na strežnik, le ta pa mu posreduje naloge ali pa ga postavi v stanje pripravljenosti, kjer čaka na delo. Vsak odjemalec posreduje strežniku informacije o napredku reševanja problema. To posredovanje se izvaja vsakih 5 minut ali ko se najde novo najboljša rešitev. Na tak način se izognemo prekomerni komunikaciji med odjemalcem in strežnikom. Ker uporabljamo stohastične algoritme, se vsakemu odjemalcu dodeli unikatno seme. Seme uporablja odjemalec, da inicializira generator naključnih števil. S tem se izognemo, da bi odjemalci reševali problem na enak način.



**Slika 2: Podatkovni model porazdeljenega računalniškega sistema.**

### 4.1 Podatkovni model

Osnova podatkovnega modela 1 sta entiteti Algorithm in Problem, kjer so shranjene njune posamezne informacije. V entiteti Solver nato definiramo, kateri algoritem bo reševal posamezni problem. Vsak novi zagon reševanja problema shranimo v entiteto SolverRun. Nova naloga, ki se pridobi od strežnika v okviru zagona algoritma, je shranjena v entiteti SolverRunWork in ima začetni ter končni čas izvajanja naloge. Lastnik vsake naloge je SolverRunParticipant, ki je končni uporabnik porazdeljenega sistema in lahko naenkrat rešuje več nalog.

### 4.2 Implementacija

Ogrodja za razvoj aplikacij niso nujno potrebna, nam pa zagotavljajo, da bo razvoj primerno strukturiran in bo potekal v skladu s poslovnimi pravili ter omogočal kasnejše nadgradnje in vzdrževanje. S ponovno uporabo generičnih modulov ogrodja, razvijalci prihranijo veliko časa in se lahko osredotočijo na druga področja. Za zaledni sistem smo izbrali ogrodje Sails.js [5], ki je zgrajeno na osnovi Node.js [4] strežniške tehnologije. Izbrali smo ga zaradi modularnosti, preproste uporabe in podpori realnočasovnih povezav z odjemalcem. Za izdelavo uporabniškega vmesnika smo uporabili ogrodje AngularJS [1], ki nas je prepričalo s sinhronizacijo podatkov med modelom in pogledom. Ko se podatki v modelu spremenijo, se to odraža v pogledu ter obratno. To se zgodi takoj in samodejno, kar zagotavlja, da sta model in pogled vedno posodobljena.

Za realno časovno povezavo skrbi komunikacijski protokol WebSocket, ki ustvari dvosmerni kanal za komuniciranje med odjemalcem in strežnikom. Preko povezave WebSocket je velikost posameznega poslanega paketa precej manjša, saj nima nobenega dodatnega balasta, ki je potreben pri navadnih zahtevah HTTP. Čas pošiljanja paketa na strežnik je precej krajši, saj ni potrebno vedno znova pošiljati zahtev, ker se vse pošlje po obstoječi povezavi. Odpade tudi periodično preverjanje po novih nalogah, saj to sporoči strežnik odjemalcu neposredno. Ogrodje Sails.js poskrbi, da se zahteve, poslane preko protokola HTTP, vežejo na enako akcijo kot WebSocket zahteve, kar je precej pospešilo razvoj, saj odpravi podvojenost programske kode.

Na odjemalcu se posamezna naloga izvaja na svoji niti, s katero lahko komuniciramo preko dogodkovnega vodila (angl. event bus). Z izvajanjem nalog v ozadju se izognemo obremenjevanju glavne niti, kar lahko povzroči, da se uporabniški vmesnik preneha odzivati. Vsako ustvarjeno nit po koncu izvajanja naloge prekinemo in tako sprostimo računalniške vire ter odjemalca pripravimo na nove naloge.

## 5. REZULTATI

Testiranje porazdeljenega sistema je potekalo v nadzornem okolju, kjer so sodelovali do trije odjemalci. Zaključni pogoj izvajanja algoritma *lssOrel* za določeno velikost problema smo definirali kot:

$$\text{maksimalno število ovrednotenj} = 2743,37 * 1,1456^{L} \quad (5)$$

Enačbo smo pridobili iz testnih podatkov v [8] in je ocena, koliko ovrednotenj je potrebno, da dobimo optimalno rešitev ali njen približek. Enačbo smo potrdili kot pravilno, saj smo v vseh zagonih algoritma dobili najboljše znane rešitve glede na [8] za velikosti problema $L \leq 149$ (glej tabelo 4.2).

## 6. ZAKLJUČEK

**Tabela 1: Doseženi rezultati s pomočjo porazdeljenega sistema. Uporabljene so naslednje oznake: $E$ - energija, $F$ - faktor (angl. merit factor), $t$ - čas v sekundah.**

| $L$ | $E$ | $F$ | število vrednotenj na sekundo | $t$ | največje število vrednotenj |
|---|---|---|---|---|---|
| 7 | 3 | 8,1667 | 2,1078e+5 | 5,4000e+1 | 7,10420e+3 |
| 15 | 15 | 7,5000 | 1,7218e+6 | 3,6000e+1 | 2,10756e+4 |
| 23 | 51 | 5,1863 | 3,2107e+6 | 3,2000e+1 | 6,25235e+4 |
| 31 | 79 | 6,0823 | 7,6905e+6 | 3,2000e+1 | 1,85484e+5 |
| 39 | 99 | 7,6818 | 7,0281e+6 | 3,2000e+1 | 5,50265e+5 |
| 47 | 135 | 8,1815 | 7,3849e+6 | 3,2000e+1 | 1,63243e+6 |
| 55 | 171 | 8,8450 | 6,7895e+6 | 3,2000e+1 | 4,84283e+6 |
| 63 | 271 | 7,3229 | 5,9672e+6 | 3,5000e+1 | 1,43669e+7 |
| 71 | 275 | 9,1655 | 5,5435e+6 | 3,7000e+1 | 4,26214e+7 |
| 79 | 407 | 7,6671 | 5,2777e+6 | 3,8000e+1 | 1,26442e+8 |
| 87 | 451 | 8,3914 | 5,4171e+6 | 7,5000e+1 | 3,75108e+8 |
| 95 | 479 | 9,4207 | 9,6036e+6 | 1,1900e+2 | 1,11281e+9 |
| 97 | 536 | 8,7771 | 8,8999e+6 | 1,8300e+2 | 1,46045e+9 |
| 99 | 577 | 8,4931 | 8,8528e+6 | 2,4000e+2 | 1,91669e+9 |
| 101 | 578 | 8,8244 | 9,4152e+6 | 2,8000e+2 | 2,51546e+9 |
| 103 | 555 | 9,5577 | 9,3575e+6 | 3,8200e+2 | 3,30129e+9 |
| 105 | 620 | 8,8911 | 7,8757e+6 | 6,3300e+2 | 4,33261e+9 |
| 107 | 677 | 8,4557 | 9,1100e+6 | 7,9700e+2 | 5,68612e+9 |
| 109 | 662 | 8,9736 | 9,3342e+6 | 8,0700e+2 | 7,46246e+9 |
| 111 | 687 | 8,9672 | 9,2006e+6 | 1,2170e+3 | 9,79373e+9 |
| 113 | 752 | 8,4900 | 8,0691e+6 | 1,7380e+3 | 1,28533e+10 |
| 115 | 745 | 8,8758 | 7,6744e+6 | 2,2370e+3 | 1,68686e+10 |
| 117 | 786 | 8,7080 | 9,1235e+6 | 2,9990e+3 | 2,21384e+10 |
| 119 | 835 | 8,4796 | 7,1910e+6 | 4,0850e+3 | 2,90544e+10 |
| 121 | 844 | 8,6736 | 9,8296e+6 | 3,9270e+3 | 3,81310e+10 |
| 123 | 893 | 8,4709 | 5,9495e+6 | 1,1004e+4 | 5,00431e+10 |
| 125 | 846 | 9,2346 | 8,0048e+6 | 8,6710e+3 | 6,56765e+10 |
| 127 | 887 | 9,0919 | 4,8515e+6 | 1,8681e+4 | 8,61938e+10 |
| 129 | 920 | 9,0440 | 5,4651e+6 | 2,4353e+4 | 1,13121e+11 |
| 131 | 913 | 9,3981 | 6,7557e+6 | 2,3223e+4 | 1,48460e+11 |
| 133 | 1010 | 8,7569 | 5,0255e+6 | 4,6720e+4 | 1,94838e+11 |
| 135 | 1027 | 8,8729 | 8,4775e+6 | 3,1901e+4 | 2,55705e+11 |
| 137 | 1052 | 8,9206 | 6,0791e+6 | 5,7210e+4 | 3,35588e+11 |
| 139 | 1133 | 8,5265 | 1,4778e+7 | 3,0284e+4 | 4,40425e+11 |
| 141 | 1126 | 8,8282 | 9,2139e+6 | 8,5836e+4 | 5,78014e+11 |
| 143 | 1191 | 8,5848 | 4,4103e+6 | 2,0027e+5 | 7,58585e+11 |
| 145 | 1208 | 8,7024 | 4,3899e+6 | 2,2762e+5 | 9,95566e+11 |
| 147 | 1265 | 8,5411 | 2,1964e+6 | 7,0083e+5 | 1,30658e+12 |
| 149 | 1218 | 9,1137 | 4,0066e+6 | 5,2562e+5 | 1,71476e+12 |

V članku smo preverili, ali je možno uporabiti spletni brskalnik kot odjemalec v porazdeljenem računalniškem sistemu. V njem smo pognali več zagonov algoritma *lssOrel*, ki rešuje problem LABS. Uspeli smo potrditi vse najboljše znane rešitve, ki so bile najdene v referenčnem članku [8]. Spletni brskalnik se je izkazal kot primerno nadomestilo namiznim programom, saj je preprostejši za uporabo, ima velik nabor obstoječih uporabnikov in je po naših meritvah samo za faktor 2 do 4 počasnejši od programa, ki je implementiran v programskem jeziku C++. V prihodnosti se bo ta razlika zmanjšala ali morda celo izničila s prihodom novih spletnih tehnologij [7].

## Zahvala

Janez Brest in Borko Bošković priznavata financiranje prispevka s strani Javne agencije za raziskovalno dejavnost Republike Slovenije, raziskovalni program P2-0041 – Računalniški sistemi, metodologije in inteligentne storitve.

## LITERATURA

[1] AngularJS . `https://angularjs.org/`, 2017. [Na spletu; dostopano 14-08-2017].

[2] Golem worldwide supercomputer . `https://golem.network/`, 2017. [Na spletu; dostopano 14-08-2017].

[3] Great Internet Mersenne Prime Search . `https://www.mersenne.org/various/works.php/`, 2017. [Na spletu; dostopano 14-08-2017].

[4] Node.js . `https://nodejs.org/en/about/`, 2017. [Na spletu; dostopano 14-08-2017].

[5] Sails.js . `http://sailsjs.com//`, 2017. [Na spletu; dostopano 14-08-2017].

[6] SETI@home . `https://setiathome.berkeley.edu/sah_about.php/`, 2017. [Na spletu; dostopano 14-08-2017].

[7] WebAssembly . `http://webassembly.org/`, 2017. [Na spletu; dostopano 14-08-2017].

[8] B. Bosković, F. Brglez, and J. Brest. Low-autocorrelation binary sequences: On improved merit factors and runtime predictions to achieve them. *Applied Soft Computing*, 56(2):262–285, 2017.

[9] J. L. A. M. A. P. J. Merelo, P. Castillo. Asynchronous distributed genetic algorithms with javascript and json. In *WCCI 2008 Proceedings*, page 1372–1379. IEEE Press, March 2008.

# Data mining big data inpatient database using Cuckoo search

Uroš Mlakar
Faculty of Electrical Engineering and Computer Science, University of Maribor
Smetanova 17, 2000 Maribor
uros.mlakar@um.si

Iztok Fister Jr.
Faculty of Electrical Engineering and Computer Science, University of Maribor
Smetanova 17, 2000 Maribor
iztok.fister1@um.si

Monika Marković
Faculty of Medicine, University of Maribor
Taborska 6b, 2000 Maribor
monika.markovic@student.um.si

Iztok Fister
Faculty of Electrical Engineering and Computer Science, University of Maribor
Smetanova 17, 2000 Maribor
iztok.fister@um.si

## ABSTRACT
This paper investigates data mining in a medical dataset by using the stochastic population-based nature-inspired Cuckoo search algorithm. Particularly, association rules are mined by applying an objective function composed of support and confidence weighted by two parameters for controlling the importance of each measure. The rules are mined in a Nationwide Inpatient Sample dataset, which is a collection of discharge records of several hospitals in the USA. Only those records, where a patient was diagnosed with Type II diabetes mellitus were extracted for association rule mining. The results show that the found rules are simple, easy to understand and also interesting, as they were verified with actual clinical studies. The results obtained can be beneficial to either doctors or insurance companies.

## Keywords
data mining, big data, association rule mining, cuckoo search

## 1. INTRODUCTION
With the increasing rate of data collected everyday, there is a need for automatic mining of useful information hidden within. But this may be a difficult task, since this data is either big in volume, has variety (different data sources or multiple data types), or is collected at a very fast pace (velocity). An example of such data are definitely the discharge records of hospital patients. There is a lot of hidden information within this data, such as interesting connections between apparently unrelated diseasepresenteds, or discovering interesting risk factors, that contribute to a particular disease (although not being directly related to the disease). Such knowledge would be beneficial to hospitals, and also to insurance compaines, which can make evidence based decisions, and can optimize, validate and refine the rules that govern their business [6]. This important hidden knowledge can be found with the help of data mining, with methods such as clustering, feature selection, association rule mining, and many more.

This paper is structured as follows. After the introduction, data mining methods are briefly discussed in Section 2, then the Cuckoo search algorithm and the Nationwide Inpatient Sample (NIS) dataset are presented in Sections 3 and 4. The preliminary results are presented in form of association rules in Section 5, then the paper is concluded with future directions in Section 6.

## 2. DATA MINING METHODS
Data mining is a computing process of discovering patterns in large datasets. The goal of data mining is to extract useful information from a dataset and transform it into an understandable structure, which may be used directly or processed further by another algorithm. There are several methods for which are used for data mining, such as cluster analysis [4], dimensionality reduction [8], association rule mining [9], etc. Association rule mining has gained a lot of attention for mining interesting patterns from large databases within the research community.

### 2.1 Association rule mining
Association rule mining (ARM) is a rule-based machine learning method for discovering interesting relations between attributes in large databases. ARM is used for identifying strong rules using measures of interestingness, where the most established method is the Apriori algorithm introduced by Agrawal et al. [1]. ARM can be mathematically expressed as follows. Let $I = \{i_i, i_2, \ldots, i_n\}$ be a set of attributes called items and $T = \{t_1, t_2, \ldots, t_m\}$ the a set of transactions (i.e. database). Each rule is defined as an implication $X \rightarrow Y$, where $X, Y \subseteq I$. $X$ and $Y$ are composed of two different set of items, which are also known as item-sets; $X$

is also defined as the antecedent, while $Y$ is called the consequent. To be able to select interesting rules from the set of all possible rules, various measures of interestingness are utilised and also constrained. The most used constraints are the minimum support and confidence, which are defined as follows:

$$Support(X \rightarrow Y) = \frac{|t \in T; X \subseteq t \wedge Y \subseteq t|}{|T|} \quad (1)$$

$$Confidence = \frac{Support(X \cup Y)}{Support(X)} \quad (2)$$

Support is defined as a proportion of transactions containing $X$ and $Y$, and the total number of transactions, while confidence is a proportion of transactions which contain $X$, and also contain $Y$.

Although being able to find interesting rules on smaller datasets, it faces computational problems when confronted with bigger datasets. To overcome this problem the research has gone in the direction of stochastic population-based nature-inspired algorithms, that treat the ARM as an optimization problem.

## 3. CUCKOO SEARCH

Cuckoo search (CS) is a stochastic population-based nature-inspired optimization algorithm proposed by Yang and Deb in 2009 [11]. It is classified as a Swarm Intelligence (SI) algorithm, since its mechanisms are inspired by the natural behaviour of some cuckoo species in nature. To be able to capture the behaviour of cuckoos and adapt it to be suitable for using as a computer optimization algorithm, the authors idealised three rules:

- A cuckoo lays only one egg, then dumps it into a randomly chosen nest,
- Nests that contain high-quality eggs, are carried over to the next generation,
- Any cuckoo egg may be discovered by the host bird with probability $p_a \in [0, 1]$. If an egg is discovered, the host bird may abandon the nest, and build a new one at a new location.

Each solution in population of the CS algorithm corresponds to a cuckoo nest, which represents the position of the egg within the search space, and can be mathematically expressed as follows:

$$\mathbf{x}_i^{(g)} = \{x_{i,j}^{(g)}\}, \quad \text{for} i = 1, \ldots, Np \quad \text{and} j = 1, \ldots, D, \quad (3)$$

where $NP$ is the population size, and $D$ the dimension of the optimization problem. In the CS algorithm, new solutions are created by exploitation of the current solutions as:

$$\mathbf{x}_i^{(g+1)} = \mathbf{x}_i^{(g)} + \alpha L(s, \lambda), \quad (4)$$

where

$$L(s, \lambda) = \frac{\lambda \Gamma(\lambda) \sin(\frac{\pi\lambda}{2})}{\pi} \frac{1}{s^{(1+\lambda)}}. \quad (5)$$

The term $L(s, \lambda)$ determines the characteristic scale and $\alpha > 0$ is the scaling factor of the step size $s$.

Table 1: Data elements considered from the NIS dataset.

| Element Name | Element description |
|---|---|
| Age | Age of patient at admission (years) |
| Atype | Admission type |
| Died | Died during hospitalization |
| Female | Indicator of sex |
| Los | Length of stay |
| DX1 | Principal diagnosis |
| DX$\{2-15\}$ | Diagnoses$\{2-15\}$ |
| PR1 | Principal Procedure |
| PR$\{2-15\}$ | Procedures$\{2-15\}$ |

### 3.1 Association rule mining using CS algorithm

Since the CS is used for ARM in this paper, the solution representation has to be adapted accordingly. There are two well established encodings available for representing rules for evolutionary algorithms (EA) and SI-based algorithms. The first is the Michigan encoding [5], where each solution represents a separate association rule. In the second, the Pittsburgh encoding [5], each solution represents a set of association rules. for the purpose of this study the Michigan encoding was used. Additionally a fitness evaluation function needs to be defined in order to find the most promising rules:

$$f(\mathbf{x}_i^{(g)}) = \frac{\beta Support(X \rightarrow Y) + \gamma Confidence(X \rightarrow Y)}{\beta + \gamma} \quad (6)$$

The fitness function $f(x)$ is defined as a weighted sum of support and confidence. The weights $\beta$ and $\gamma$ control the importance of both said measures. The user can set the values of these weights according to the importance of each measure in the domain of association rule mining. For the purpose of this study, the values of $\beta = \gamma = 1$.

## 4. NATIONWIDE INPATIENT SAMPLE DATASET

The Nationwide Inpatient Sample (NIS) dataset holds the records of hospital inpatient discharges, that date back to 1988, and is used for identifying, tracking and analysing trends in health care access, quality, and outcomes. It is a publicly available dataset, without any patient identifiers. It is worth noticing that it consists merely of US hospital discharges. It holds about 64 million records, with 126 clinical and non-clinical data elements. Only the elements listed in Table 1 were used in this study.

The DX1 and DX$\{2-15\}$ are the principal diagnosis and other diagnoses, respectively. The diagnoses are represented as codes by following the International Classification of Diseases, Ninth Revision, Clinical Modification (ICD-9-CM). Since the NIS dataset holds a lot of records, it is hard to find association rules by considering the whole dataset. The goal of this research is to uncover other risk factors for patients, who suffer from one particular disease. For this reason, we chose a disease with code '250.30', which is Type II diabetes mellitus (TIIDM), which is a heterogeneous group of disorders characterized by a variable degree of insulin resistance, impaired insulin secretion, and increased glucose production. There are many causes of which doctors and patients should be aware of and maybe even more of those

Table 2: Association rules as found by the Cuckoo search rule miner.

| Antecedent | Consequent | Fitness value |
|---|---|---|
| $(DX = 201.23) \wedge (SEX = FEMALE)$ | $(LOS =< 44)$ | 0.508 |
| $(DX = 201.23) \wedge (SEX = FEMALE) \wedge (TYPE = URGENT)$ | $(LOS =< 44)$ | 0.503 |
| $(DX = 142.8) \wedge (AGE = 40 - 49)$ | $(LOS =< 44)$ | 0.501 |
| $(DX = 202.88)$ | $(DIED = NO)$ | 0.501 |
| $(DX = 070.22)$ | $(DIED = NO)$ | 0.501 |
| $(DX = 016.04) \wedge (AGE = 60 - 69)$ | $(TYPE = EMERGENCY)$ | 0.5 |
| $(DX = 036.3) \wedge (DIED = NO)$ | $(SEX = MALE)$ | 0.5 |
| $(DX = 197.8)$ | $(DIED = NO)$ | 0.5 |
| $(DX = 255.8) \wedge (DIED = NO)$ | $(TYPE = EMERGENCY)$ | 0.5 |
| $(DX = 206.00) \wedge (AGE = 80 - 89) \wedge (SEX = FEMALE)$ | $(TYPE = EMERGENCY)$ | 0.5 |
| $(DX = 010.04)$ | $(SEX = FEMALE)$ | 0.5 |
| $(DX = 012.33)$ | $(LOS =< 44)$ | 0.5 |
| $(DX = 201.66)$ | $(DIED = NO) \wedge (LOS =< 44)$ | 0.5 |
| $(DX = 079.88) \wedge (SEX = MALE) \wedge (DIED = NO)$ | $(TYPE = EMERGENCY)$ | 0.5 |
| $(DX = 211.7)$ | $(SEX = FEMALE)$ | 0.5 |
| $(DX = 010.03) \wedge (SEX = FEMALE) \wedge (DIED = YES)$ | $(DX = 015.55)$ | 0.5 |
| $(DX = 201.66) \wedge (SEX = MALE) \wedge (DIED = NO)$ | $(LOS =< 44)$ | 0.5 |
| $(DX = 171.4)$ | $(SEX = FEMALE)$ | 0.5 |
| $(DX = 085.5) \wedge (DIED = YES)$ | $(LOS =< 44)$ | 0.5 |
| $(DX = 232.8)$ | $(DIED = NO)$ | 0.5 |

that we might not know [3]. The latter is also the reason for this study, and with this in mind, all records containing the disease with ICD-9-CM code '250.30' (Type II diabetes mellitus) were extracted from the whole NIS dataset to form a new smaller dataset.

## 5. RESULTS

In this section the results of association rule mining on the NIS dataset using the CS algorithm is presented. The results are reported in Table 2 in form of association rules. Additionally the fitness value of each rule is reported. Only the best 20 rules are reported in Table 2, but only the top five are additionally commented on. The average number of antecedent obtained in this study is 1.8, while the average number of consequent is 1.05. This is favourable for the user, since shorter association rules are easier to understand. It also worth emphasizing that all rules produces are somehow related to the TIIDM. The first two rules indicate that the TIIDM is involved with the pathogenesis of non-Hodgkin's sarcoma. This fact is supported by several studies in literature [2]. The third and fourth rules state that there is a connection of Maligant neoplasm of major salivary gland and other maligant lymphomas with TIIDM, which is supported by a study in [10] where an increased prevalence of diabetes was found in patients with salivary gland tumour. A chronic viral hepatitis B was found to be in connection with TIIDM in the fifth rule [7].

## 6. CONCLUSION

The CS algorithm was investigated as a association rule miner on a hospital discharge dataset. The CS produces rules, which are simple, easy to understand, and also interesting. The rules are found with the help of a objective function, which weighs the support and confidence of the rules. The weights control how each interestingness measure is important, and thus guides the search in the desired direction.

The obtained rules were compared with research in the field of Type II diabetes mellitus, where all results were confirmed to be reasonable and supported by a study.

In future we would like to use the CS algorithm for mining association rules for other diseases which occur commonly in the modern world.

## 7. REFERENCES

[1] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. In *Acm sigmod record*, volume 22, pages 207–216. ACM, 1993.

[2] Chun Chao and John H. Page. Type 2 diabetes mellitus and risk of non-hodgkin lymphoma: A systematic review and meta-analysis. *American Journal of Epidemiology*, 168(5):471–480, 2008.

[3] Anthony S Fauci et al. *Harrison's principles of internal medicine*, volume 2. McGraw-Hill, Medical Publishing Division New York, 2008.

[4] Iztok jr. Fister. *Algoritmi računske inteligence za razvoj umetnega športnega trenerja*. PhD dissertation, University of Maribor, Faculty of Electrical Engineering and Computer Science, 2017.

[5] John H. Holland. Adaptation*. In Robert Rosen and Fred M. Snell, editors, *Progress in Theoretical Biology*, pages 263–293. Academic Press, 1976.

[6] Mohammed Khalilia, Sounak Chakraborty, and Mihail Popescu. Predicting disease risks from highly imbalanced data using random forest. *BMC medical informatics and decision making*, 11(1):51, 2011.

[7] Kar Neng Lai, Fernand Mac-Moune Lai, Nancy WY Leung, Stephen T Lo, and John S Tam. Hepatitis with isolated serum antibody to hepatitis b core antigen: a variant of non-a, non-b hepatitis? *American journal of clinical pathology*, 93(1):79–83, 1990.

[8] Uroš Mlakar, Iztok Fister, Janez Brest, and Božidar Potočnik. Multi-objective differential evolution for feature selection in facial expression recognition systems. *Expert Systems with Applications*, 89:129–137, 2017.

[9] Uroš Mlakar, Milan Zorman, Iztok Fister Jr, and Iztok Fister. Modified binary cuckoo search for association rule mining. *Journal of Intelligent & Fuzzy Systems*, 32(6):4319–4330, 2017.

[10] Zsuzsanna Suba, József Barabás, György Szabó, Daniel Takács, and Márta Ujpál. Increased prevalence

of diabetes and obesity in patients with salivary gland tumors. *Diabetes Care*, 28(1):228–228, 2004.

[11] Xin-She Yang and Suash Deb. Cuckoo search via lévy flights. In *Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*, pages 210–214. IEEE, 2009.

# Reševanje problema inverzne kinematike s pomočjo algoritma diferencialne evolucije

Timi Kupčič
Univerze v Mariboru
Fakulteta za elektrotehniko, računalništvo in informatiko
Koroška cesta 46
2000 Maribor
timi.kupcic@gmail.com

Janez Brest
Univerze v Mariboru
Fakulteta za elektrotehniko, računalništvo in informatiko
Koroška cesta 46
2000 Maribor
janez.brest@um.si

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

## POVZETEK

V članku predstavimo reševanje problema inverzne kinematike s pomočjo algoritma diferencialne evolucije. Algoritem preizkusimo na primeru robotske roke PUMA 560. Dobljeni rezultati so pokazali, da je algoritem relativno hiter in uspešen pri reševanju problema inverzne kinematike. Z velikostjo populacije 300 je algoritem dosegel 100 % uspešnost za 10 naključno izbranih točk.

## Kjučne besede

diferencialna evolucija, inverzna kinematika, robotska roka

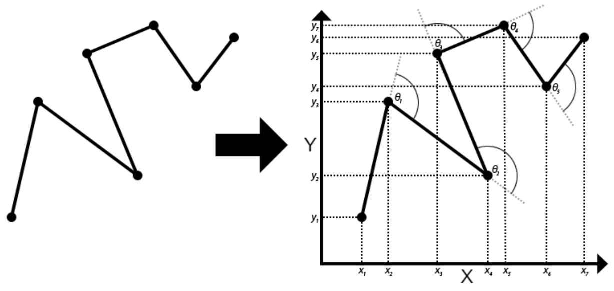
## 1. UVOD

Inverzna kinematika predstavlja problem določanja parametrov sklepov roke, da le-to spravimo v željen položaj. Ta problem zasledimo na različnih področjih. Nekatera izmed teh področij obsegajo zabavno industrijo, kjer se je poslužujejo liki v računalniških igrah/animacijah, pa vse do robotskih rok, ki bodisi sodelujejo pri zapletenih operacijah v medicini ali pa zgolj opravljajo dela za tekočim trakom. Slednje prihaja vedno bolj v uporabo. S pomočjo podobnih rešitev, kot je naša, lahko roboti opravljajo vedno bolj zapletena in natančna opravila.

V članku bomo predstavili rešitev določanja parametrov robotske roke oz. položaja končnega člena s pomočjo algoritma diferencialne evolucije. V naslednjem poglavju bomo predstavili problem inverzne kinematike. V tretjem poglavju bomo podali opis algoritma diferencialne evolucije. Opis eksperimenta in dobljenih rezultatov podajamo v četrtem poglavju. Na koncu podajamo še kratek zaključek.

## 2. PROBLEM INVERZNE KINEMATIKE

Zamislimo si poljubno verigo, sestavljeno iz $n$ točk in $n-1$ poljubno dolgih povezav, ki te točke povezujejo. Ta veriga se



**Slika 1: Preslikava verige v 2D prostor**

prične s točko $O$ (”*origin*”) in konča s točko $E$ (”*endpoint*”). Zaradi lažje predstavitve preselimo verigo v 2D kartezični prostor (slika 1), v katerem bomo predstavili nekatere lastnosti, ki pa veljajo tudi za 3D prostor.

V 2D kartezičnem prostoru se tako prva točka $O(x_0, y_0)$, zmeraj nahaja na znanih in nespremenljivih koordinatah. Ponavadi je to kar koordinatno izhodišče. Koordinate vsake naslednje točke $(x_i, y_i)$ pa lahko izračunamo iz koordinat prejšnje točke in naslednjih dveh parametrov:

- dolžine povezave - $d_i$, med sosednjima točkama in
$$d_i = d((x_i, y_i), (x_{i+1}, y_{i+1})); \quad 0 \leq i < n$$
- kotom - $\theta_i$, ki ga ta povezava določa.
$$\theta_i = \theta((x_i, y_i), (x_{i+1}, y_{i+1})); \quad 0 \leq i < n$$

V nadaljevanju bomo celotno verigo zaradi lažjega prehoda na rešitev imenovali *roka*, vsaki točki pa bomo rekli *sklep* - $S$. Posamezen sklep $\mathbf{S}_i$ bo tako definiran kot vektor komponent:

$$\mathbf{S}_i = \{d_i, \theta_i\}; \quad 0 \leq i < n-1$$

Vidimo, da zgornja enačba ne vsebuje končnega sklepa $E$. Razlog je v tem, da slednji nima naslednika, zatorej ni kota in razdalje, za katera bi bil naslednji sklep bodisi obrnjen ali oddaljen od $E$.

Predstavimo še dva navidez podobna, vendar povsem različna problema. Vsak izmed njiju namreč rešuje svojo specifično nalogo, glede na znane podatke.

DOI: https://doi.org/10.26493/978-961-7023-40-4.23-26

- Direktna kinematika ("*Forward Kinematics*") - FK: opisuje matematični proces računanja koordinat končnega sklepa $E$ iz parametrov vseh predhodnih sklepov [?].
- Inverzna kinematika ("*Inverse Kinematics*") - IK: opisuje matematični proces določanja parametrov vseh sklepov, da se bo končni sklep $E$ nahajal na željenih koordinatah [?].

Reševanje FK je trivialno in obsega zgolj geometrično računanje, medtem ko IK povzroča veliko večje probleme. Rešitev problema IK se namreč nahaja v kompleksnih, tesno povezanih in skrajno nelinearnih matematičnih enačbah, ki vračajo več različnih rešitev [?, ?]. Zato ne preseneča, da je tema najrazličnejših metod in raziskav. Za mnoge probleme IK obstajajo ti. analitične rešitve, ki so na voljo v zaprti obliki. Parametri, ki jih vračajo te rešitve, so točni in se izračunajo v zgolj nekaj mikrosekundah [?]. Vendar analitične rešitve bodisi še niso odkrite za vse probleme ali pa jih sploh ni mogoče pridobiti. V teh primerih se uporabijo ti. računske metode, katere pa vračajo zgolj približno rešitev in so ponavadi računsko, in posledično tudi časovno dokaj zahtevne. Nekatere izmed teh metod so [?]: *Jacobian inversion method, Jacobian construction*, iterativna metoda, optimizacijska metoda, *Cyclic coordinate descent*, *Jacobian transpose method*, genetsko programiranje, itd.

## 3. DIFERENCIALNA EVOLUCIJA

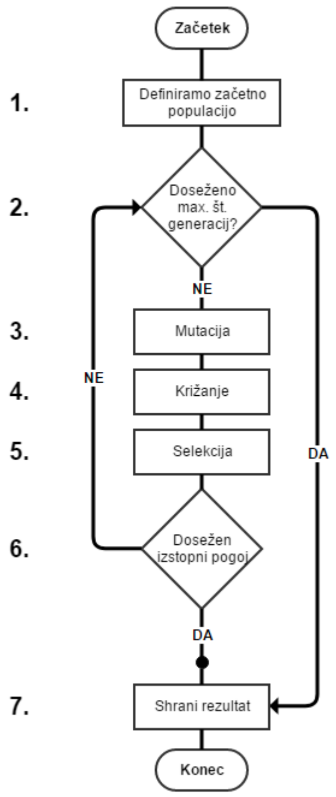
Algoritem diferencialne evolucije (DE) predstavlja metodo za globalno optimizacijo danega problema [?, ?]. Leta 1996 ga je predstavil K. Price kot eno izmed možnih metod za reševanje polinomov Čebišova. Še istega leta je algoritem dosegel zavidljive rezultate na prvem tekmovanju evolucijskih algoritmov v Nagoyi. Zaradi učinkovitosti in preprostosti je bil uporabljen za reševanje različnih probemov kot so npr. uglaševanje šahovske ocenitvene funkcije [?], optimizacijacija zvijanja proteinov [?], multimodalno optimizacijo [?] itd. Nekatere prednosti DE so, da je izjemno stabilen pri problemih, ki vključujejo nekonveksne, multimodalne in nelinearne funkcije. Torej je nalašč primeren za naš problem, kjer rešujemo IK.

Način delovanja diferencialne evolucije prikazuje slika 2 in vsebuje naslednje mehanizme:

1. Definiranje začetne populacije - naključno definiramo $Np$ število posameznikov - $\mathbf{X}_i^G$ (imenovanih tudi *starševski vektorji*), kjer vsak posameznik predstavlja morebitno rešitev obravnavanega problema:

   $\mathbf{X}_i^G = \{x_0, x_1, \dots, x_j\};\quad 0 \le i < Np, 0 \le j < D, G = 0,$

   - $x_j$ - $j$-ta neznanka v problemu
   - $Np$ - število posameznikov znotraj populacije
   - $D$ - dimenzija problema
   - $G$ - generacija
2. Sprehod skozi vse generacije - v vsaki iteraciji DE se izvede ena generacija ($G$), vse dokler ni zadoščeno zaustavitvenemu kriteriju, kot sta npr.: maksimalno število ovrednotenj in doseganje določene kvalitete rešitve.



**Slika 2: Potek algoritma diferencialne evolucije.**

3. Mutacija - iz starševskih vektorjev ($\mathbf{X}_i^G$) ustvarimo ti. mutiran vektor ($\mathbf{M}_i^G$).

$$\begin{aligned} \mathbf{M}_i^G &= \mathbf{X}_{\mathbf{r}_1^i}^G + F * (\mathbf{X}_{\mathbf{r}_2^i}^G - \mathbf{X}_{\mathbf{r}_3^i}^G); \\ 0 &\le i < Np, \quad r_0^i \ne r_1^i \ne r_2^i \ne i \end{aligned} \tag{1}$$

   - $\mathbf{X}_{r_k^i}^G$ - naključni starševski vektor; $k$={1,2,3}
   - $F$ - mutacijski faktor

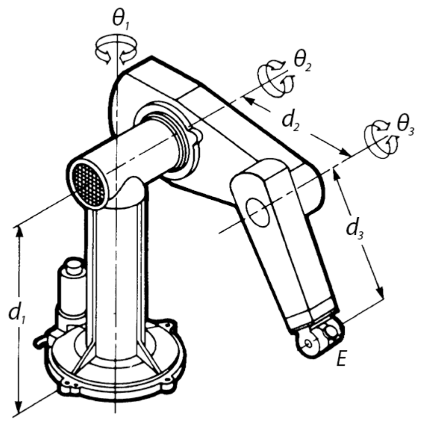
4. Križanje - definira postopek, v katerem iz starševskih in mutiranih vektorjev sestavimo preizkusni vektor ($\mathbf{P}_i^G$):

$$\mathbf{P}_{i,j}^G = \begin{cases} \mathbf{M}_{i,j}^G, & \text{če je Rand}_{i,j}[0,1) \le Cr \\ \mathbf{X}_{i,j}^G, & \text{drugače} \end{cases} \tag{2}$$

   - $Cr$ - faktor križanja

5. Selekcija - vsak $i$-ti preizkusni in pripadajoči starševski vektor ocenimo s pomočjo ocenitvene funkcije ($eval$). V kolikor je ocena za preizkusni vektor boljši od ocene za starševski vektor, potem preizkusni vektor zamenja starševski vektor v populaciji, kot prikazuje naslednja enačba:

$$\mathbf{X}_i^{G+1} = \begin{cases} \mathbf{P}_i^G, & \text{če je } eval(\mathbf{P}_i^G) \ge eval(\mathbf{X}_i^G) \\ \mathbf{X}_i^G, & \text{drugače} \end{cases} \tag{3}$$



**Slika 3: Puma 560 s parametri [?].**

6. Doseganje izstopnega pogoja - evolucijski proces algoritma DE se zaključi, ko obvelja kateri izmed naštetih pogojev:
   - doseženo maksimalno število ovrednotenj posameznikov,
   - preseženo časovno obdobje,
   - najboljša rešitev v populaciji že določeno število generacij ostaja ista - algoritem je obstal v minimumu / maksimumu in
   - najboljša rešitev je enaka ali boljša od pričakovane.
7. Rezultati DE - rezultat je posameznik, ki je dosegel najboljši rezultat glede na ocenitveno funkcijo (*eval*).

## 4. REŠITEV

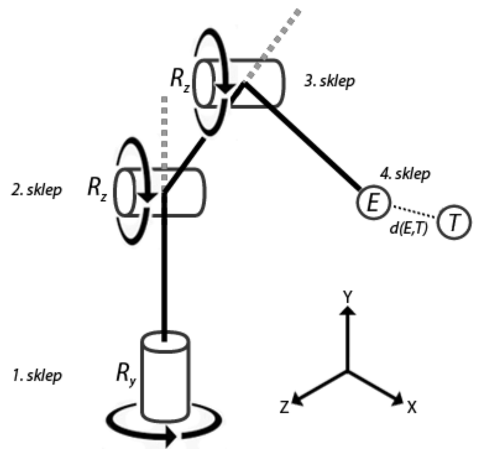
Problem IK zasledimo na mnogih področjih, a na nobenem drugem področju ne pride tako do izraza kot v robotiki. Iz tega razloga bomo predlagano rešitev predstavili na problemu IK robotske roke PUMA 560 (Programmable Universal Machine for Assembly).

Roka obsega vsega 7 sklepov, kjer je prvih 6 sklepov nastavljivih, zadnji pa predstavlja končno točko IK - $E$. Ker so povezave pri PUMA 560 fiksnih dolžin ($d_{min,i} = d_{max,i}; i = \{0-6\}$), to pomeni, da ima vsak sklep nastavljiv samo parameter $\theta_i$. Pri 6 nastavljivih sklepih je PUMA 560 omejena s 6 DOF (Degree Of Freedom - število neodvisnih parametrov robotske roke, s katerimi jo lahko manipuliramo).

V našem primeru se bomo omejili na prve 3 sklepe iz naslednjih razlogov:

- prvi trije sklepi opravijo večino pozicioniranja in
- glava roke je rotacijska in vsebuje preostale sklepe, ki pa so analitično rešljivi.

Vse 3 sklepe iz glave robotske roke bomo združili v končni četrti sklep $E$ (slika 3). Koordinate sklepa $E$ bodo tiste, ki



**Slika 4: Skelet PUME 560.**

**Tabela 1: Omejitve za prve 3 sklepe PUMA 560 [?].**

| sklep | $\boldsymbol{\theta}_{min,i}(^\circ)$ | $\boldsymbol{\theta}_{max,i}(^\circ)$ | $\mathbf{d}_{min,i}(mm)$ | $\mathbf{d}_{max,i}(mm)$ | os$(R)$ |
|---|---|---|---|---|---|
| 1 | -160 | 160 | 672 | 672 | $\mathbf{R}_y$ |
| 2 | -180 | 70 | 432 | 432 | $\mathbf{R}_z$ |
| 3 | -45 | 220 | 433 | 433 | $\mathbf{R}_z$ |

nas bodo v nadaljevanju tudi zanimale, saj bomo skušali ta sklep s pomočjo algoritma DE čim bolj približati poljubni točki v prostoru ($T$ - *target*).

Preselimo sedaj problem IK PUME 560 v 3D kartezični prostor. Kot smo že omenili, bomo rešitev IK dobili tako, da bomo minimizirali razdaljo med sklepom $E$ in točko $T$ (slika 4). To bomo dosegli z računanjem razdalje med dvema točkama oz. uporabo naslednje ocenitvene funkcije:

$$eval = d(E,T) = \sqrt{\sum_{i=0}^{n}(x_i - y_i)^2} \;\; ; \quad n < 3.$$

Kot vse robotske roke, ima tudi PUMA 560 nekatere omejitve (glej tabelo 1). Poleg dolžin povezav, so tu tudi omejitve pri rotaciji. Posamezen sklep lahko namreč zavzame poljuben kot z intervala $[\theta_{min,i}, \theta_{max,i}]$ okrog določene osi.

Iz omejitev (tabela 1) definiramo začetno populacijo algoritma DE. Vsak njen posameznik, $\mathbf{X}_i$, je vektor z enako dimenzijo, kot je DOF obravnavanega problema. V našem primeru, kjer imamo 3DOF problem, imamo tako posameznike z $D = 3$. Posamezna komponenta posameznika predstavlja kot $\theta_i$ $j$-tega sklepa ($\mathbf{X}^0_{i,j} = \theta_j$).

Vsak posameznik tako predstavlja morebitno rešitev problema, saj vsebuje vse parametre, s katerimi upravljamo z robotsko roko. Pri mutaciji moramo paziti, da s katero od komponent, (v našem primeru $\theta_j$) ne zapustimo predefiniranega intervala. V kolikor bi kršili omejitve, bi jih lahko kršila tudi končna rešitev in bi bila nedopustna. Vsakič, ko kličemo ocenitveno funkcijo v koraku selekcije, moramo najprej iz parametrov posameznega vektorja izračunati FK. Šele ko poznamo rezultat FK, torej koordinate točke $E$, lahko iz-

Tabela 2: **Uporabljene konstante pri meritvah.**

| Konstanta | Vrednost |
|---|---|
| *robot* | PUMA 560 |
| *št. zagonov* | 100 |
| $F$ | 0,5 |
| $Cr$ | 0,9 |
| $\epsilon$ | 1 mm |
| maksimalno število ovrednotenj | 100.000 |
| časovna omejitev | – |

**Tabela 3: Meritve nad PUMA 560.**

| # | T | Uspešnost (%) | | |
|---|---|---|---|---|
| | | 10 | 100 | 300 |
| 1 | {350,00; 350,00; 350,00} | 23 | 98 | 100 |
| 2 | {250,30; 420,80; 630,10} | 11 | 100 | 100 |
| 3 | {-315,60; 430,20; -650,30} | 23 | 100 | 100 |
| 4 | {420,30; 601,20; -420,80} | 34 | 100 | 100 |
| 5 | {301,40; 720,10; -400,60} | 42 | 100 | 100 |
| 6 | {415,30; 903,80; -700,90} | 32 | 100 | 100 |
| 7 | {202,80; 1000,00; 712,67} | 38 | 100 | 100 |
| 8 | {-385,20; 170,34; -584,53} | 32 | 100 | 100 |
| 9 | {-537,65; 892,97; 364,23} | 33 | 100 | 100 |
| 10 | {-754,34; 734,67; 734,14} | 0 | 0 | 0 |

**Tabela 4: Podrobnejši rezultati za točko 6.**

| $Np$ | Povprečje | | Najboljši posameznik | |
|---|---|---|---|---|
| | čas (s) | napaka (mm) | čas (s) | napaka (mm) |
| 100 | 0,42 | 0,74 | 0,39 | 0,28 |
| 300 | 2,28 | 0,74 | 0,74 | 0,30 |

računamo evklidsko razdaljo. Algoritem DE bo tako skozi generacije optimiziral parametre posameznikov v populaciji, dokler ne bo našel rešitve IK ali pa bo dosegel katerega od izstopnih pogojev. Rešitev za IK je najdena, kadar je evklidska razdalja med $E$ in $T$ manjša od tiste, ki smo jo predpisali: $E - T < \epsilon$.

## 5. EKSPERIMENTI

Eksperiment smo izvajali s pomočjo algoritma DE in nastavitev, ki so prikazane v tabeli 2. Tabela 3 prikazuje dobljene rezultate eksperimenta. Za vsako točko in tri različne vrednostih $Np = \{10, 100, 300\}$ smo izvedli 100 neodvisnih zagonov. Uspešnost predstavlja odstotek, koliko zagonov izmed 100-ih je našlo rešitev IK. Vidimo, da za zadnjo točko algoritem ni našel rešitev. Razlog temu je ciljna točka, ki je izven dosega robota. Možno je tudi, da algoritem za isti problem pri drugačnem semenu najde različno rešitev, ki pa je lahko prav tako pravilna (dopustna).

V tabeli 4 so podrobneje predstavljeni rezultati za točko 6, kjer je algoritem v obeh primerih za $Np = 100$ in $Np = 300$ dosegel 100 % uspešnost. Iz tabele vidimo, da se v prvem primeru algoritem konča več kot petkrat hitreje in s skoraj štiri krat manj klici ocenitvene funkcije. Iz tega lahko sklepamo, da je $Np = 100$ veliko boljša izbira za velikost populacije. Vendar moramo biti pazljivi, saj ni nujno, da bo rešitev zmeraj najdena (kar vidimo na primeru točke 1). V primeru majhne populacije $Np = 10$, pa se algoritem ni izkazal, saj zgolj z 10 posamezniki ni zmožen v celoti raziskati iskalnega prostora in se je verjetno ujel v lokalnem optimumu.

## 6. ZAKLJUČEK

Kot je razvidno iz rezultatov, algoritem uspešno najde rešitev za vsako podano točko (seveda za tiste, pri katerih je IK rešljiva). Uspešnost algoritma se povečuje z večanjem velikosti populacije, vendar moramo paziti, saj se obenem poveča tudi čas, ki je potreben, da to rešitev dosežemo. Algoritem je nastavljiv in lahko sprejme različne nastavitve različnih robotskih rok (poljubno število DOF), tudi takšnih, kjer dolžine povezav niso fiksne. Vendar z bolj kompleksnimi robotskimi rokami ali nasplošno s problemi IK moramo ponovno najti ustrezne nastavitve algoritma.

## Zahvala

Janez Brest in Borko Bošković priznavata financiranje prispevka s strani Javne agencije za raziskovalno dejavnost Republike Slovenije, raziskovalni program P2-0041 – Računalniški sistemi, metodologije in inteligentne storitve.

## LITERATURA

[1] Ikfast: The robot kinematics compiler. http://openrave.org/docs/. 29.8.2017.

[2] Inverse kinematics - basic methods. http://old.cescg.org/CESCG-2002/LBarinka/. 29.8.2017.

[3] B. Bošković and J. Brest. Differential evolution for protein folding optimization based on a three-dimensional AB off-lattice model. *Journal of Molecular Modeling*, 22:1–15, 2016.

[4] B. Bošković and J. Brest. Clustering and Differential Evolution for Multimodal Optimization. In *2017 IEEE Congress on Evolutionary Computation*, pages 698–705, 2017.

[5] B. Bošković, J. Brest, A. Zamuda, S. Greiner, and V. Žumer. History Mechanism Supported Differential Evolution for Chess Evaluation Function Tuning. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, 15:667–682, 2011.

[6] J. Craig. *Introduction to Robotics: Mechanics and Control*. Addison-Wesley series in electrical and computer engineering: control engineering. Pearson/Prentice Hall, 2005.

[7] G. Pampara, A. P. Engelbrecht, and N. Franken. Binary differential evolution. In *2006 IEEE International Conference on Evolutionary Computation*, pages 1873–1879, 2006.

[8] R. Paul. *Robot Manipulators: Mathematics, Programming, and Control : the Computer Control of Robot Manipulators*. Artificial Intelligence Series. MIT Press, 1981.

[9] J. Rutherford. Using the puma560 robot;a user's guide to basic operation of the puma arm robot, 2012.

[10] R. Storn and K. Price. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359, Dec 1997.

[11] F. Zhongtao, W. Yang, and Z. Yang. Solution of inverse kinematics for 6r robot manipulators with offset wrist based on geometric algebra. 5:310081–310087, 08 2013.

# Evolucijski algoritem za poravnavo sekvenc

Adel Bureković
Univerza v Mariboru
Fakulteta za elektrotehniko, računalništvo in informatiko
Koroška cesta 46
2000 Maribor
burekovic.adel@gmail.com

Janez Brest
Univerza v Mariboru
Fakulteta za elektrotehniko, računalništvo in informatiko
Koroška cesta 46
2000 Maribor
janez.brest@um.si

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

## POVZETEK

V članku predstavimo reševanje problema poravnave bioloških sekvenc s pomočjo evolucijskega algoritma. Kandidate v populaciji smo evaluirali z metodo sume parov. Ker je problem lahko časovno zahteven, smo algoritem implementirali s pomočjo jezika C++. Uspešnost algoritma smo testirali na DNK sekvencah različnih dolžin. Na skupini sekvenc povprečne dolžine 212 simbolov smo poravnali 111 stolpcev sekvenc. Pri sekvencah povprečne dolžine 1092 simbolov smo pa dosegli slabše rezultate z samo 43 poravnanimi stolpci.

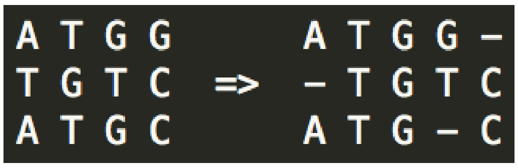
## Kjučne besede

poravnava sekvenc, evolucijski algoritem, križanje, mutacija, optimizacija

## 1. UVOD

Poravnava sekvenc je metoda, pri kateri je cilj najti podobnosti med naborom dveh (angl. pair-wise alignment) ali več (angl. multiple sequence alignment) sekvenc simbolov [8]. V bioinformatiki se sekvence DNK, RNK in proteinov poravnavajo z namenom iskanja skupnih značilnosti, ki bi nakazovale evolucijsko povezavo [8]. Težave pri poravnavah se pojavijo zaradi tega, ker se s procesom evolucije skozi čas v sekvencah pojavijo mutacije, kot so vnosi, brisanje in substitucija genetskih informacij, kar povzroči razlike med sekvencami [7].

Sekvence so predstavljene po vrsticah, v katerih poskušamo simbole preurediti na takšen način, da bo končni rezultat matrika, ki ima po stolpcih čim več enakih ali podobnih simbolov [8]. Po potrebi se v sekvence vnesejo tudi vrzeli (angl. gaps) z namenom pridobitve čim bolj optimalne poravnave [1]. V primeru sekvenc DNK so baze adenin, citozin, gvanin in timin v sekvenci predstavljene s simboli `A`, `C`, `G` in `T`. Primer poravnave treh manjših sekvenc DNK z vnosom vrzeli prikazuje slika 1. Najbolj optimalna poravnava je bila dosežena z povečanjem maksimalne dolžine sekvenc in vstavljanjem vrzeli.



```
A T G G       A T G G -
T G T C  =>   - T G T C
A T G C       A T G - C
```

**Slika 1: Primer poravnave sekvenc z vnosom vrzeli.**

V literaturi zasledimo, da so se pri poravnavi dveh sekvenc izkazali za zelo uspešne algoritmi, ki uporabljajo metode dinamičnega programiranja. Že leta 1970 sta Needleman in Wunsch predlagala metodo za poravnavo dveh sekvenc [9]. Dinamično programiranje se ne uporablja za primerjavo večjega števila sekvenc zaradi računske zahtevnosti $O(n^k)$, kjer $k$ predstavlja število sekvenc [7]. Zaradi velike računske zahtevnosti se pri primerjavi večjega števila sekvenc uporabljajo metode, ki ne zagotavljajo najboljše poravnave [8]. Med takšne spadajo progresivne metode [5], ki so med drugimi implementirane tudi v Clustal Omega [12] in T-Coffee [10]. Drugi možen pristop so iterativne metode, ki poskušajo skozi iteracije izboljšati začetno rešitev tako, da maksimizirajo dano ocenitveno funkcijo [8]. Med iterativne pristope spada tudi naša implementacija z evolucijskim algoritmom [8].

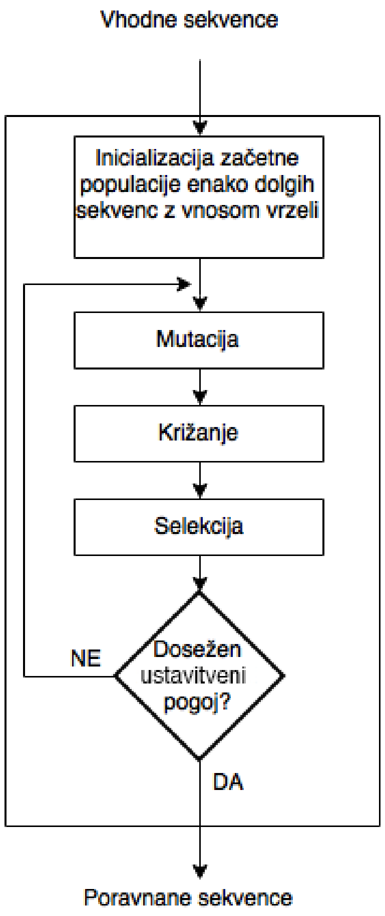
V drugem poglavju opišemo implementiran algoritem za poravnavo sekvenc s pomočjo evolucijskega algoritma. V tretjem poglavju predstavimo program in eksperimente, kjer testiramo uspešnost algoritma na različnih sekvencah DNK. Zadnje poglavje je namenjeno razpravi.

## 2. EVOLUCIJSKI ALGORITEM

Evolucijski algoritmi so metode, ki poskušajo z oponašanjem naravnih procesov križanja, mutacije in selekcije izboljšati populacijo kandidatov. Primerni so za reševanje različnih kombinatoričnih problemov kot sta npr. optimizacija zvijanja proteinov [2] in poravnava sekvenc. V našem algoritmu so sekvence v kandidatih predstavljene v dvodimenzionalnem polju po vrsticah. V algoritmu je vsak kandidat v populaciji generiran naključno glede na vhodne sekvence in predstavlja potencialno rešitev. Implementacijo algoritma prikazuje slika 2.

Kvaliteto kandidatov smo ocenili z metodo vsote parov (angl. sum of pairs) [11], pri kateri smo vsak stolpec ovrednotili po naslednji enačbi:

DOI: https://doi.org/10.26493/978-961-7023-40-4.27-30


Vhodne sekvence
Inicializacija začetne populacije enako dolgih sekvenc z vnosom vrzeli
Mutacija
Križanje
Selekcija
Dosežen ustavitveni pogoj?
NE
DA
Poravnane sekvence

**Slika 2: Potek evolucijskega algoritma.**

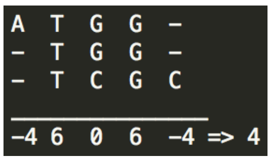
$$SP\_C(i) = \sum_{1\leq p\leq q\leq k} eval(S_{p,i}, S_{q,i})$$

- $i$ - oznaka stolpca
- $p$, $q$ - oznaki sekvenc
- $k$ - število vseh sekvenc
- $S$ - sekvenca

V vsakem stolpcu smo medsebojno primerjali simbole, kjer se je h končni oceni ocenitvene funkcije dodala vrednost po naslednjih kriterijih [7]:

- 2 v primeru dveh enakih simbolov,
- -2 v primeru dveh vrzeli in
- -1 v primeru vrzeli in simbola.

Primer uporabe metode sume parov na treh sekvencah prikazuje slika 3, kjer je bil končni rezultat ocenitvene funkcije suma vseh vrednosti stolpcev.



```
A  T  G  G  –
–  T  G  G  –
–  T  C  G  C
______________
–4 6  0  6  –4 => 4
```

**Slika 3: Primer izračuna sume parov na treh sekvencah.**

Za uspešno poravnavo sekvenc je potrebno imeti sekvence enakih dolžin. Zaradi mutacij zelo pogosto to ni možno, kar smo rešili z vstavljanjem vrzeli. Dolžino sekvenc smo zato določili po enačbi:

$$L = l_{max} \times (1 + r_{sp}) \tag{1}$$

kjer $l_{max}$ predstavlja najdaljšo sekvenco in $r_{sp}$ skalirni faktor, s katerim povečamo dolžino sekvenc [7]. $r_{sp}$ je pozitivno število za katerega je priporočena vrednost 0,2 [7]. Določiti primerno dolžino sekvenc je pomembno, ker se v primeru premajhne vrednosti lahko zgodi, da ne najdemo optimalne rešitve pri sekvencah s slabšo podobnostjo, in nasprotno, pri predolgih sekvencah prevelik iskalni prostor lahko pomeni, da bo iskanje optimalne rešitve lahko trajalo veliko dlje časa [7]. Vsako sekvenco naključno zapolnimo z vrzelmi tako, da bo imela dolžino $L$.

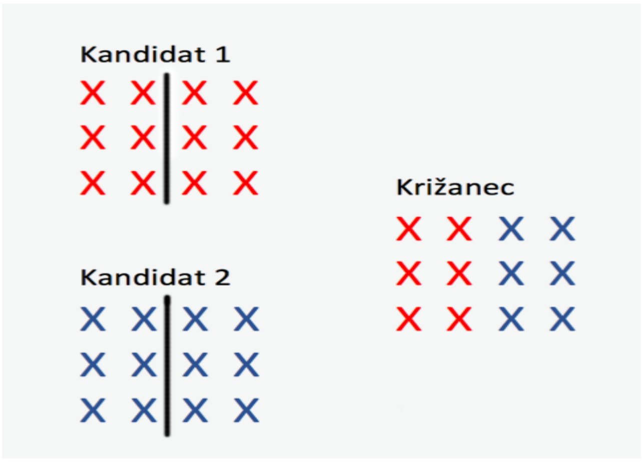
Lokacije vrzeli smo generirali podobno kot v [3], s tem da smo v vsaki sekvenci ustvarili permutacijo $L$ števil za vsako sekvenco. Iz vsake permutacije smo vzeli toliko števil, kot je originalna dolžina sekvence in jih uredili po naraščajočem vrstnem redu. Ta števila predstavljajo lokacije simbolov v sekvenci. Preostale indekse smo zapolnili z vrzelmi. Če imamo v kandidatu na primer DNK sekvenco: `CTG`, kjer je vrednost $L$ enaka 5 in generirano permutacijo: 3, 2, 5, 1, 4; bo izgled končne sekvence: `–CT–G`.

Križanje smo povzeli po [6], kjer se je naključnega kandidata iz populacije z trenutnim lahko križalo horizontalno ali vertikalno. Tip križanja je bil izbran naključno, možnost obeh križanj je bila 50 %. Horizontalno križanje deluje tako, da smo za vsako vrstico v križancu naključno izbrali sekvenco iz enega od kandidatov (slika 5). Pri vertikalnem križanju smo naključno izbrali točko preseka, pri kateri dobi križanec vsebino vsake sekvence do točke preseka skopirano od prvega kandidata in od točke preseka naprej od drugega kandidata (slika 4). Pri vertikalnem križanju smo morali paziti, da se struktura bioloških sekvenc ohrani.

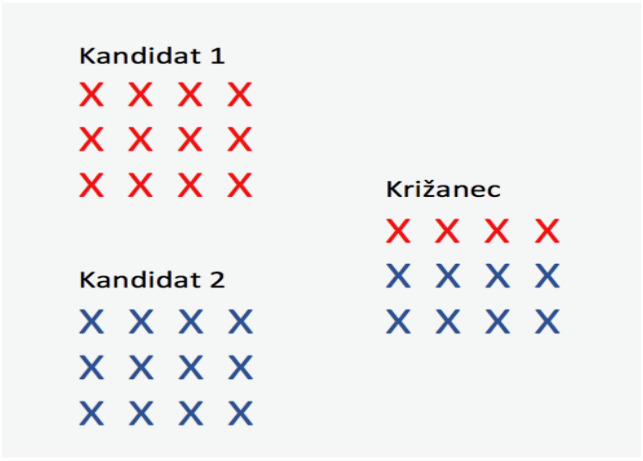
Mutacijski operator je bil implementiran tako, da smo naključno izbrali sekvenco v kandidatu in poiskali takšen simbol, ki je imel vsaj eno sosednjo vrzel. Za vsako sosednjo vrzel izbranega simbola smo zamenjali simbola v indeksih sekvence in preverili novo vrednost ocenitvene funkcije. Od vseh možnih zamenjav smo izbrali različico kandidata z največjo vrednostjo ocenitvene funkcije [3].

V procesu selekcije smo mutiranca zamenjali s trenutnim kandidatom v populaciji v primeru, če je bil bolje ovrednoten s strani ocenitvene funkcije.

## 3. EKSPERIMENTI



Slika 4: Vertikalno križanje.



Slika 5: Horizontalno križanje.

Zaradi hitrosti smo algoritem implementirali v jeziku C++. Eksperimente smo izvajali na osebnem računalniku z 16GB spomina, 2.2GHz Intel Core i7 procesorjem in MacOS operacijskim sistemom.

Učinkovitost algoritma smo primerjali z uveljavljenim programskim orodjem Clustal Omega, ki spada v družino programov Clustal (http://www.ebi.ac.uk/Tools/msa/clustalo/). Uspešnost poravnav obeh algoritmov smo ocenili s številom polno ujemajočih se stolpcev v poravnavi. Uporabili smo podoben nabor podatkov, kot je bil uporabljen v [4] (tabela 2). Sekvence omenjene v [4] smo prenesli iz spletne strani evropskega inštituta za bioinformatiko (http://www.ebi.ac.uk).

V implementiranem programu smo sekvence podali v tekstovnem formatu FASTA in nastavili parametre, kot jih prikazuje tabela 1. Algoritem smo ustavili, po 100000 klicih ocenitvene funkcije.

Kot je vidno iz tabele 3, nam je na skupini sekvenc D1 in D2 uspelo polno poravnati 43 in 111 stolpcev. Programsko orodje Clustal Omega pa se je na istih sekvencah izkazalo za boljše z 854 polno poravnanimi stolpci v skupini sekvenc D1 in 206 v skupini sekvenc D2.

Tabela 1: Nastavitve algoritma.

| Parameter | Vrednost |
|---|---|
| Velikost populacije | 60 |
| Dovoljeno št. ovrednotenj | 100000 |
| Prostorsko razmerje | 0,2 |
| Horizontalno križanje | 0,5 |
| Vertikalno križanje | 0,5 |

Tabela 2: Nabor podatkov [4].

| Ime | Št. sekvenc | Povprečna dolžina (Najkrajša sekv., Najdaljša sekv.) | Identifikatorji sekvenc |
|---|---|---|---|
| D1 | 5 | 1092 (1006, 1142) | M60059, M60060, M60061, M60062, M60063 |
| D2 | 6 | 212 (211, 212) | Z75841, Z75850, Z75854, Z75852, Z75858, Z75851 |

Tabela 3: Rezultati

| Ime | Št. ujemajočih stolpcev z EA | Št. ujemajočih stolpcev z Clustal Omega |
|---|---|---|
| D1 | 43 | 854 |
| D2 | 111 | 206 |

## 4. ZAKLJUČEK

V članku smo predstavili metodo poravnave sekvenc s pomočjo evolucijskega algoritma. Z našo implementacijo smo uspeli doseči poravnavo sekvenc DNK. Algoritem bi lahko izboljšali v procesu mutacije, kjer bi z več različnimi tipi mutacij lahko dosegli večjo raznolikost populacije. Potrebno bi bilo narediti tudi analizo vrednosti krmilnih parametrov. Npr. z večjo populacijo in daljšim evolucijskim procesom bi verjetno dobili boljše rezultate. Implementacija bolj kompleksne ocenitvene funkcije in generacija začetne populacije, ki ni naključno generirana, predstavljata tudi dve izboljšavi za pridobitev boljše končne poravnave. Razvoj algoritma se bo nadaljeval v magistrskem delu, kjer bomo poleg poravnave sekvenc DNK implementirali še zmožnost poravnave proteinov s pomočjo različnih ocenitvenih matrik. Algoritem bomo bolj podrobno primerjali z drugimi sorodnimi rešitvami na večjem naboru podatkov.

## Zahvala

Janez Brest in Borko Bošković priznavata financiranje prispevka s strani Javne agencije za raziskovalno dejavnost Republike Slovenije, raziskovalni program P2-0041 – Računalniški sistemi, metodologije in inteligentne storitve.

## LITERATURA

[1] PORAVNAVANJE ZAPOREDIJ. http://web.bf.uni-lj.si/bi/biokemija/bioinfo/2007/Material/Poravn1.htm, 2007. [Online; accessed 29-August-2017].
[2] B. Bošković and J. Brest. Genetic Algorithm with Advanced Mechanisms Applied to the Protein

Structure Prediction in a Hydrophobic-Polar Model and Cubic Lattice. *Applied Soft Computing*, 45:61–70, 2016.

[3] K. Chellapilla and G. B. Fogel. Multiple sequence alignment using evolutionary programming. In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, volume 1, page 452 Vol. 1, 1999.

[4] S. Chen, C. Lin, et al. Multiple dna sequence alignment based on genetic simulated annealing techniques. *International journal of information and management sciences*, 18(2):97, 2007.

[5] D.-F. Feng and R. F. Doolittle. Progressive sequence alignment as a prerequisitetto correct phylogenetic trees. *Journal of Molecular Evolution*, 25(4):351–360, Aug 1987.

[6] C. Gondro and B. Kinghorn. A simple genetic algorithm for multiple sequence alignment. *Genetics and Molecular Research*, 6(4):964–982, 2007.

[7] J.-T. Horng, L.-C. Wu, C.-M. Lin, and B.-H. Yang. A genetic algorithm for multiple sequence alignment. *Soft Computing*, 9(6):407–420, Jun 2005.

[8] D. W. Mount. *Bioinformatics: Sequence and Genome Analysis 2nd edition*. Cold Spring Harbor Laboratory Press, 1986.

[9] S. Needleman and C. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443–453, 3 1970.

[10] C. Notredame, D. G. Higgins, and J. Heringa. T-coffee: A novel method for fast and accurate multiple sequence alignment. *J Mol Biol*, 302(1):205–17, Sep 8 2000. Journal ArticleResearch Support, Non-U.S. Gov't.

[11] J. C. Setubal and J. Meidanis. *Introduction to Computational Molecular Biology*. Computer Science Series. PWS Pub., 1997.

[12] F. Sievers, A. Wilm, D. Dineen, T. J. Gibson, K. Karplus, W. Li, R. Lopez, H. McWilliam, M. Remmert, J. Söding, J. D. Thompson, and D. G. Higgins. Fast, scalable generation of high-quality protein multiple sequence alignments using clustal omega. *Molecular Systems Biology*, 7(1), 2011.

# Bežalec: The sport competition management software

Dušan Fister
Univerza v Mariboru
Fakulteta za strojništvo
Smetanova 17, Maribor
dusan.fister@
student.um.si

Uroš Mlakar
Univerza v Mariboru
Fakulteta za elektrotehniko, računalništvo in informatiko
Koroška 46, Maribor
uros.mlakar@um.si

## ABSTRACT
There is a need for automatic management software in every area. This paper proposes a simple software application for managing competitors registration, editing, exporting and generating their standings and printing certificates. QtCreator was used to handle the visualization and integration of different tasks. The managing software, called ”Bežalec”, has been practically tested on a local running competition and performed well enough.

## Keywords
management software, automation, integration

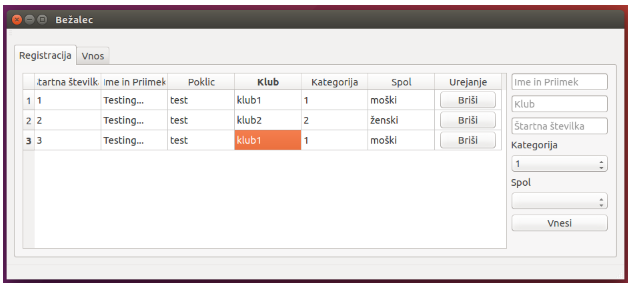
## 1. INTRODUCTION
Automatic management software is intended to simplify the registration, generation of result lists and commemoration certificate printing for sport competitions. It is desired, that this software is completely automatic and minimum user intervention is required. The application has to offer some additional tasks, e.g. uploading result lists to the website. The software, in general, has to be suitable for running competitions, but should be extendible to cycling and swimming competitions and other sports with single criterion - performed time.

This paper proposes the structure of an application, that was built for the purposes of a local running competition. This application was built in the C++ programming language in programming environment QtCreator, under Linux operating system [2]. The Qt library was used for designing the graphical user interface (GUI) of the application. Besides the GUI design, *awk*, *sed* and *Perl* scripts were integrated into the main application, as well as MySQL database. Some other components are included into the applications, e.g. *lp* printing tool, *pdflatex* and html generator. The application is currently suitable for running competitions, as shown in logo (Fig. 1).

**Figure 1: The management software ”Bežalec” logo.**

The structure of this paper is as follows: in the second chapter, the sport management software and its parts are described. In the third chapter, a short discussion is outlined regarding the usability of the application and the potential to extend it for using in other sports. The paper is concluded in chapter 4.

## 2. THE SOFTWARE
The software was written, as already mentioned, in QtCreator programming environment. The product of source code is a single executable script, which runs the following main dialog window, shown in Fig. 2. It is worth to mention, that the management software is completely operated from this dialog window. The dialog window consists of two pages, i.e. first ”Registracija” page and second ”Vnos” page. Names are written in Slovene and therefore require translations.



| | tartna številk | Ime in Priimek | Poklic | Klub | Kategorija | Spol | Urejanje |
|---|---|---|---|---|---|---|---|
| 1 | 1 | Testing... | test | klub1 | 1 | moški | Briši |
| 2 | 2 | Testing... | test | klub2 | 2 | ženski | Briši |
| 3 | 3 | Testing... | test | klub1 | 1 | moški | Briši |

**Figure 2: The management software main dialog window.**

The first page of the shown main dialog window is dedicated for registration of competitors - athletes. They are organized in a list in tabular way, with each athlete occupying one row of the table. Basic data about every athlete is taken, as following:

- starting number,
- name and surname,
- profession,
- club name,
- category and
- sex.

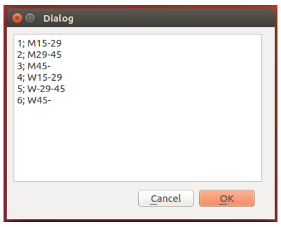
Those data are inputed into the management software via input fields on the right side of the main dialog window, shown in Fig. 2. After confirming input data, the athlete is added to the list, as well as exported into a text file and a MySQL database for back-up [1]. Details are given in subsection 2.2.

Every registered athlete is then treated by an incremental starting number, which is her/his identification number (ID). Data about an athlete can be, after successful registration, easily changed for any modifications. Athletes can be, in any case, removed from the registration list.

The main display window is extended by an auxiliary display window, shown in subsection 2.1. The main display window also contains the toolbar, from which basic settings can be arranged and the auxiliary window can be invoked.

## 2.1 Treating the categories

If basic data about athletes are standardized for every competition, categories however are not. Different sport competitions may have different ranges for age categories, e.g. some competitions rank young athletes into an age category from fifteen to twenty-nine years (M15-29), while some into under twenty-three (U23). This problem was solved by a custom category list, by assigning each category with a unique incremental ID. Fig. 3 shows the auxiliary dialog window for assigning categories.



1; M15-29
2; M29-45
3; M45-
4; W15-29
5; W-29-45
6; W45-

**Figure 3: Auxiliary dialog window for assigning categories.**

From the Fig. 3 it is seen, that any number of categories can be included, or added into the application, separated by a unique ID.

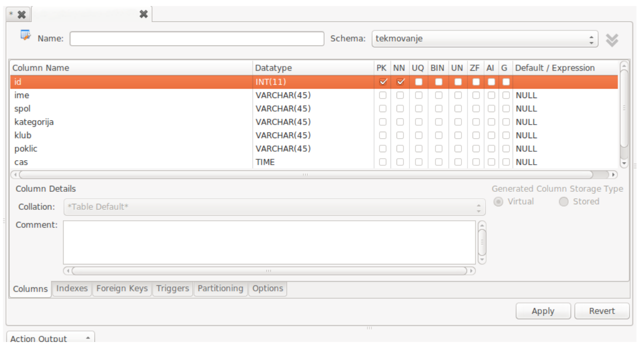
## 2.2 Import/export of registration data

The desire for importing/exporting athlete's data into a text file is applicable for easier manipulation of registration. Some competitions have an online pre-registration, where athletes register a few days before via Internet. It is therefore possible for a manager of this software to assemble pre-obtained data before the competition at home. After his work, the latest text file is generated, which can be imported on other computers. Generated text file is for computers without an installed MySQL database an essential tool, since it stores the data over a longer period of time.

Exporting athlete's registration data consists of two variants:

- exporting data into a text file and
- exporting data into a MySQL database.

The manner of using the MySQL database, in our case, is to back-up data, written in the text file. It is well-known, that MySQL databases offer different automatic functions over data, e.g. sorting, which is highly desired for organizing the start list. In case of a main dialog window error, it can restore data and guarantee other basic manipulations with data. Firstly, it is however necessary for the manager to install the MySQL database. The auto-generated MySQL table offers an outlook, as illustrated in Fig. 4.



Schema: tekmovanje

| Column Name | Datatype | PK | NN | UQ | BIN | UN | ZF | AI | G | Default / Expression |
|---|---|---|---|---|---|---|---|---|---|---|
| id | INT(11) | ✓ | ✓ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | |
| ime | VARCHAR(45) | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | NULL |
| spol | VARCHAR(45) | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | NULL |
| kategorija | VARCHAR(45) | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | NULL |
| klub | VARCHAR(45) | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | NULL |
| poklic | VARCHAR(45) | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | NULL |
| cas | TIME | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | ☐ | NULL |

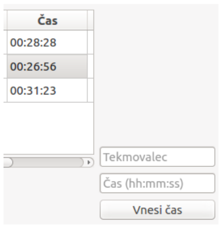
Column Details
Collation: *Table Default*
Comment:
Generated Column Storage Type: Virtual / Stored

**Figure 4: Snapshot from the MySQL database.**

From the obtained data and sorting of starting numbers, a start list can be generated.

## 2.3 Editing athlete's standings and result list generation

After the competition, it is necessary to input athlete's results. They are stored in the form of time in following format: hh:mm:ss. This task is performed on the second page of the main dialog window, which is illustrated in the Fig. 5.



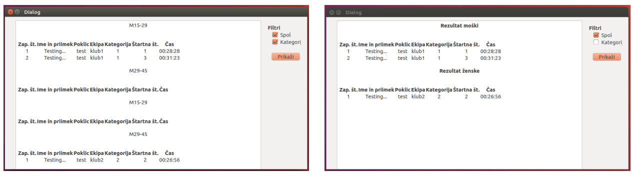
| Čas |
|---|
| 00:28:28 |
| 00:26:56 |
| 00:31:23 |

**Figure 5: Second page for inputting results.**

Two input fields are used for storing results, i.e. the unique ID of an athlete and his/her result. Using results from all athletes, a result list is generated, according to their performed time. Results are, besides shown in the table on the second page, also exported into the text file and the MySQL database. The following variants for generating the result lists are implemented:

- general classification results (for men and women) and
- age group results (for men and women).

General classification results consist of all athletes (who finished the competition), while age group results comprise only of athletes of suitable age. The difference between general classification results and age group results is shown in Fig. 6a and Fig. 6b.



a)

```
Dialog
                              M15-29

Zap. št. Ime in priimek Poklic Ekipa Kategorija Štartna št.   Čas
   1        Testing...    test  klub1     1          1      00:28:28
   2        Testing...    test  klub1     1          3      00:31:23

                              M29-45

Zap. št. Ime in priimek Poklic Ekipa Kategorija Štartna št. Čas

                              M15-29

Zap. št. Ime in priimek Poklic Ekipa Kategorija Štartna št. Čas

                              M29-45

Zap. št. Ime in priimek Poklic Ekipa Kategorija Štartna št.   Čas
   1        Testing...    test  klub2     2          2      00:26:56
```

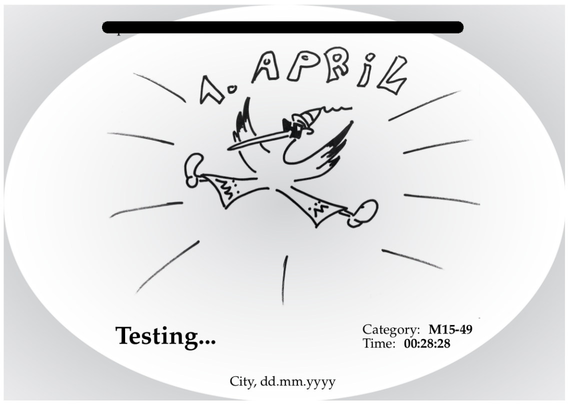
b)

```
Dialog
                          Rezultat moški

Zap. št. Ime in priimek Poklic Ekipa Kategorija Štartna št.   Čas
   1        Testing...    test  klub1     1          1      00:28:28
   2        Testing...    test  klub1     1          3      00:31:23

                          Rezultat ženske

Zap. št. Ime in priimek Poklic Ekipa Kategorija Štartna št.   Čas
   1        Testing...    test  klub2     2          2      00:26:56
```

**Figure 6: Difference between the general classification and age group results.**

## 2.4 Printing commemorative certificates

The local running competition provides commemorative certificates for their athletes. Thus, we would like to create them and print in a suitable format. Hence, we employed *sed* and *awk* tools for automatic editing of athlete's name, result and category in a pre-prepared latex source file, before a *pdflatex* tool was run under Linux operating system. The created certificate was printed using the *lp* printing tool in the next step. Fig. 7 displays the created commemorative certificate. Due to keeping rights of the competition, the figure is retouched.



[redacted]

**Testing...**

Category: **M15-49**
Time: **00:28:28**

City, dd.mm.yyyy

**Figure 7: Automatically created commemorative certificate.**

The whole "Bežalec" software package can be summarized by a use-case diagram in Figure 8.

## 2.5 Upload to the web page

In the end of the athlete's result list manipulation, it is desired that results are automatically uploaded to the website. For that reason, a Node.js JavaScript run-time environment was used [3].

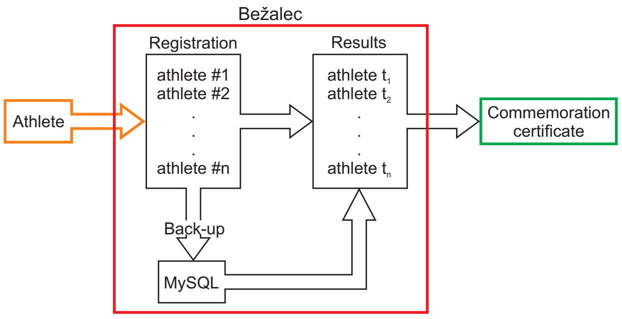
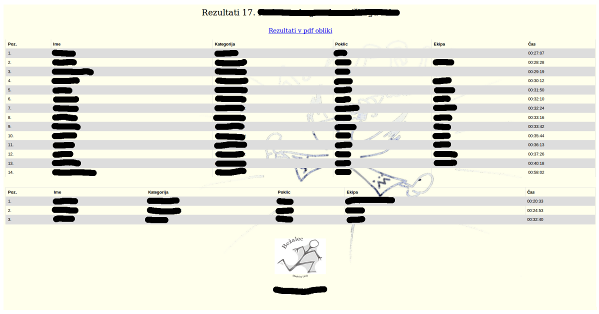


**Figure 8: The use-case diagram for software package "Bežalec".**



Rezultati 17. [redacted]

Rezultati v pdf obliki

| Poz. | Ime | Kategorija | Poklic | Ekipa | Čas |
|---|---|---|---|---|---|
| 1. | [redacted] | [redacted] | [redacted] |  | 00:27:07 |
| 2. | [redacted] | [redacted] | [redacted] | [redacted] | 00:28:28 |
| 3. | [redacted] | [redacted] | [redacted] |  | 00:29:19 |
| 4. | [redacted] | [redacted] | [redacted] | [redacted] | 00:30:12 |
| 5. | [redacted] | [redacted] | [redacted] | [redacted] | 00:31:50 |
| 6. | [redacted] | [redacted] | [redacted] | [redacted] | 00:32:10 |
| 7. | [redacted] | [redacted] | [redacted] | [redacted] | 00:32:24 |
| 8. | [redacted] | [redacted] | [redacted] | [redacted] | 00:33:16 |
| 9. | [redacted] | [redacted] | [redacted] | [redacted] | 00:33:42 |
| 10. | [redacted] | [redacted] | [redacted] | [redacted] | 00:35:44 |
| 11. | [redacted] | [redacted] | [redacted] | [redacted] | 00:36:13 |
| 12. | [redacted] | [redacted] | [redacted] | [redacted] | 00:37:26 |
| 13. | [redacted] | [redacted] | [redacted] | [redacted] | 00:40:18 |
| 14. | [redacted] | [redacted] | [redacted] |  | 00:58:02 |

| Poz. | Ime | Kategorija | Poklic | Ekipa | Čas |
|---|---|---|---|---|---|
| 1. | [redacted] | [redacted] | [redacted] | [redacted] | 00:20:33 |
| 2. | [redacted] | [redacted] | [redacted] | [redacted] | 00:24:53 |
| 3. | [redacted] | [redacted] | [redacted] | [redacted] | 00:32:40 |

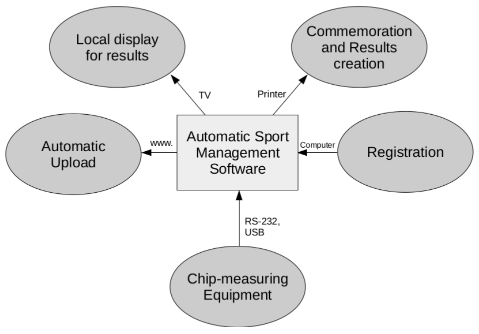
[redacted]

**Figure 9: Automatically uploaded results list.**

Fig. 9 (which is retouched, too) shows the uploaded result list in a html format (pdf format is opened in a new tab).

## 3. DISCUSSION

The idea of a sport management software does not merely present an application, but has a higher point of view. It can be upgraded by an autonomous chip-measuring equipment, where athletes run over a magnetised carpet and trigger an interrupt, which means precisely athlete's performed time (integration of chip-measuring equipment with management software was primarily our goal). The structure of a proposed automatic measuring system is shown in Fig. 10. It remains our goal for future work.



Local display for results
Commemoration and Results creation
TV
Printer
Automatic Upload
www.
Automatic Sport Management Software
Computer
Registration
RS-232, USB
Chip-measuring Equipment

**Figure 10: The structure of automatic measuring system.**

## 4. CONCLUSION

The application of sport management software has been tested on a local running competition, where seventeen athletes participated. The application successfully served during the registration, but stopped working after the start. However, the database provided a back-up and all data were restored. In this case, it was important, that MySQL database was employed for a back-up.

## 5. REFERENCES

[1] Oracle Corporation. MySQL 5.7 Reference Manual. Accessible at: https://dev.mysql.com/doc/refman/5.7/en/. Accessed: 29.8.2017.

[2] R. Rischpater. *Application development with qt creator*. Packt Publishing Ltd, 2013.

[3] S. Tilkov and S. Vinoski. Node. js: Using javascript to build high-performance network programs. *IEEE Internet Computing*, 14(6):80–83, 2010.

# Multi-agent system based on self-adaptive differential evolution for solving dynamic optimization problems

Aleš Čep
Faculty of Electrical Engineering and Computer Science
University of Maribor
ales.cep@um.si

Iztok Fister
Faculty of Electrical Engineering and Computer Science
University of Maribor
iztok.fister@um.si

## ABSTRACT
This article presents the multi-agent system based on a self-adaptive differential evolution algorithm for solving dynamic problems. Characteristics of dynamic problems are that objective function, with which the quality of solutions is evaluated, changes over time. These changes can occur either after some predefined number of generations or, more commonly, in each generation, where the online responses are desired by the evolutionary algorithms. In this study, the former kind of problems were taken into consideration and were solved using multi-agent systems. Each agent in the systems is implemented as a self-adapted differential evolution with constant population size. In our comparative study, multi-agent systems consisting of various population sizes in combination with various number of agents were compared, by solving the benchmark functions provided for CEC'09 Competition on Dynamic Optimization with respect to the same number of the fitness function evaluation. The main obstacle when using smaller populations is to prevent the stagnation of the population. That was partially overcome by using additional mechanisms such as: diversity measurement and information sharing between agents. As a result, the most appropriate multi-agent system was searched for, and the results of the best found multi-agent system were compared with the state-of-the-art algorithms.

## Keywords
Differential evolution, self-adaptive differential evolution, multi-agent system, dynamic problems

## 1. INTRODUCTION
Many of real-world problems are dynamic in their nature. Dynamic optimization problems (DOP) [4] can be described as problems, where decision maker has to make multiple decisions over time and the overall performance depends on all decisions made in that time. Decisions are made sequentially over time, but it depends on the type of problems, if this decisions are made by time triggered (periodic) or event triggered events. Consequently, the value of a fitness function that is optimal at some time, is not necessary optimal at another and vice versa.

Differential evolution (DE) was proposed by Storn and Price in [11] and is a powerful evolutionary algorithm (EA) frequently used for solving global optimization problems. Its simplicity and effectiveness have been proven in many real-world applications. DE emulates natural evolution process using three operators: mutation, crossover and selection. Brest et al [1] proposed jDE algorithm which extends the original DE algorithm with self-adaptation of DE control parameters, i.e., the scale rate and the frequency of crossover rate. This self-adaptation is performed before generating the trial vector and therefore it influences creation of new trial vectors. These controls parameters are added to a representation of individuals and undergo operation of variation operators (i.e., mutation and crossover) during the evolutionary cycle.

Morrison [8] presented new EA architecture for solving DOP where he uses so-called sentinels, i.e., individuals in a population that are spread across search space whose location does not change. In this way they can be used for detecting changes of environment. Brest et al [2] presented multi-population jDE (jDE*) algorithm with the ageing mechanism and the use of archive where they stored the current best individuals after the environment change was detected. There was no information sharing between individuals. Yang et al [13] proposed an algorithm for improving DE with population adaptation approach, where it calculates the standard deviation of individuals' in $j$-th dimension at generation $G$. Halder et al [5] presented an algorithm CDDE_Ar that uses a multi-population method where clusters (sub-populations) are partitioned according to the spatial locations of trial solutions. During the evolution process, population is periodically divided in different number of clusters which causes certain information sharing during the optimization process. Lepagnot et al [6] presented MLSDO algorithm, which is based on several coordinated local searches and on the archiving of the found local optima, in order to track them after a change in the objective function. Novoa et al [9] proposed an algorithm mSQDE-i, a multi-population algorithm with self-adaptive strategy for controlling the population diversity and an interaction mechanism between individuals.

This paper proposes multi-agent system based on self-ada-

DOI: https://doi.org/10.26493/978-961-7023-40-4.35-41

ptive differential evolution (MAS-jDE) for solving DOP. These are a class of problems, where the environment changes over time. Autonomous agents that are capable of detecting these changes on their own, try to solve DOP by using jDE and maintaining various sized populations. An evolutionary progress can typically stagnate when using agents with smaller populations. Therefore, we tried to prevail this problem by implementing two mechanism: detection of stagnation by calculating population's diversity and migration of individuals between agents. Experiments with various configurations of multi-agent systems were conducted, where the number of agents and the population sizes were varied. Indeed, these both parameters were selected such that the algorithms in each configurations spent the same number of the fitness function evaluations. In this way, fair comparison between different MAS-jDE configurations can be expected. Additionally, the performance of the proposed algorithm MAS-jDE was also compared to the others state-of-the-art algorithms for solving DOP.

The remainder of the paper is organized in the following way. Section 2 describes the DOPs. Section 3 describes the original DE, its extension jDE and the proposed MAS-jDE algorithm. Section 4 illustrates the results of experiments that were conducted and provides discussion on them. Finally, Section 5 concludes this paper with summarizing the performed work and outlining directions for the future work.

## 2. PROBLEM DEFINITION

This paper is devoted to solving DOPs. Li et al. in [7] defined these kind of problems as follows:

$$\text{DOP} = f(\mathbf{x}, \phi, t), \tag{1}$$

where DOP is the dynamic optimization problem, $f$ is an evaluation function, $x$ is a feasible solution in the solution set X, $\phi$ (so called change type) is a system control parameter determining the solution's distribution in the fitness landscape and $t$ is real-world time. Changes in environment can be results of a change of system control parameters describing environment state or a change in evaluation function. These changes can occur either after some predefined number of generations or, more commonly, in each generation, where the online responses are desired by the evolutionary algorithms.

The goal of the algorithm for solving DOP is to detect the change of environment and find the new optimum. Problem frequently occurring in evolutionary algorithms for DOP is a stagnation of population. Evolutionary process moves population to or very near to the local or global optimum, which causes a very small diversity of the population. When it comes to environmental change, the population with small diversity cannot explore search space efficiently enough. Usually this could mean, that the algorithm will not be able to find the next global optimum.

## 3. THE PROPOSED ALGORITHM

The proposed MAS-jDE algorithm combines a knowledge from two domains: evolutionary algorithms (EAs) [3] and multi-agent systems (MAS) [12]. In line with this, three algorithm are described at first, i.e., differential evolution (DE), followed by self-adaptation differential evolution (jDE) that is an extension of DE. Finally, the proposed MAS-jDE is presented from the multi-agent aspect.

### 3.1 Differential evolution

DE [11] is a well established EA, developed by Storn and Price [11]. It is a population-based algorithm, whose population is defined as a set of real-valued vectors representing different solutions of the problem:

$$\mathbf{x_i} = \{x_{i,1}, x_{i,2}, \ldots, x_{i,D}\}, \tag{2}$$

for $i = 1, \ldots, NP$, where $NP$ represents the population size and $D$ is the dimensionality of the problem. Like in other EAs, the first step in DE is a random initialization of the population. Until the termination condition is satisfied, the population undergoes operation of three evolutionary operators by creating trial vectors: mutation, crossover and selection.

Mutation creates mutant vector $v_i^{(G+1)}$ from parent's population. Different mutation DE strategies have been used in literature [10, 11], but in original article [11] authors used the so-called 'DE/rand/1/bin':

$$v_i^{(G+1)} = x_{r0}^{(G)} + F \cdot (x_{r1}^{(G)} - x_{r2}^{(G)}), \tag{3}$$

where $r0$, $r1$ and $r2$ represent random integer numbers in range $[1, NP]$, where it holds $r0 \neq r1 \neq r2$, and the scale factor is a real number in range $F \in [0, 2]$.

Crossover operator introduces a little diversity in the population of solutions. It creates a trial vector $u_{i,j}^{(G+1)}$ from a combination of elements from either the mutation vector or the corresponding parent vector as:

$$u_{i,j}^{(G+1)} = \begin{cases} v_{i,j}^{(G+1)} & \text{if } rand(0,1) \leq CR \text{ or } j = j_{rand}, \\ x_{i,j}^{(G)} & \text{otherwise,} \end{cases} \tag{4}$$

where the crossover rate is defined in range $CR \in [0, 1)$ and determines a probability of creating a trial vector from the mutation vector. Index $j_{rand}$ is randomly selected integer defined in range $[1, D]$ that is used to ensure that the algorithm makes at least one change in trial vector regarding previous generation.

Selection compares the results of fitness function from newly created trial vector $u_{i,j}^{(G+1)}$ and parent vector $x_{i,j}^{(G)}$. If the problem is minimization problem, it is defined as follows:

$$x_i^{(G+1)} = \begin{cases} u_i^{(G+1)} & \text{if } f(u_i^{(G+1)}) \leq f(x_i^{(G)}) \\ x_i^{(G)} & \text{otherwise.} \end{cases} \tag{5}$$

The fitness function depends on the problem to be solved.

### 3.2 Self-adaptive DE

Brest et al. [1] presented self-adaptive differential evolution algorithm (jDE), where control parameters $F_i$ and $CR_i$ are added to representation of individuals and adapted during the evolution process using following equations:

$$F_i^{(G+1)} = \begin{cases} F_l + rand_1 \cdot F_u & \text{if } rand_2 < \tau_1, \\ F_i^{(G)} & \text{otherwise,} \end{cases} \tag{6}$$

$$CR_i^{(G+1)} = \begin{cases} rand_3, & \text{if } rand_4 < \tau_2, \\ CR_i^{(G)} & \text{otherwise,} \end{cases} \tag{7}$$

where $rand_j, j \in [1-4]$ represents random number in range $[0,1]$ and $F_i^{(G+1)}$ is in range $[0.1, 1]$. Constants $\tau_1$ and $\tau_2$ are so-called learning rates determining the frequency of updating control parameters $F$ and $CR$, and were set to 0.1.

Adaptation of control parameters according to Eq. (6)-(7) is performed after 10 generations in average, and therefore has a big influence on mutation and crossover operators.

## 3.3 Multi-agent system based on jDE for solving DOPs

Multi-agent system based on jDE for solving DOPs (MAS-jDE) implements system of multiple autonomous agents, that run autonomously based on jDE algorithm using separated populations of the same size. Multi-agent systems are systems composed of multiple interfacing intelligent agents. Each agent is a computationally entity like an algorithm that is placed in an environment in order to achieve its objectives [12]. In our case, each agent is devoted to search for a best solution in own region of the whole search space. Using the interaction between agents, they are able to enrich their limited knowledge with informations obtained from other agents in the MAS.

There are four main issues that must be solved when developing the algorithm: agent's population size determination, environment change detection, maintaining the population diversity and information exchange between agents. According to our assumption of fairness, it holds that if system contains more agents, the population size of each agent is decreased. This fact can cause a stagnation problem, where algorithm gets stuck in a local optimum and can not get out, due to the lower population diversity. To avoid this problem, we introduce mechanism for measuring the population diversity that uses the following principle: When the diversity of population is insignificant, the MAS-jDE begins migration of individuals between agents. The pseudo-code of the algorithm is illustrated in Algorithm 1.

**Algorithm 1** Algorithm MAS-jDE

```
NP_MAS: sum of populations sizes
NP: size of agent's population
N: number of agents
 1: NP = $NP_{MAS}$/N
 2: Generate and evaluate all agent's populations
 3: Update global best
 4: while the termination condition is not meet do
 5:     for each agent's population $i$ do
 6:         Update parameters $F$ and $CR$ (Eq. (6)-(7))
 7:         Create trial population $j$ with DE operators mutation and crossover
 8:         Evaluate $j$
 9:         if environment change detected then
10:             Generate and evaluate new population
11:         else
12:             Selection between $i$ and $j$
13:     Update global best
14:     Check diversity and exchange individuals
```

In the remainder of the paper, the exposed issues of the proposed MAS-jDE algorithm are discussed in details.

### 3.3.1 Agent's population size determination

In order to ensure that the comparison between different MAS is as fair as possible, we introduce the total population size $NP_{MAS}$, that determines the number of vectors in the so-called grand population. Obviously, the MAS-jDE with grand population is equivalent to the original jDE using a single population. In MAS-jDE, the members of this grand population are divided among agents evenly. The higher the number of agents in the MAS, the smaller is their population size. In line with this, the agent's population size $NP$ is obtained as:

$$NP = \frac{NP_{MAS}}{N}, \tag{8}$$

where $N$ represents the number of agents. Because all three mentioned variables have to be integers, not all combinations of $NP_{MAS}$ and $N$ are feasible.

### 3.3.2 Environment change detection

An evolutionary cycle starts after a random initialization and evaluation of agent's population. The DE mutation and crossover operate on $NP-1$ trial vectors only, because the last vector is used as a sentinel [8] that never changes. Thus, the sentinel's fitness function value is changed with the change of the environment. Therefore, we first check after the newly generated population is evaluated, if the fitness value of sentinel distinguishes from the old value. In this case, we declare that the environment has changed, consequently reinitialize agent population randomly and evaluate the new fitness values for the whole population.

### 3.3.3 Maintaining the population diversity

The stagnation problem can be evident at the population as whole or at the component level [13]. In MAS-jDE, we detect stagnation by calculating agent's population diversity using the following equations [8, 13]:

$$a_j^{(G)} = \frac{\sum_{i=1}^{NP-1} x_{i,j}^{(G)}}{NP-1}, \tag{9}$$

$$diversity = \sqrt{\sum_{j=1}^{D} \sum_{i=1}^{NP-1} (x_{i,j}^{(G)} - a_j^{(G)})^2}, \tag{10}$$

where $a_j^{(G)}$ stores the mean of population in $j$th dimension. $j$ is in range $[1, D]$, where $D$ is a problem dimension. $\mathbf{x}_i$ is an individual in population.
Diversity in population is checked after each generation to detect stagnation as quickly as possible. If it falls bellow threshold level, migration between agents starts. Threshold level must be set high enough, so that agent has enough diversity in population to continue exploring search-space and low enough, so it can close-in to the optimum.

### 3.3.4 Information exchange between agents

When diversity in population falls bellow threshold, migration between agents starts (Algorithm 2). Idea is as follows. Each agent begins exploration in different part of search space and therefore it's search is limited to the specific region. When new vectors are migrated from another agent, it is expected that the diversity of population will increase. Thereby, the agent will be attracted to explore the new promising regions in the search space. Procedure of sharing information between agents is described in three steps, as follows:

1. step: The best vectors from all agents are found.
2. step: From these vectors, find the one that is the most diverse from the best vector in current agent based on Euclidian distance (Eq. (11)).
3. step: Using Euclidian distance (Eq. (11)), find two vectors that are closest to each other in the current agent (Algorithm 3) and replace the worse based on fitness function result, with the vector found is step 2.

Distance between vectors is calculated as Euclidean distance:

$$d(p, q) = \sqrt{\sum_{j=1}^{D} (p_j - q_j)^2}, \tag{11}$$

where $p$ and $q$ represent $D$ dimensional vector.

**Algorithm 2** Calculate diversity and exchange vectors

```
1: for each agent $a$ do
2:     Calculate population diversity for $a$ (Eq. (9, 10))
3:     if $a.diversity < threshold$ then
4:         $bestVectors[]=BestVectorForEachAgent()$
5:         Find the most diverse vector from the best vector
   in $a$
6:         $r$ = $IndexOfMostSimillar$(a)
7:         Replace vector on index $r$ with the most remote
   vector
```

**Algorithm 3** Algorithm that finds vector to replace

NP: size of agent's population
Agent's population is ordered by fitness value

```
1: function IndexOfMostSimillar(a)
2:     for $i$ = 0 to ($NP$ - 2) do
3:         for $j$ = $i$ + 1 to ($NP$ - 1) do
4:             $distance$ = $EuclidianDistance$(a[i], a[j])
5:             if $distance < smallestDistance$ then
6:                 $smallestDistance$ = $distance$
7:                 $ind = j$
8: return $a[ind]$
```

## 4. EXPERIMENT AND RESULTS

The goal of our experimental work was twofold. Firstly to test the MAS-jDE with different parameters settings, and thus find the best settings for parameters: the population size $NP$ and the number of agents $N$. Secondly to compare the results obtained by the MAS-jDE with the results of the other state-of-the-art algorithms. In both experiments, all algorithms solved CEC'09 benchmark test functions for DOP.

The algorithm MAS-jDE used the following parameters during experiments:

- the total population size was varied as $NP_{MAS} \in \{50, 100, 200, 400\}$,
- the number of agents was set to values $N \in \{1, 5, 8, 10, 16, 20, 25\}$, from which only the feasible (Table 1) were considered,
- the scale size was initially drawn randomly from uniform distribution in interval $F_i^{(0)} \in [0, 1]$,
- the crossover probability was initially drawn randomly from uniform distribution in interval $CR_i^{(0)} \in [0, 1]$,
- the diversity threshold in Algorithm 2 was set to $threshold = 1$.

### 4.1 Dynamic optimization benchmark

Algorithms in the study were tested on benchmark test suite provided by CEC'09 organizers for Competition on Dynamic Optimization [7], which uses the generalized dynamic benchmark generator. Benchmark defines seven change types: small-step change ($T_1$), large-step change ($T_2$), random change ($T_3$), chaotic change ($T_4$), recurrent change ($T_5$), recurrent change with noise ($T_6$), and random change with changed dimension ($T_7$).
There are six test functions defined in the real-space:

- $F_1$: Rotation peak function (with 10 and 50 peaks)
- $F_2$: Composition of Sphere's function
- $F_3$: Composition of Rastrigin's function
- $F_4$: Composition of Griewank's function
- $F_5$: Composition of Ackley's function
- $F_6$: Hybrid Composition function

In summary, there are 49 specific test cases and each test case contains 60 changes. Test cases run independently 20 times to obtain the mark on specific case. Overall performance $perf$ is the sum of the performance measures obtained for each function and each change type as follows:

$$perf = \sum_{i=1}^{49} mark_i \cdot 100, \tag{12}$$

where the performance measure $mark_i$ is defined as follows:

$$mark_i = \frac{w_i}{R \cdot C} \sum_{l=1}^{R} \sum_{j=1}^{C} r_{lj}, \tag{13}$$

where $w_i$ denotes weight of the problem $i$ (see [7] for more details), R the number of runs and C is the number of changes (in our case $R = 20$ and $C = 60$). The variable $r_{lj}$ is expressed as:

$$r_{lj} = r_{lj}^{last} / (1 + \sum_{s=1}^{S} (1 - r_{lj}^{s}) / S), \tag{14}$$

where $r_{lj}^{last}$ denotes the relative ratio between fitness value of best individual and global optimum of the $j$-th environment, $r_{lj}^{s}$ the relative ratio between fitness value of best individual and global optimum at the $s$-th sampling during one change, and $S$ is total number of samples for each environment.

## 4.2 Computer configuration

The computer used for experimental work had the following configuration:

- **System:** Windows 10
- **CPU:** Intel Core i7-4790
- **RAM:** 16 GB
- **Programming language:** C++

## 4.3 Results

As mentioned in Section 3.3.1, not all combinations of parameters $NP_{MAS}$ and $N$ are feasible. Therefore, in the first experiment, the following feasible combinations as presented in Table 1 were used in our study.

**Table 1: Feasible combinations of agent's number $N$.**

| $NP_{MAS}$ | $N$ | | | | | | |
|---|---|---|---|---|---|---|---|
| | 1 | 5 | 8 | 10 | 16 | 20 | 25 |
| 50 | 50 | 10 | **6.25** | 5 | **3.12** | **2.5** | **2** |
| 100 | 100 | 20 | **12.5** | 10 | **6.25** | 5 | 4 |
| 200 | 200 | 40 | 25 | 20 | **12.5** | 10 | 8 |
| 400 | 400 | 80 | 50 | 40 | 25 | 20 | 16 |

Values in the table present agent's population size $NP$ and must be integer. When the results of division of $NP_{MAS}$ by $N$ is not an integer value, this combination of parameters is declared as infeasible. The infeasible values are presented bold in the table. Value in row $NP_{MAS} = 50$ and column $N = 25$ is also infeasible, because mutation strategy 'DE/rand/1/bin' in DE (Eq. (3)) requires at least 4 individuals in population.

Table 2 shows the overall performance *perf* (Eq. (12)) obtained in the first experiment by the MAS-jDE algorithm for all combinations of $NP_{MAS}$ and $N$. Higher values in table present the better results. Non-valid combinations of parameters are marked with "n/a" in the table. Thus, the last row depicts the average overall performance values according to the number of agents ($N$), where the infeasible values of population sizes were ignored.

The highest performance value $perf = 70.1968$ was achieved, when the grand population size was $NP_{MAS} = 200$ and the MAS consisted of $N = 10$ agents. The last row with average performance results shows, that the MAS-jDE consisting of 8 agents returned the best results. However, this result was based only on two instances. Obviously, when all four overall performance values are taken into consideration, the MAS-jde using 5 agents has shown to be the best.

The results in Table 2 also show the stagnation effect in algorithm, although both features, i.e., the diversity measurements and migration of individuals were activated. Consequently, the worst results were achieved, when the grand population was divided between large number of agents. Performance also worsened, when the number of agents in the MAS with the single population increased.

Results from Table 2 are illustrated in Figure 1, comparing the multi-agent systems of different total population sizes

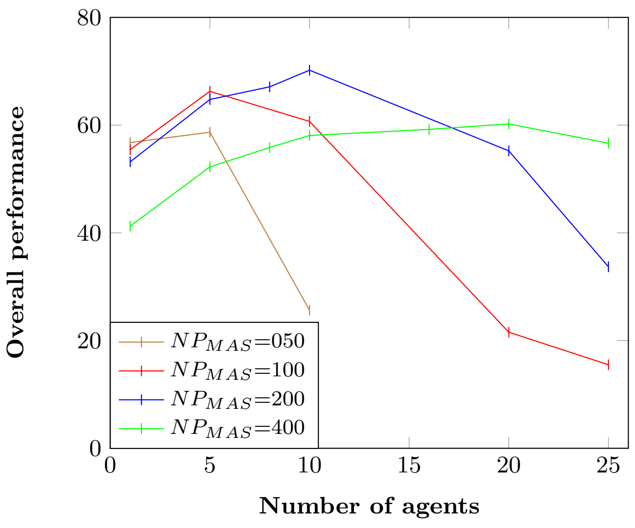


**Figure 1: Comparison of various population sizes**

and different number of agents according to overall performance results of CEC'09 benchmark. It can be observed that the best result of MAS-jDE was obtained by the MAS with the total population size of $NP_{MAS} = 200$. Thus, it is evident that the total population size of $NP_{MAS} = 50$ is inappropriate for building MAS, because of insufficient population size in agent. In general, the MASs with higher total population sizes are more convenient for achieving the best results.

Table 3 presents more detailed results for the best solution of MAS-jDE, with $N = 10$ agents and the total population size of $NP_{MAS} = 200$. The best results were obtained by solving the Ackley's function ($F_5$) for all 7 change types. Good results were achieved for all functions at small-step ($T_1$) and chaotic ($T_4$) change. Algorithm performed very well for functions $F_2$ and $F_4$ for recurrent change with noise ($T_6$) and for function $F_6$ with random change ($T_3$), recurrent change ($T_5$) and recurrent change with noise ($T_6$).

In the second experiment, the solution with highest overall performance *perf* (Eq. (12)) found in the first experiment, was compared to the DOP DE, DOP jDE (i.e., equivalent to single agent MAS-jDE) and the following state-of-the-art algorithms: jDE* [2], CDDE_Ar [5], MLSDO [6] and mSQDE-i [9]. Algorithms DOP DE and DOP jDE are original DE [11] and jDE [1] algorithms that were adopted to solve DOP, by randomly reinitializing population when change in environment is detected. The results for algorithms DOP DE and DOP jDE were contributed using our own implementations of the algorithms, while the results for others were obtained from their respective literature.

The comparative study of mentioned algorithms was made according to the overall performance values *perf* for 49 test cases of CEC'09 benchmark functions. The results are depicted in Table 4 and show that the MAS-jDE algorithm achieved better results than the DOP DE, DOP jDE, jDE* and CDDE_Ar algorithms, while the mSQDE-i and MLSDO

**Table 2: Overall performance by total population sizes and number of agents**

| $NP_{MAS}$ | $N$ | | | | | | |
|---|---|---|---|---|---|---|---|
| | 1 | 5 | 8 | 10 | 16 | 20 | 25 |
| 50 | 56.7574 | 58.6826 | n/a | 25.6366 | n/a | n/a | n/a |
| 100 | 55.4424 | 66.2785 | n/a | 60.6823 | n/a | 21.5483 | 15.5059 |
| 200 | 53.1856 | 64.7585 | 67.1079 | **70.1968** | n/a | 55.2277 | 33.7031 |
| 400 | 41.2382 | 52.2476 | 55.8367 | 58.0758 | 59.2057 | 60.2217 | 56.6458 |
| Avg | 51.6559 | 60.4918 | 61.4723 | 53.6478 | 59.2057 | 45.6659 | 35.2849 |

**Table 3: Algorithm MAS-jDE overall performance for the best solution ($NP_{MAS} = 200$ and $N = 10$)**

| | $F_1(10)$ | $F_1(50)$ | $F_2$ | $F_3$ | $F_4$ | $F_5$ | $F_6$ |
|---|---|---|---|---|---|---|---|
| $T_1$ | 0.0149787 | 0.0148135 | 0.0183885 | 0.0152204 | 0.0190092 | 0.0220642 | 0.0152638 |
| $T_2$ | 0.0141223 | 0.0137885 | 0.00997478 | 0.00335469 | 0.0108478 | 0.0227195 | 0.0156269 |
| $T_3$ | 0.0137606 | 0.0129679 | 0.0145688 | 0.0065958 | 0.013584 | 0.02233 | 0.0170243 |
| $T_4$ | 0.0149597 | 0.0149308 | 0.0142835 | 0.0112827 | 0.0178799 | 0.0229582 | 0.0135977 |
| $T_5$ | 0.0141646 | 0.0146022 | 0.0115689 | 0.00435399 | 0.0114108 | 0.0225239 | 0.0177137 |
| $T_6$ | 0.0136714 | 0.01301 | 0.0165458 | 0.0106462 | 0.0171642 | 0.0231245 | 0.0150692 |
| $T_7$ | 0.0088615 | 0.00938833 | 0.0109534 | 0.00691136 | 0.0112994 | 0.0148324 | 0.0132556 |
| Mark | 0.0945188 | 0.09350123 | 0.09628368 | 0.05836514 | 0.10119530 | 0.1505527 | 0.1075512 |
| Performance (summed the mark obtained for each case and multiplied by 100): 70.1968 | | | | | | | |

algorithms outperformed it.

**Table 4: Overall performance comparison**

| Algorithm | Performance |
|---|---|
| MAS-jDE | 70.1968 |
| DOP DE | 45.4996 |
| DOP jDE | 55.4424 |
| jDE* | 69.7269 |
| CDDE_Ar | 69.4700 |
| MLSDO | **81.2760** |
| mSQDE-i | 77.1458 |

## 5. CONCLUSIONS

In this paper we proposed MAS-jDE algorithm that is a MAS, where agents independently of each other explore the search space using the jDE. To prevent stagnation of the agent's populations, theirs population diversity is measured. If diversity falls bellow threshold, migration mechanism between agents takes place.

Algorithm was tested on CEC'09 benchmark functions for DOP. In line with this, two experiments were conducted. In the first, we compared multi-agent systems with different configurations, where the population sizes and the number of agents were varied. The results were compared according to the benchmark's overall performance measure. The best solution was found when the total population size was 200 and the number of agents was 10. From the results of the experiments we can observe, that although we used diversity measurements and migration of individuals, stagnation of population appears in combinations, where agent's had smaller population sizes.

The results of the MAS-jDE with the best parameter setting from the first experiment were also compared with other state-of-the-art optimization algorithms. The MAS-jDE showed much better results then DOP DE and DOP jDE. Similar, but better results were achieved against jDE* and CDDE_Ar. Similarity between MAS-jDE and JDE* results can be contributed to the fact that both algorithms use the same self-adaptive algorithm to search for a global optimum. Our algorithm was outperformed by MLSDO and mSQDE-i.

Future plan is to study the results of diversity measurements and consequences of migrations of individuals more closely, and add mechanisms like k-means clustering algorithm to spread initial population more equally over search-space. Initial tests show promising results.

## 6. REFERENCES

[1] J. Brest, S. Greiner, B. Bošković, M. Mernik, and V. Žumer. Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. *IEEE Transactions on Evolutionary Computation*, 10(6):646–657, 2006.

[2] J. Brest, A. Zamuda, B. Bošković, M. S. Maučec, and V. Žumer. Dynamic optimization using self-adaptive differential evolution. In *Evolutionary Computation, 2009. CEC'09. IEEE Congress on.*, pages 415–422. IEEE, May 2009.

[3] A. Eiben and J. E. Smith. *Introduction to Evolutionary Computing*. Natural Computing Series. Springer-Verlag Berlin Heidelberg, 1 edition, 2003.

[4] H. Fu, P. R. Lewis, B. Sendhoff, K. Tang, and X. Yao. What are dynamic optimization problems? In *Evolutionary Computation (CEC), 2014 IEEE Congress on*, 2014.

[5] U. Halder, S. Das, and D. Maity. A cluster-based differential evolution algorithm with external archive for optimization in dynamic environments. *IEEE transactions on cybernetics*, 43(3):881–897, June 2013.

[6] J. Lepagnot, A. Nakib, H. Oulhadj, and P. Siarry. A multiple local search algorithm for continuous dynamic optimization. *Journal of Heuristics*, 19(1):35–76, Feb. 2013.

[7] C. Li, S. Yang, T. T. Nguyen, E. L. Yu, X. Yao, Y. Jin, H.-G. Beyer, and P. N. Suganthan. Benchmark generator for cec'2009 competition on dynamic optimization. Technical report, University of Leicester, University of Birmingham, Nanyang Technological University, 2008.

[8] R. W. Morrison. *Designing Evolutionary Algorithms for Dynamic Environments*. Springer-Verlag Berlin Heidelberg, 2004.

[9] P. Novoa-Hernández, C. C. Corona, and D. A. Pelta. Self-adaptive, multipopulation differential evolution in dynamic environments. *Soft Computing*, 17(10):1861–1881, Mar. 2013.

[10] K. V. Price, R. M. Storn, and J. A. Lampinen. *Differential Evolution: A Practical Approach to Global Optimization*, chapter The differential evolution algorithm, pages 37–134. Springer-Verlag Berlin Heidelberg, 2005.

[11] R. Storn and K. Price. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359, Dec. 1997.

[12] G. Weiss, editor. *Multiagent Systems*. MIT Press, 2nd edition, 2013.

[13] M. Yang, C. Li, Z. Cai, and J. Guan. Differential evolution with auto-enhanced population diversity. *IEEE transactions on cybernetics*, 45(2):302–315, 2015.

# Aktivni BMS sistem za litij-ionske celice

Primož Bencak
Univerza v Mariboru
Fakulteta za strojništvo
Smetanova 17, Maribor
primoz.bencak@
student.um.si

Dušan Fister
Univerza v Mariboru
Fakulteta za strojništvo
Smetanova 17, Maribor
dusan.fister@
student.um.si

## POVZETEK

Litij-ionske baterije so s svojimi odličnimi lastnostmi skoraj povsem nadomestile ostale tipe baterij. Zaradi visoke nazivne napetosti so primerne za zaporedno vezavo v baterijske sklope, vendar se z naraščanjem števila celic v paketu zvišuje kompleksnost sistema ter posledično možnost napačnega delovanja. V ta namen se za nadzor nad baterijskim paketom in izravnavo napetosti na posameznih celicah, ki so posledica kemijske sestave, uporabljajo baterijski nadzorni sistemi (angl. battery management system, krajše BMS). V tem članku je predstavljena izdelava prototipnega BMS sistema, opisana je simulacija sistema v simulacijskem okolju MATLAB/Simulink, kakor tudi fizična izdelava elektronskega vezja. Določena je nadzorna in regulacijska funkcija blokovno programiranega digitalnega signalnega krmilnika TMS320F28335. Predstavljeni so rezultati nadzorovanja realnega delovanja aktivnega BMS sistema na treh zaporedno vezanih litij-ionskih celicah, ki kažejo vzpodbudno delovanje zasnovanega BMS sistema.

## Ključne besede

digitalni signalni krmilnik, zaporni pretvornik, balansiranje, litij-ionske baterije, MATLAB/Simulink

## 1. UVOD

Izenačevanje nivoja napetosti (balansiranje) zaporedno vezanih baterijskih celic (paketa) pomembno vpliva na odražanja celotnega napajanega sistema. Za učinkovito odpravo motenj izhodne napetosti so želene minimalne ali nične razlike med posameznimi litij-ionskimi (li-ion) celicami. V pričujočem članku prikazujemo balansiranje treh zaporedno vezanih Li-ion celic. razložen je princip uporabe dveh zapornih pretvornikov za potrebe balansiranja. Delovanje smo najprej preizkusili v simulacijskem okolju, nato pa še v realnem.

Ideja o uporabi prihaja od avtorjev Hoque-a, Hannan-a in Mohamed-a, ki so v [7] predvidili uporabo dveh zapornih (angl. *flyback*) pretvornikov za balansiranje posameznih celic v obe smeri. Bonfiglio in Roessler sta v [5] izbrala podobno zasnovo - zaporni pretvornik, vendar sta idejo dvosmernih stikal, ki izbirajo med posameznimi celicami, nadgradila z uporabo transformatorskih navitij.

Predstavljen članek je rezultat združitve dobrih lastnosti obeh idej. Iz prvega smo vzeli zasnovo strojne opreme, medtem ko iz drugega programsko opremo. Izdelali smo prototip aktivnega BMS sistema, ki je sposoben nadzorovati tri nezaščitene li-ion celice med delovanjem. Ob napačnem delovanju je zahtevan trenuten izklop povezave med BMS in porabnikom ter signaliziranje napake (režim napake).

Struktura članka v nadaljevanju je naslednja: drugo poglavje govori o simulacijah BMS sistema v MATLAB/Simulink, tretje poglavje opisuje postopek načrtovanja fizičnega sistema, četrto poglavje pa blokovno programiranje omenjenega sistema. Članek je zaključen s petim poglavjem, kjer so predstavljene možne izboljšave našega dela v prihodnje.

## 2. BATERIJSKI NADZORNI SISTEM

Baterijski nadzorni sistem zagotavlja varno in učinkovito delovanje baterijskega sistema. Naslednje alineje predstavljajo pogoje za varno in učinkovito delovanje z li-ion baterijami:

- spremljanje napetosti na posamezni celici,
- merjenje pritekajočega in odtekajočega toka,
- nadzor temperature,
- določanje stanja napolnjenosti ('SOC – *state of charge*'),
- določanje dejanskega stanja paketa ('SOH - *state of health*').

Zbrani podatki so posredovani uporabniku, zagotovljeni pa so tudi ukrepi ob delovanju izven meja. Poleg spremljanja omenjenih parametrov, BMS skrbi tudi za enakomerno polnjenje in praznjenje celic, saj nobena izmed njih ne sme preseči maksimalne, ki znaša približno 4.25 V, ali minimalne napetosti (približno 2.5 V). Postopku izravnavanja napetosti posameznih celic pravimo balansiranje [8].

Potreba po balansiranju se pojavi zaradi notranje, oz. kemijske strukture li-ion baterij. Ta je odražena s karakteristiko polnjenja in praznjenja. Pogosto se zgodi, da se nekatere celice polnijo (ali praznijo) nekoliko hitreje, kar povzroča

DOI: https://doi.org/10.26493/978-961-7023-40-4.43-48

neželene razlike med napetostmi. Za krajši čas delovanja to ni problem, se pa lahko ta pojavi po daljšem času obratovanja pri intenzivni rabi. Možne so poškodbe celic, zato se zmanjša tudi izkoristek paketa [3, 6].

## 2.1 Tipi baterijskih sistemov

V osnovi delimo BMS sisteme v dve skupini: na *aktivne*, kjer ”odvečno” energijo posamezne celice prerazporedimo nazaj v sistem in *pasivne*, kjer ”odvečno” energijo posamezne celice potrošimo v obliki toplote na uporih. Pasivni BMS sistemi so s stališča enostavne izvedbe privlačni predvsem za cenejše projekte, kjer kapacitete baterij niso velike. Treba je zagotoviti ustrezno hlajenje, saj se energija pretvarja v toploto, zato se uporaba slednjih z ekonomskega stališča ne obrestuje. Po drugi strani so aktivni BMS sistemi precej bolj kompleksni, a je njihov izkoristek mnogo višji, saj se višek energije prerazporedi nazaj v sistem. Četudi ima ta princip določene izgube, so te dosti manjše. Dodatna prednost aktivnih BMS sistemov je, da omogočajo dvosmerno balansiranje: iz celice v paket in iz paketa v celico [4].

Struktura naše naprave sledi članku [7]. Zaporni pretvornik je izbran zaradi svoje enostavne izvedbe, majhnih induktivnih elementov ter galvanske ločitve. Smer pretoka energije določa stikalni del, ki omogoča dvosmerni pretok energije. Merilni del opravlja meritve toka in napetosti na posameznih celicah, medtem ko nadzor nad preklopi stikalnih elementov in regulacijo sistema vrši nadzorna funkcija, ki je izvedena na krmilniku. V omenjenem članku je predvidena raba osmih zaporedno vezanih li-ion celic, ki tvorijo približno 33.6 V napetosti. V naslednjem poglavju je prikazana simulacija delovanja, izvedena v orodju MATLAB/Simulink, ki omogoča modeliranje elementov močnostne elektronike s pomočjo knjižnice SimPowerSystems [4].

## 3. SIMULACIJA BMS SISTEMA

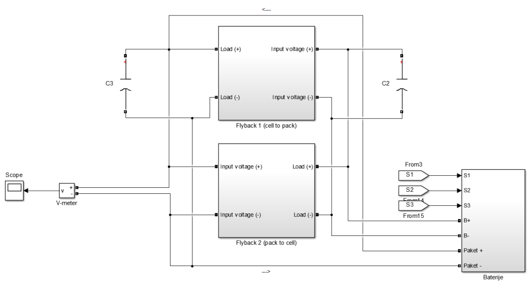
Za uspešno simulacijo BMS sistema je treba implementirati naslednje sestavne dele:

- zaporni pretvornik,
- algoritem balansiranja in
- model li-ion celic.

Za dvosmerni pretok energije sta potrebna dva zaporna pretvornika, katerima določimo maksimalno vhodno in izhodno napetost ter maksimalni tok. Algoritem balansiranja je izveden v diagramu prehajanja stanj (angl. *Stateflow*), medtem ko je model li-ion celic predstavljen z zaporedno vezanim RC-členom. R predstavlja notranjo upornost baterijske celice, C pa kapacitivnost energijske posode, kamor se shranjuje ali izteka naboj. Kapacitivnost celice ocenimo po naslednji enačbi:

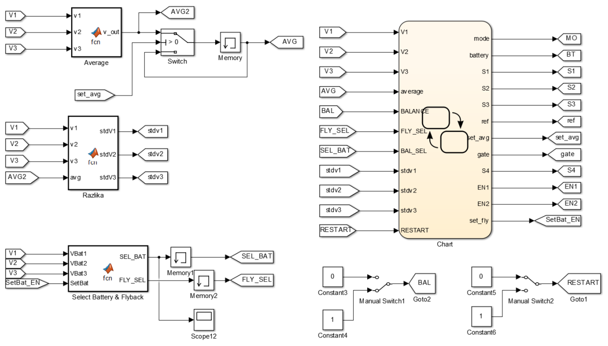
$$C = \frac{Q}{V} = \frac{75 \cdot 10^{-3} \cdot 3600 \text{ As}}{4.2 \text{ V}} = 64.27 \text{ F} \qquad (1)$$

Zvezni (CCM) način delovanja pretvornika je dosegljiv, ko tok skozi dušilko (transformator) nikoli ne pade na vrednost nič. V skladu s tem je treba izračunati veličine komponent, kot npr. [11] in [13]. Določiti je treba komponente zapornega pretvornika: dušilko (flyback transformator), diodo, vhodni in izhodni kondenzator ter MOSFET tranzistor kot stikalni element. Parametre izračunanih komponent smo vnesli v simulacijsko shemo ter na podlagi analitične metode povprečenja v prostoru stanj določili parametre regulatorja [1, 11]. Baterijski paket oz. celice balansiramo s konstantnim tokom, zato je uporabljena kaskadna vezava PI-napetostnega regulatorja in P-tokovnega regulatorja. Stikalni del, ki služi izbiri celic, ki jih je treba balansirati, je bil modeliran z blokom *Ideal Switch*, ki je bil na eni strani povezan z zapornim pretvornikom, na drugi pa z baterijskim paketom. Shemo simulacije prikazuje slika 1.



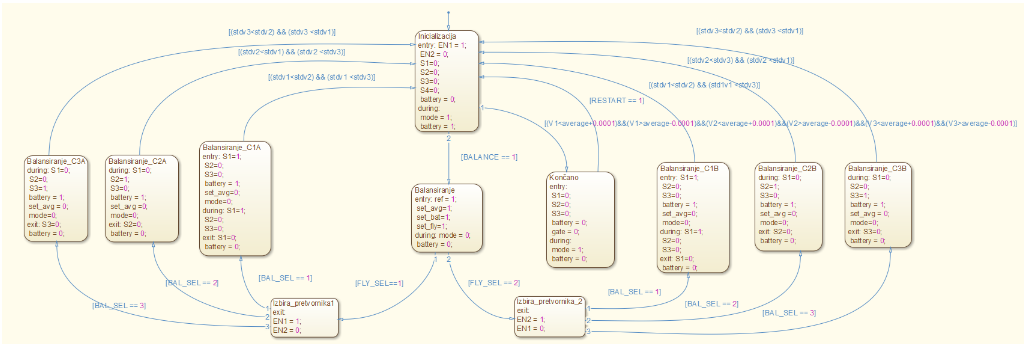
Slika 1: Način povezave zapornih pretvornikov.

Čeprav uporaba člena prvega reda (RC člen) za modeliranje li-ion celice ne predstavlja točnega modela, je približek dovolj dober, da predvidimo obnašanje sistema. Preklapljanje stikal, izbiro pretvornika in način regulacije določa nadzorna funkcija, izvedena z MATLAB-ovo funkcijo in diagramom prehajanja stanj (sliki 2 in 3).



Slika 2: Izvedba algoritma balansiranja s Stateflow diagramom in MATLAB funkcijo.

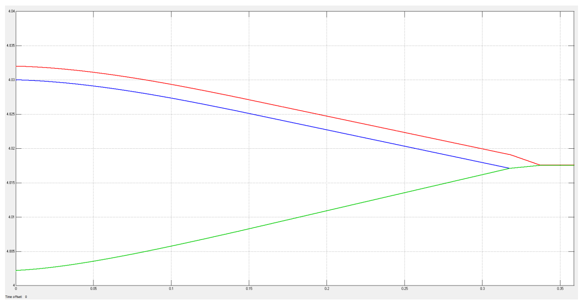
Diagram prehajanja stanj ob vnešenih pogojih zamenjuje režime delovanja, npr. polnenje druge celice in praznjenje tretje celice. V vsakem trenutku mora biti definiran vsaj en režim, zato se algoritem ciklično ponavlja. Naslednje podpoglavje predstavlja algoritem balansiranja.



Slika 3: Diagram prehajanja stanj.

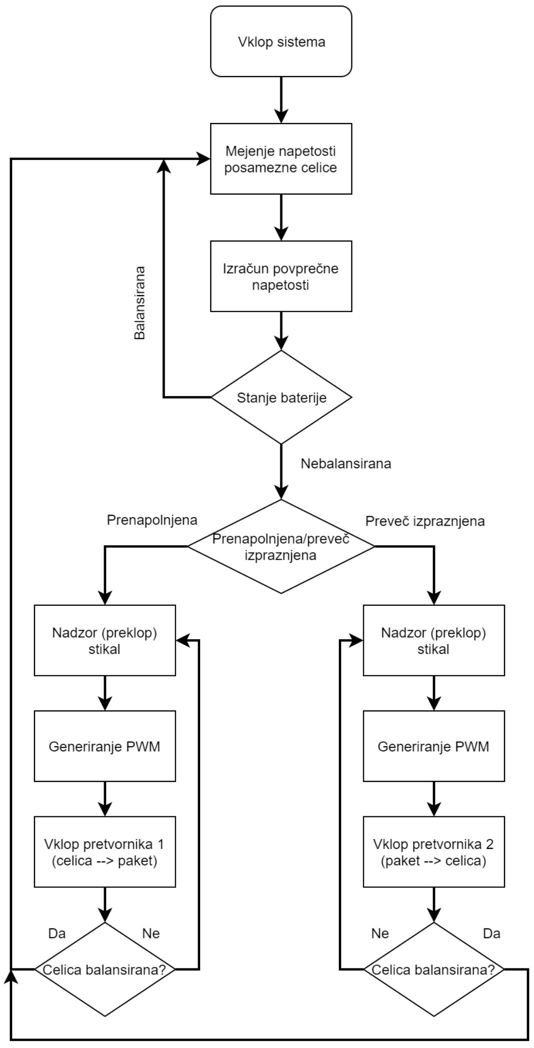
## 3.1 Algoritem balansiranja

Algoritem balansiranja v režimu inicializacije preveri napetosti na posamezni celici. Iz slednjih se izračuna povprečje. Iz razlike povprečja in dejanskih napetosti je določena celica, ki najbolj odstopa od povprečja. V kolikor je odstopanje pozitivno, bo treba celico nekoliko izprazniti in obratno - če je razlika negativna, bo treba celico napolniti. Polnjenje in praznenje poteka z interakcijo vseh celic, to pomeni, da če eno izmed celic polnimo, ostali dve praznimo [5, 7]. V praksi je treba algoritem prilagoditi dejanskim zmožnostim obeh pretvornikov. Možno je, da lahko kateri izmed pretvornikov zaradi izbranih komponent, v eni smeri celico ali paket polni hitreje. Tabela 1 in slika 4 predstavljata simulacijski rezultat, medtem ko slika 5 diagram poteka algoritma balansiranja, primeren za realno izvedbo (oznaka * pomeni balansiranje iz paketa v celico, ** pa iz celice v paket).



Slika 4: Rezultat simulacije balansiranja.

Iz simulacije balansiranja je razvidno, da vse tri napetosti celic konvergirajo proti skupnemu povprečju, zato lahko iz simulacije potrdimo pravilno delovanje algoritma balansiranja. Preseneča hitra časovna konstanta izenačevanja, ki znaša 350 ms. Pred testiranji tako hitrega reagiranja nismo pričakovali.



Slika 5: Diagram poteka algoritma balansiranja.

## 4. IZVEDBA REALNEGA ELEKTRONSKEGA VEZJA

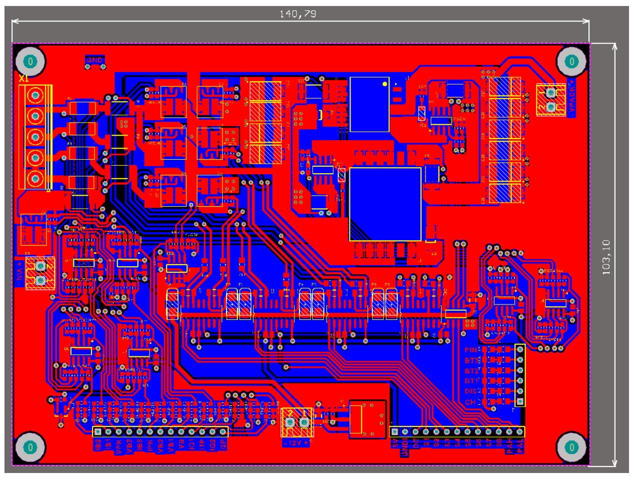
Teoretično delovanje je kakopak treba preveriti tudi v praksi, zato smo izbrali ustrezne elektronske komponente ter načrtali elektronsko vezje BMS sistema. Posebna zahvala v tej fazi velja dr. Mitji Truntiču, ki je pomagal pri načrtovanju.

Za stikalni del so bili zaradi nizke prevodne upornosti izbrani dvojni MOSFET tranzistorji, ki omogočajo dvosmerni pretok. Meritve tokov so izvedene s shunt upori, ki informacijo o toku pretvarjajo v napetost (slednjo lahko izmerimo z AD pretvornikom). Za prilagojevanje nivojev informacije napetosti in tokov so uporabljeni operacijski ojačevalniki v diferenčni vezavi z uporovnim delilnikom. Baterijske celice pred previsokim tokom varujejo SMD varovalke, preko doda-

Tabela 1: Tabela parametrov

| Celica | Začetna napetost | Končna napetost | Kapaciteta baterije | Čas izenačevanja | Tok balansiranja |
|---|---|---|---|---|---|
| Modra | 4.030 V | 4.0175 V | 75 mAh, 0.05 Ω | 350 ms | 4.5*/2.2** A konst. |
| Rdeča | 4.032 V | 4.0175 V | 75 mAh, 0.05 Ω | 350 ms | 4.5*/2.2** A konst. |
| Zelena | 4.020 V | 4.0175 V | 75 mAh, 0.05 Ω | 350 ms | 4.5*/2.2** A konst. |

tnega MOSFET stikala pa jih je možno direktno integrirati s polnilcem baterij. Močnostni in signalni del ločujejo gonilniki (angl. *driver*), ki sistema galvansko ločujejo, da se motnje ne prenašajo iz enega v drug sistem. Pri načrtovanju stikalnih pretvornikov je posebno pozornost treba nameniti načrtovanju čim krajših in širših povezav, ki so na fizično ločeni strani. Močnostni in signalni del se sklepata le v eni točki, na digitalnem signalnem krmilniku (DSK), kar pomeni da so signali neobremenjeni z motnjami. Celotna slika povzetih besed je ilustrirana na sliki 6.

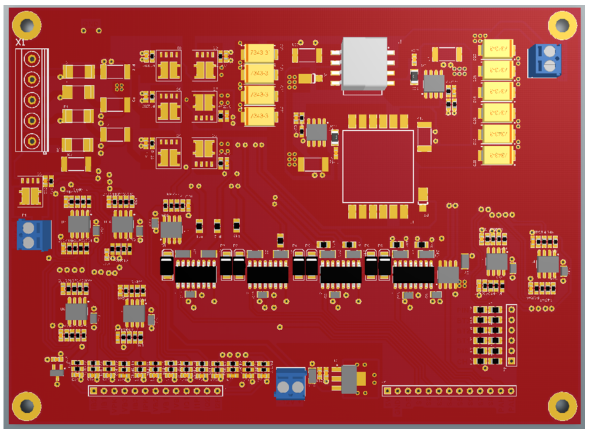


Slika 6: Elektronsko vezje narisano v programu Altium Designer.

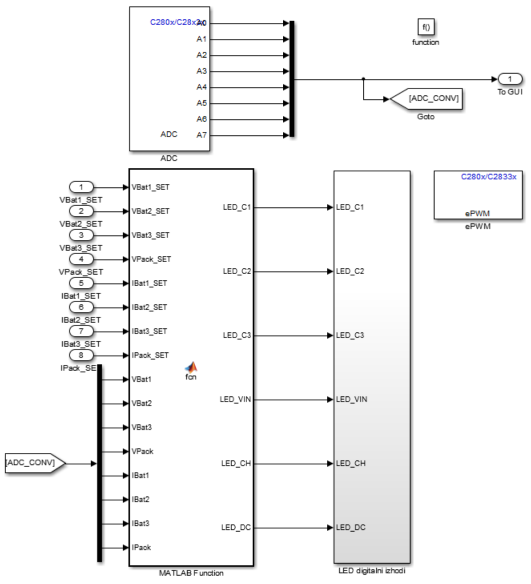
Načrtovano vezje smo izdelali s strojem za izdelavo prototipnih ploščic Protomat S63. Zaradi velikega števila vij (angl. via), tj. *povezav med spodnjim in zgornjim slojem*, smo opravili galvanizacijo ter ročno prispajkali vseh 237 komponent, prikazanih na sliki 7 [2, 9, 10, 12]. Sledil je osnovni električni test napajanja in delovanja prožilnih signalov za preklop dvojnih MOSFET stikal. Zaradi časovne stiske smo se odločili, da bo naš BMS sistem zaenkrat opravljal le funkcijo nadzorovanja baterijskega paketa (tj. merjenje napetosti in toka), funkcijo balansiranja pa bo dodana kasneje. Za omogočitev funkcionalnosti komponent jih je treba povezati s DSK-jem, o čemer priča naslednje podpoglavje.

## 4.1 Programiranje digitalnega signalnega krmilnika

Izdelano elektronsko vezje, ki ga prikazuje slika 9, je treba nadzorovati z digitalnim signalnim krmilnikom Texas Instruments TMS320F28335, ki ga programiramo s pomočjo blokov v simulacijskem orodju MATLAB/Simulink (slika 8). Zaenkrat je ta, zaradi enostavnejše izvedbe, pritrjen na zunanjo ploščico. Vsekakor bi bilo, za odpravo induciranja motenj v sistem, v prihodnosti nujno razmisliti o integraciji zu-

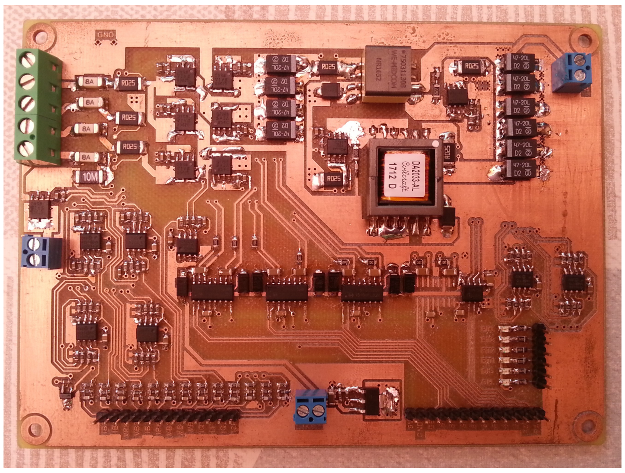

Slika 7: 3D pogled elektronskega vezja.

nanje ploščice s krmilnikom na elektronsko ploščo. Slabost, ki se ob tem pojavlja, je omejevanje na en tip krmilnika.

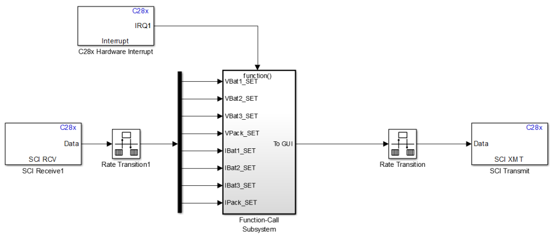


Slika 8: Sestavljanje sheme nadzorne funkcije krmilnika.

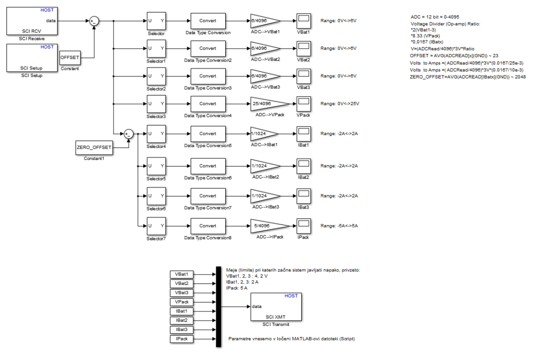
Zaradi enostavnejše izvedbe je DSK kot nadzorni element (slika 10) trenutno pritrjen na dodatno ploščico. Vsekakor bi bilo, zaradi odprave induciranja motenj v sistem, v prihodnosti nujno razmisliti o uporabi krmilnika na elektronski plošči. Na ta način se izognemo dodatnim povezavam ter prihranimo na prostoru, po drugi strani pa postanemo omejeni na en krmilnik.


Slika 9: Končano elektronsko vezje.

Programiranje DSK-ja poteka podobno kot sestavljanje sheme, le da sedaj uporabljamo bloke, namenjene točno določeni skupini DSK-jev (C2000). Uporabljene so deterministične prekinitve, ki služijo vnaprej določenemu, periodičnemu programu. Periodično se izvaja proženje AD pretvornika in enote e-PWM, manipulacija z digitalnimi izhodi ter serijska povezava podatkov preko protokola SCI (opazovanje podatkov je omogočeno v ločenem Simulink modelu, slika 11). Simulink prispele podatke z nekaj sekundnim zamikom osvežuje in jih izrisuje v graf. Podroben postopek aktivnosti krmilnika, oz. diagrama prehajanja stanj, predstavlja algoritem 1.



Slika 10: Krmilnik kot nadzorni element.



Slika 11: Serijska povezava SCI.

**Algoritem 1** Določevanje celic in pretvornika

```
ADCRead(VBat[X]); //Preberi vrednosti AD pretvornika
MinMax(VBat[X]); //Razvrsti napetosti od najmanjše do največje
Average(VBat[X]); //Izračunaj povprečje napetosti
if (Max-Avg)>(Avg-Min) then
  Flyback = 1;//Izbira pretvornika: bal. celica–>paket
else
  Flyback = 2;//Izbira pretvornika: bal. paket–>celica
end if
if (Flyback == 1)&(Max == VBat1) then
  SWA=1; SWB=1;//Izbira stikal prve celice
else if (Flyback == 1) &(Max == VBat2) then
  SWA=2; SWB=2;
else if (Flyback == 1)&(Max==VBat3) then
  SWA=3; SWB=3;
end if
if (Flyback == 2)&(Min == VBat1) then
  SWA=1; SWB=1;
else if (Flyback == 2) &(Min == VBat2) then
  SWA=2; SWB=2;
else if (Flyback == 2)&(Min==VBat3) then
  SWA=3; SWB=3;
end if
Balance.Start();//Začni balansiranje
repeat
  Balance ();//Balansiraj določeno celico/paket, dokler izbrana celica/paket najbolj izstopa od povprečja
until (VBat[X]-Avg)>(VBat[Y]-Avg)&
(VBat[Z]-Avg)>(VBat[Y]-Avg)
if (VBat[X] == VBat[Y] == VBat[Z]) then
  Balance.End()//Ko so napetosti dovolj blizu, končaj balansiranje
end if
```

## 4.2 Rezultati delovanja realnega aktivnega BMS sistema

Vezje aktivnega BMS sistema, ki ga prikazuje slika 9, je zaenkrat sposobno opravljati le nadzor nad baterijskim paketom, kar zajema meritve napetosti in toka na posameznih celicah, ne pa tudi aktivnega balansiranja baterijskih celic. Meritve napetosti in toka so brez dodatnega filtriranja stabilne, treba pa je bilo določiti ustrezne korekcijske faktorje in nastaviti zamik (angl. *offset*) AD pretvornika. S tem smo dosegli kakovostnejše podatke, zato je BMS sistem deloval učinkoviteje.

Na podlagi zbranih podatkov lahko krmilnik z diagramom prehajanja stanj odloča o tem, ali je baterijski sistem v območju normalnega delovanja ali v režimu napake. Podatki o meritvah so uporabniku na voljo, zato lahko ta nastavlja prag (meje) javljanja napake za posamezno celico/paket in izklopa povezave med polnilcem in baterijskim paketom v primeru nadzorovanega polnjenja.

Dodatno je vezje elektronskega BMS sistema opremljeno z zunanjim priključkom za polnilec, ki ga lahko preko elektronskega stikala (MOSFET tranzistorja) vključimo in izključimo. Na ta način je možno izvajati nadzorovano polnjenje z balansiranjem.

## 5. SKLEP

Ob eksperimentalnem testiranju načrtanega vezja smo ugotovili, da bi bilo enega izmed priključkov treba zamenjati. Prav tako bi za MOSFET stikala morali uporabiti povečane bakrene priključke, saj smo imeli nekaj težav pri spajkanju. Zaradi velikega števila komponent smo prav tako imeli nekaj težav s slabimi povezavami, ki pa smo jih po prvem testu vezja odpravili. Na mestih priklopa posameznih celic bi lahko namestili temperaturne senzorje (NTK upore ali termometre na osnovi PN spoja), ki bi dodatno varovali paket pred morebitnim napačnim delovanjem (a s tem povečevali kompleksnost).

Elektronsko vezje bi lahko vgradili v plastično ali kovinsko ohišje, ki bi onemogočilo nezaželjene stike s komponentami, bilo bi pa tudi treba poskrbeti za primeren dotok hladnega zraka, saj se nekatere komponente (tranzistorji) ob daljšem delovanju precej segrevajo.

Z izdelanim vezjem smo zadovoljni, saj naprava zaenkrat služi namenu. Rezultati trdega dela se bodo v celoti pokazali šele, ko bo vezje sposobno opravljati nadzor in balansiranje celice.

## ZAHVALA

Iskreno se zahvaljujemo profesorjema doc. dr. Miranu Rodiču in red. prof. dr. Miro Milanoviču ter asistentu dr. Mitji Truntiču s Fakultete za elektrotehniko, računalništvo in informatiko v Mariboru za pomoč pri izvedbi simulacij in načrtovanju elektronskega vezja.

## REFERENCE

[1] S. Bacha, I. Munteanu, A. I. Bratcu, et al. Power electronic converters modeling and control. *Advanced textbooks in control and signal processing*, 454:454, 2014.

[2] Battery University. Basic to Advanced Battery Information from Battery University. Dostopno na: http://batteryuniversity.com/. Dostopano: 1.5.2017.

[3] Battery University. BU-908: Battery Management System. Dostopno na: http://batteryuniversity.com/learn/article/how_to_monitor_a_battery. Dostopano: 29.8.2017.

[4] P. Bencak. *Aktivni BMS sistem za Li-ion akumulatorske baterije*. Diplomsko delo. Univerza v Mariboru, Fakulteta za strojništvo, 2017.

[5] C. Bonfiglio and W. Roessler. A cost optimized battery management system with active cell balancing for lithium ion battery stacks. In *Vehicle Power and Propulsion Conference, 2009. VPPC'09. IEEE*, pages 304–309. IEEE, 2009.

[6] Electronic Design. A Look Inside Battery-Management Systems. Dostopno na: http://www.electronicdesign.com/power/lookinside-batterymanagementsystems. Dostopano: 29.8.2017.

[7] M. Hoque, M. Hannan, and A. Mohamed. Voltage equalization for series connected lithium-ion battery cells. In *Smart Instrumentation, Measurement and Applications (ICSIMA), 2015 IEEE 3rd International Conference on*, pages 1–6. IEEE, 2015.

[8] C.-H. Kim, M.-Y. Kim, and G.-W. Moon. A modularized charge equalizer using a battery monitoring IC for seriesconnected Liion battery strings in electric vehicles. *IEEE Transactions on Power Electronics*, 28(8):3779–3787, 2013.

[9] Linear Technology. LT3575 - Isolated Flyback Converter without an Opto-Coupler. Dostopno na: http://www.linear.com/product/LT3575. Dostopano: 1.5.2017.

[10] Linear Technology. LTC3300-1 - High Efficiency Bidirectional Multicell Battery Balancer. Dostopno na: http://www.linear.com/product/LTC3300-1. Dostopano: 24.7.2017.

[11] M. Milanovič. *Močnostna elektronika*. Fakulteta za elektrotehniko, računalništvo in informatiko, 2010.

[12] TechWeb. Designing Isolated Flyback Converter Circuits: Transformer Design. Dostopno na: http://micro.rohm.com/en/techweb/knowledge/acdc/acdc_pwm/ Dostopano: 1.5.2017.

[13] Texas Instruments. Power Topologies Handbook Analog & MixedSignal. Dostopno na: https://www.ti.com/seclit/ug/slyu036/slyu036.pdf. Dostopano: 24.7.2017.

# Multiset representation of objects in information retrieval systems

Mikita Akulich[1]
89172019@student.upr.si

Matjaž Krnc[1,2]
matjaz.krnc@sbg.ac.at

Iztok Savnik[1]
iztok.savnik@famnit.upr.si

Riste Škrekovski[1,3,4]
riste.skrekovski@famnit.upr.si

[1] University of Primorska, FAMNIT, Koper, Slovenia
[2] University of Salzburg, Department of Efficient Algorithms, Salzburg, Austria
[3] Faculty of Information Studies, Novo Mesto, Slovenia
[4] University of Ljubljana, FMF, Ljubljana, Slovenia

## ABSTRACT

In this paper we present *multiset-trie* – a novel data structure which operates on objects represented as multisets. The multiset-trie is a search-tree-based data structure with properties similar to those of a trie. In particular, we efficiently implement the standard search tree operations together with the special set containment operations, i.e. subset and superset queries in the context of multisets. These are called *submultiset* and *supermultiset*, respectively, and are used for implementation of various queries that can be performed on multisets in a multiset-trie. The corresponding running times of the developed functions are mathematically and experimentally analyzed. One of the most important queries is the search of the nearest neighbor given an input object. The nearest neighbor search of a multiset-trie makes it a good alternative for the index data structures that are used in information retrieval systems. In particular, our research is focused on the application of the multiset-trie to full-text search systems.

## Keywords

multiset, trie, multiset-trie, information retrieval, inverted index, full-text search

## 1. INTRODUCTION

Information retrieval (IR) systems are most commonly used for searching a text-based content such as text documents in a database. Such IR systems are called *full-text search systems*. The application of the full-text search techniques on the database almost always requires an index for a good performance of the queries. Indexes narrow down the search using pre-generated meta data constructed from the data in the database.

In IR most systems use the concept of an inverted index to achieve full-text indexing of a database. The so-called dictionary of the inverted index can be organized in different ways in order to meet the required types of queries and specification of data. In our research we will be focused on search trees [2, 4, 5, 7].

The proposed data structure *multiset-trie* can be used as an alternative implementation of the dictionary in an inverted index. It is an extension of the *set-trie* data structure proposed by Savnik [6]. Set-trie is used for storing and fast retrieval of objects represented as sets and provides the nearest-neighbor search by implementing methods that perform set-containment queries. Multiset-trie extends the abilities of set-trie and provides support for storing and retrieving objects that can be represented as both sets and multisets.

## 2. MULTISET-TRIE DATA STRUCTURE

### 2.1 Description

Let $\Sigma$ be a set of distinct symbols that define an alphabet and let $\sigma$ be the cardinality of $\Sigma$. Define $\phi$ to be a bijective mapping $\phi : \Sigma \to I$, where $I = \{1, 2, 3, \dots, \sigma\}$. In this way, we obtain an indexing of elements from the alphabet $\Sigma$, so we can work directly with integers rather then with specific symbols from $\Sigma$.

The multiset-trie $\mathcal{M}$ is an $n$-ary tree based data structure with the properties of trie. A node in multiset-trie always has out-degree $n$. Some of the child nodes may be *Null*, but at most $n - 1$ of them. All the children of a node, including the *Null* children, are uniquely labeled from left to right with labels $c_j$, where $j \in \{0, 1, \dots, n - 1\}$.

The height of a multiset-trie is always $\sigma + 1$ if the structure is not empty. Levels in multiset-trie are enumerated by their height, i.e. a level $L_i$ has height $i$, where $L_1$ is the root level. The levels $L_i$ and symbols from $\Sigma$ are related as follows. For $i \in \{1, 2, \dots, \sigma\}$, a level $L_i$ represents a symbol $s \in \Sigma$, such that $\phi^{-1}(i) = s$. The last level $L_{\sigma+1}$ does not represent any symbol and is named *leaf level* (*LL* for short).

Consider two nodes $u, v$ in a multiset-trie at levels $L_i, L_{i+1}$

respectively. Let a node $u$ be a parent node of a node $v$. Suppose that a child node $v$ is not *Null* and has a label $c_j$. Then the *path* $u \to v$ represents a symbol $\phi^{-1}(i) \in \Sigma$ with multiplicity $j$. Such a transition $u \to v$ is called a *path of length* 1 and is allowed if and only if a node $v$ is not *Null* and $u$ is a parent node of a node $v$.

We define a *complete path* to be the path of length $\sigma$ in a multiset-trie with the end points at root node (the 1st level) and $LL$. Thus, a multiset $m$ is inserted into a multiset-trie if and only if there exists a complete path in a multiset-trie that corresponds to $m$. Note that every complete path in a multiset-trie is unique. Therefore, the multisets that share a common prefix in a multiset-trie can have a common path of length at most $\sigma - 1$. Thus, any multiset $m$ that is composed of symbols from $\Sigma$ with maximum multiplicity not greater than $n-1$ can be represented by a complete path in a multiset-trie.

Let us have an example of a multiset-trie data structure, where $\sigma = 2, \Sigma = I = \{1, 2\}, n = 3$ and the mapping $\phi$ is an identity mapping. The figure 1 presents an example of the multiset-trie, where *Null* children are omitted. Let a pair

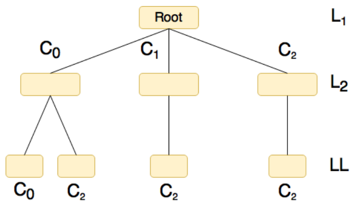


**Figure 1: Example of multiset-trie structure.**

$(L_i, c_j)$ represents a node with label $c_j$ at a level $L_i$. The pair $(L_1, c_j)$ is equivalent to $(L_1, root)$, for every $j$. According to the figure 1 we can extract inserted multisets as follows:

$$\begin{aligned}
(L_1, root) \to (L_2, c_0) \to (LL, c_0) &\equiv \{1^0, 2^0\} = \emptyset \\
(L_1, root) \to (L_2, c_0) \to (LL, c_2) &\equiv \{1^0, 2^2\} = \{2, 2\} \\
(L_1, root) \to (L_2, c_1) \to (LL, c_2) &\equiv \{1^1, 2^2\} = \{1, 2, 2\} \\
(L_1, root) \to (L_2, c_2) \to (LL, c_2) &\equiv \{1^2, 2^2\} = \{1, 1, 2, 2\}
\end{aligned}$$

where $e^k$ represents an element $e$ with multiplicity $k$.

## 2.2 Operations

*Basic tree operations.* The procedure INSERT($\mathcal{M}$, $m$) inserts a new instance $m$ of type Multiset into multiset-trie $\mathcal{M}$. If the complete path already exists, then procedure leaves the structure unchanged. Otherwise it extends partially existing or creates a new complete path. The procedure does not return any result.

The function SEARCH($\mathcal{M}$, $m$) checks if the complete path corresponding to a given multiset $m$ exists in the structure $\mathcal{M}$. The function returns true if the multiset $m$ exists in $\mathcal{M}$, and returns false otherwise.

The function DELETE($\mathcal{M}$, $m$) searches for the complete path that corresponds to $m$ in order to remove it. If the path can not be found, the function immediately returns false. During search, the function collects additional data about common prefixes. When search is completed, the function removes the branch preserving common prefixes, and returns true.

*Sub- and super-multiset existence.* The SUBMSETEXISTENCE($\mathcal{M}$, $m$) function checks if there exists a multiset $x$ in $\mathcal{M}$, that satisfies the condition $x \subseteq m$. The function starts with searching for an exact match $x = m$ in $\mathcal{M}$, since $m \subseteq m$ by definition of submultiset inclusion. If an exact match is not found in $\mathcal{M}$, the function uses multiset-trie to find the closest (the largest) submultiset of $m$ in $\mathcal{M}$ by decreasing multiplicity of elements in $m$. At every level the function tries to proceed with the largest possible multiplicity of an element that is provided by $m$. However, when the function reaches some level where it meets a *Null* node and can not go further using path provided by $m$, it decreases the multiplicity of an element that corresponds to a current level. Thus, the function can decrease multiplicity of an element or eventually skip it in order to find the closest $x \subseteq m$.

The function SUPERMSETEXISTENCE($\mathcal{M}$, $m$) checks if there exists supermultiset $x$ of a given multiset $m$ in $\mathcal{M}$. By analogy to the function SUBMSETEXISTENCE, the function SUPERMSETEXISTENCE starts by searching for an exact match $x = m$ in $\mathcal{M}$. If an exact match is not found in $\mathcal{M}$, the function searches for the closest (the smallest) supermultiset $x$ of $m$ in $\mathcal{M}$, in this case, by increasing multiplicity of elements in $m$.

*Get all sub- and super-multisets.* The algorithms for functions GETALLSUBMSETS and GETALLSUPERMSETS are based entirely on algorithms for SUBMSETEXISTENCE and SUPERMSETEXISTENCE functions that do not terminate on the first existing sub/supermultiset, but store the results and continue procedure until all existing sub/supermultisets in $\mathcal{M}$ are found and stored.

## 2.3 Analysis of the multiset-trie

By the design of the multiset-trie, it is easy to see that the functions SEARCH, DELETE and the procedure INSERT have complexity of $O(\sigma)$. Because $\sigma$ is defined when the structure is initialized and does not depend on the user input afterwards, the asymptotic complexity of the functions SEARCH, DELETE and the procedure INSERT is $O(1)$. Nonetheless, in the general case the complexity is $O(\sigma)$.

In what follows, we focus on analysis of the more involved functions which correspond to a multiset-containment queries: SUBMSETEXISTENCE, SUPERMSETEXISTENCE, GETALLSUBMSETS and GETALLSUPERMSETS.

*Mathematical model.* We start with the basics of our mathematical model. Let $\Sigma$ be an alphabet of cardinality $\sigma$, such that $\Sigma = \{1, 2, \ldots, \sigma\}$. Define $N$ to be the power multiset of $\Sigma$ with respect to $n$, where $n$ is the maximal degree of a node in multiset-trie. Thus, the number of multisets in a complete multiset-trie is $|N| = n^\sigma$. Let $M$ be a collection of multisets inserted into multiset-trie $\mathcal{M}$. Our model assumes that all the inserted multisets from $M$ are chosen with the same probability $p$ from $N$, for some $p \in (0, 1)$.

Let $\xi_1, \xi_2, \ldots, \xi_{\sigma+1}$ be random variables such that $\xi_i$ represents the number of nodes in a multiset-trie on $i$-th level. For every node $j$ on $i$-th level we assign a random variable $\xi_{ij}$ to be the number of its children, such that $j \in [1, \xi_i]$. Then $\forall\ i \in [1, \sigma]$ the following holds:

$$\xi_{i+1} = \sum_{j=1}^{\xi_i} \xi_{ij}. \tag{1}$$

The random variable $\xi_{ij}$ is modeled by a zero-truncated binomial distribution on parameters $n$ and $p_{i+1}$, where $p_i = 1 - (1-p)^{n^{\sigma+1-i}}$. The distribution is zero-truncated, because any internal node has at least 1 descendant if $\mathcal{M}$ is non empty, and is binomial, since our model assumes a random input.

Applying the tools from Galton-Watson process [3] and using the fact that $\xi_1 = 1$, we calculate the expected number of nodes in a multiset-trie given parameters $n, \sigma$ and $p$

$$\mathbb{E}(|\mathcal{M}|) = \sum_{i=1}^{\sigma+1} n^{i-1} \frac{1-(1-p)^{n^{\sigma+1-i}}}{1-(1-p)^{n^{\sigma}}}. \tag{2}$$

With the expected number of nodes in a multiset-trie $\mathcal{M}$ obtained from (2), we can now generalize the result for a subtree in $\mathcal{M}$ parametrized by an input multiset $m$. The subtrees that we are interested in are the ones that contain all the submultisets or all the supermultisets of $m$. In order to calculate the expected cardinality of such subtrees we need the following definition.

Definition 1. *Let $m = \{1^{k_1}, 2^{k_2}, \ldots, \sigma^{k_\sigma}\}$, where $e^{k_e}$ is an element $e$ with multiplicity $k_e$. Let $M_1, M_2$ be the subsets of the set $M$, such that $M_1 = \{x \in M : x \subseteq m\}$ and $M_2 = \{x \in M : x \supseteq m\}$. Define $\alpha_i$ and $\beta_i$ as follows*

$$\alpha_i = \begin{cases} 1, & i = 0 \\ \prod_{j=1}^{i}(k_j + 1), & 1 \le i \le \sigma \end{cases}$$

*and*

$$\beta_i = \begin{cases} 1, & i = 0 \\ \prod_{j=1}^{i}(n - k_j - 1), & 1 \le i \le \sigma \end{cases}.$$

The expected cardinality of a multiset-trie subtree $\mathcal{M}_{M_1}$ that contains all the multisets from the set $M_1$ is equal to

$$\mathbb{E}(|\mathcal{M}_{M_1}|) = \sum_{i=1}^{\sigma+1} \alpha_{i-1} \frac{1-(1-p)^{\alpha_{i-1}}}{1-(1-p)^{\alpha_\sigma}}. \tag{3}$$

The expected cardinality of a multiset-trie subtree $\mathcal{M}_{M_2}$ that contains all the multisets from the set $M_2$ is equal to

$$\mathbb{E}(|\mathcal{M}_{M_2}|) = \sum_{i=1}^{\sigma+1} \beta_{i-1} \frac{1-(1-p)^{\beta_{i-1}}}{1-(1-p)^{\beta_\sigma}}. \tag{4}$$

The worst case time complexity of the multiset-trie operations is presented in the table below.

| **Function** | **Worst case complexity** |
|---|---|
| INSERT, SEARCH, DELETE | $O(\sigma)$ |
| GETALLSUBMSETS | $O(\lvert\mathcal{M}_{M_1}\rvert)$ |
| GETALLSUPERMSETS | $O(\lvert\mathcal{M}_{M_2}\rvert)$ |
| SUBMSETEXISTENCE | $O(\lvert\mathcal{M}_{M_1}\rvert - \lvert M_1\rvert)$ |
| SUPERMSETEXISTENCE | $O(\lvert\mathcal{M}_{M_2}\rvert - \lvert M_2\rvert)$ |

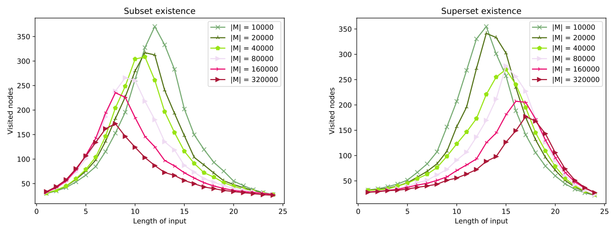


**Figure 2: Experiment 1, submsetExistence and supermsetExistence functions.**

## 2.4 Experiments

The implementation of multiset-trie is done in the C++ programming language and uses only the standard library of C++14 version of the standard.

In our experiments we will test the following functions: SUBMSETEXISTENCE, SUPERMSETEXISTENCE, GETALLSUBMSETS and GETALLSUPERMSETS. Performance of the functions will be measured by the number of visited nodes in multiset-trie by the particular function. In particular the performance is inversely proportional to the number of visited nodes. The selected parameters of the data structure that will be varied in experiments are as follows:

| | | |
|---|---|---|
| $\sigma$ | – | the cardinality of the alphabet $\Sigma$; |
| $n$ | – | the maximal degree of a node; |
| $\phi$ | – | mapping $\phi : \Sigma \to \{1, 2, \ldots, \sigma\}$; |
| $d$ | – | density of a multiset-trie, where $d$ is the function $d(\lvert M\rvert) = \frac{\lvert M\rvert}{\lvert N\rvert}$. |

The artificially generated multisets are constructed with respect to parameters $\Sigma, \sigma$ and $n$. The collection $M$ is sampled from $N$ with uniform distribution which simulates a random user input.

*Experiments 1 and 2.* The Experiments 1 and 2 demonstrate the performance of multiset-trie depending on the density. In both experiments the data is artificially generated at random, meaning that $\phi$ has no influence on results, and therefore is set to an identity mapping.

In the first experiment multiset-trie is used for storing and retrieving sets and the parameters $n$ and $\sigma$ are set to 2 and 25 respectively. The parameter $|M|$ varies from 10000 sets up to 320000 sets.

In the second experiment the multisets are used and the parameters $n$ and $\sigma$ are set to 6 and 25 respectively. The parameter $|M|$ varies from 40000 multisets up to 640000 sets.

The performance of the "existence" functions is different in Experiments 1 and 2. With increase of density it increases in the first experiment (see figure 2) and decreases in the second (see figure 3). It happens, because in the first case the multiset-trie is dense enough, so the increase of the density increases the probability of finding submultiset or supermultiset in multiset-trie, which lowers the number of visited nodes. However, in the second case the multiset-trie is

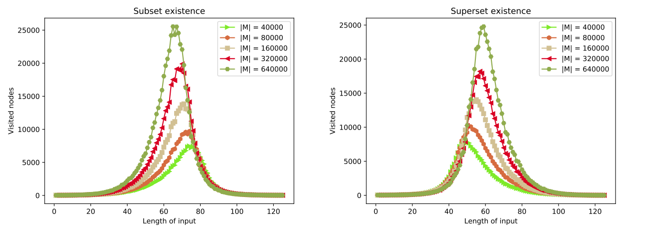


**Figure 3: Experiment 2, submsetExistence and supermsetExistence functions.**

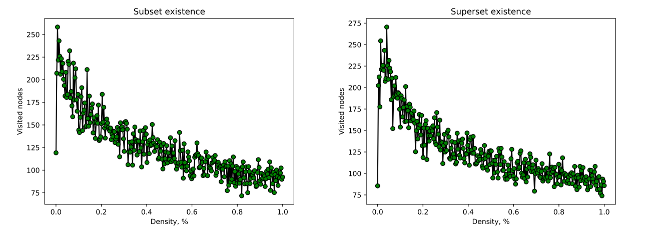


**Figure 4: Experiment 3, submsetExistence and supermsetExistence functions.**

very sparse. Multisets in a sparse multiset-trie differ more, which increases the number of visited nodes during search. We explain the reason of such a contradiction in the next Experiment 3.

The performance of the "get all" functions decreases as the density increases and is exponential in both experiments. The behavior is as expected, because the number of multisets in multiset-trie increases with increase of density, which means that any multiset in the data structure will have more sub- and supermultisets.

*Experiment 3.* This experiment, we empirically find the extremum of density for functions SUBMSETEXISTENCE and SUPERMSETEXISTENCE. The parameters $\sigma$ and $n$ are set to 12 and 5 respectively. The density varies from $1.0 \times 10^{-4}\%$ to 1.0%. The number of visited nodes was chosen to be maximal for each value of particular density.

The dependence of the number of visited nodes on density appears to be a non linear function. Yet the function is continuous on the interval $[0, 1]$, and hence, there exists a global maximum. As we see on figure 4 both functions SUBMSETEXISTENCE and SUPERMSETEXISTENCE indeed have the maximum around $d \approx 7.0 \times 10^{-3}\%$.

*Experiment 4.* This experiment shows the influence of the mapping $\phi$ from alphabet $\Sigma$ to a set of consecutive integers.

The test data is taken from English dictionary which contains 235883 different words. Those words are mapped to multisets of integers according to the $\phi$. In particular, we are interested in cases when $\phi(\Sigma)$ enumerates letters by their relative frequency in English language. We say that $\phi(\Sigma)$

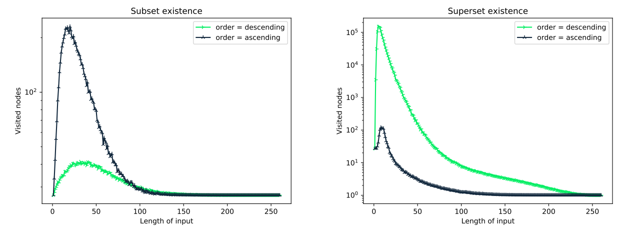


**Figure 5: Experiment 4, submsetExistence and supermsetExistence functions.**

maps letters in *ascending order* if the most frequent letter is mapped to number $\sigma$. Conversely, in *descending order* this letter is mapped to number 1. The parameters $\sigma$ is set to 26 and $n$ is set to 10. The results on figure 5 are more balanced when letters are ordered by frequency in ascending order. Letters that have the least frequencies are now located at the top of multiset-trie according to ascending order of letters by frequency. This means that the search becomes narrower, because a lot of invalid paths will be discarded on top most levels. Thus, multiset-trie can be traversed faster.

## 3. CONCLUSIONS AND FUTURE WORK

Our studies show that multiset-trie is an input sensitive data structure. However, pre-processing of data, letter frequency analysis in particular, determines the best encoding of input data and ensures the best performance of the multiset-trie.

The performance of multiset-trie observed in our research opens even more interesting questions for the future investigation. Further steps in our research will be to extend the functionality of the multiset-trie. We are interested in more flexible multiset containment queries, where the types of sub and supermultisets can be specified.

## 4. REFERENCES

[1] R. Baeza-Yates, B. Ribeiro-Neto, et al. *Modern information retrieval*, volume 463. 1999.

[2] A. Z. Broder, N. Eiron, M. Fontoura, M. Herscovici, R. Lempel, J. McPherson, R. Qi, and E. Shekita. Indexing shared content in information retrieval systems. In *International Conference on Extending Database Technology*, pages 313–330. Springer, 2006.

[3] C. W. Gardiner. *Stochastic methods*. Springer-Verlag, Berlin–Heidelberg–New York–Tokyo, 1985.

[4] J. M. Hellerstein, J. F. Naughton, and A. Pfeffer. *Generalized search trees for database systems*. September, 1995.

[5] C. D. Manning, P. Raghavan, H. Schütze, et al. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge, 2008.

[6] I. Savnik. Index data structure for fast subset and superset queries. In *International Conference on Availability, Reliability, and Security*, pages 134–148. Springer, 2013.

[7] J. Zobel and A. Moffat. Inverted files for text search engines. *ACM computing surveys (CSUR)*, 38(2):6, 2006.

# A development environment for medical image registration procedures

Matjaž Šuber
University of Primorska
Faculty of Mathematics, Natural Sciences
and Information Technologies
Glagoljaska 8, 6000 Koper
Koper, Slovenia
matjaz.suber@gmail.com

## ABSTRACT

Development of novel image registration methods and procedures is a tedious task because of the need of large number of specialized lower level functions and an environment with the abbility to observe results of each step of the processing. Although several software libraries and environments are available to simplify the development process, they require exhaustive learning and adaptation. To overcome this problem we aimed to implement a development environment that offers all this with only minimal influence on the user development strategies. We have decided to built it on top of Matlab environment that is already well established in this research field. The image registration environment includes a graphical user interface that simplifies testing and application of image registration procedures as well as their development by incorporating a large set of lower level processing functions. The environment was tested by using it for the development of the rigid registration procedure. The results clearly showed that we obtained a reliable environment, which has simplified and improved the implementation.

## Keywords

Medical image registration, Development environment, Graphical user interface, Image registration toolbox

## 1. INTRODUCTION

Medical image registration is a very important component of many clinical applications. It can be used for the detection and diagnosis of diseases, for planning the therapy, for guidance of interventions and for the followup and monitoring of patients.

The primary goal of image registration is to find the corresponding anatomical or functional locations in two or more images. Image registration can be applied to images from the same subject, acquired by the same (mono-modal image registration) or by different (multi-modal image registration) imaging modalities or at different time points (serial image registration).

In general, the process of image registration involves finding the optimal geometric transformation which maximizes the correspondences across images. Depending on the problem, geometric transformation can model only rotations, translations and scaling of images (rigid transformation) or can include deformations (nonrigid transformation).

In order to provide high quality image registrations, researchers need development tools, which simplify development and testing of methods by offering easy accesible supporting functions. Such an environment needs to work with 3D images, to provide a slice by slice visualization, to compare the geometric relations between images, to obtain image statistics like image histograms, to apply and display registration results and to provide tools that helps implementing new registration procedures.

Development of image registration methods can aid from different types of supporting software including programs like 3DSlicer [4] or ImageJ [1], programer libraries like ITK [2] and VTK [10] or toolboxes for common development environments like Matlab [11] or Python [9]. The problem of most of these tools is that they require user adaptation and learning.

In this paper we describe a new image registration environment for Matlab, that improves the implementation of image registration procedures. This environment includes a graphical user interface that simplifies testing and application of image registration procedures as well as their development by incorporating a large set of lower level processing functions. Its advantage is that it is straightforward and does not need extensive learning. In the next section we will firstly describe our environment, in section 3 we show how it can support the development and testing of registration procedures on an example of developing a rigid registration procedure. In section 4 we conclude the paper with a discussion.

## 2. SPECIFICATIONS

The image registration environment is meant to improve and speed-up the process of implementing new image registration procedures. In order to improve and speed-up the implementation phase this environment needs to satisfy the following requirements.

The first requirement is a data structure that will store images and related registration properties, such as image origin, global transformation, region of interest and much more. This data structure is meant to help implementing different image registration procedures in a common way. The second requirement is a graphical user interface that simplifies work with medical images and registration techniques. It needs to work with multiple 3D medical images which are saved in different file formats. The third requirement is the possibility to select different image registration procedures and to display their results. The fourth requirement are various tools that can be easily used to implement image registration components, such as measuring similarity using different similarity measures, resampling images and much more.

### 2.1 Data structure

The first requirement of the registration environment is a well defined data structure that stores images, their properties, registration results and other registration parameters. The main reason of such data structure is to provide a common way to manipulate images in different implementations of registration procedures and to display those images and registration results in the graphical user interface. The described data structure unifies data requirements of previous image registration implementations and is organized as an array of images. It includes a predefined set of properties which can be easily extended to suffice the needs of image registration procedures.

*Predefined properties of the data structure:*

**RefIdx:** Index of the image in the Img array to be used as a reference image.

**MovIdx:** Index of the image in the Img array to be used as a moving image.

**Img:** Array which stores all medical images and related registration properties.

**Img[i]:** Structure defining a single medical image and related registration properties.

**Img[i].name:** Image name.

**Img[i].path:** Absolute path to the image data.

**Img[i].voxelSize:** Image voxel size in millimeters.

**Img[i].data_orig:** Original image data in case of losing image information in the process of conversion to uint8 format.

**Img[i].data:** Image data used in registration procedures which is converted to uint8 format. The conversion is an established compromise between the necessary image brightness and the statistical significance of the registration procedure.

**Img[i].roi:** Image coordinates of region of interest.

**Img[i].O:** Image origin in millimeters. It is the point around which the global image registration transformations are defined.

**Img[i].T:** 3D global geometric transformation defined as 4x4 matrix.

**Img[i].D:** Deformation field that enables describing complex, including nonrigid geometric transformations, defined as a 3D displacements of individual image voxels.

### 2.2 Graphical user interface

The second requirement of the image registration environment is a graphical user interface that simplifies work with medical images and registration techniques. It allows loading 3D images in different file formats like DICOM [7], BrainWeb [3] and RIRE (Retrospective Image Registration Evaluation project) [12]. It provides a slice by slice visualization, image statistics like histograms and comparison of geometric relations between images. The graphical user interface also allows exporting and importing the described data structure in order to provide the capability to save temporary results that can be reused in future. One of the main features of the graphical user interface is to load and execute different image registration procedures. This feature simplifies and improves the testing phase of the development process.

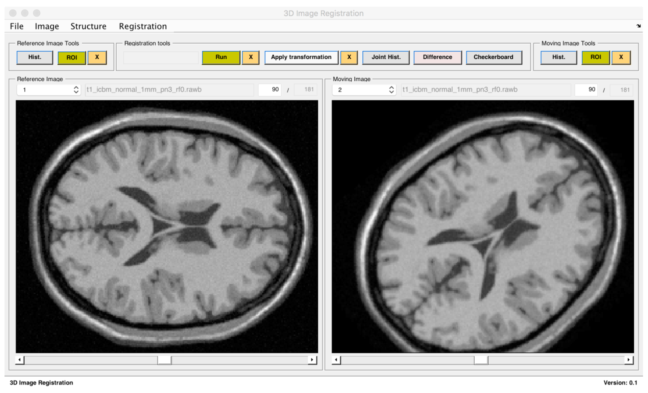


**Figure 1: Graphical user interface.**

The graphical user interface provides a set of tools that helps researchers to analyze image relations and registration results. This tools are divided into two groups. The first group consists of tools that can be executed over a single image like plotting a histogram as shown in figure 2 or setting a region of interest.

The second group consists of tools that helps analyzing geometric relations between two images. In this scope the graphical user interface includes displaying the joint intensity distribution histogram [8], absolute difference and checkerboard image.

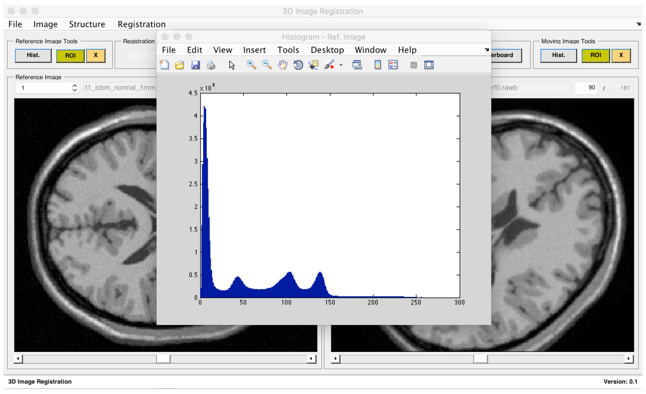


**Figure 2: Image histogram as an example of the first set of tools.**

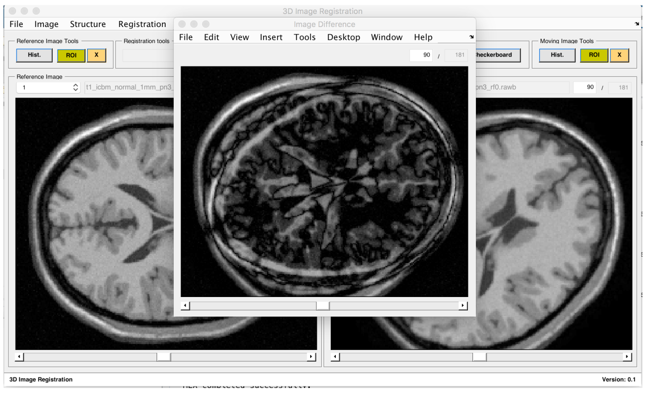


**Figure 3: Absolute difference as an example of the second set of tools.**

The graphical user interface is build with Matlab GUI [6] toolbox and can be easily initialized within Matlab workspace.

### 2.3 Registration Toolbox

Image registration procedures typically consists of three components. The first component is the geometric transformation which describes the movement, rotation, translation and/or deformation of objects in the image. The second component is the similarity metrix, which is used to estimate degree of image alignement. The third component is the optimization process. The goal of it, is to find the optimal geometric transformation which maximizes or minimizes the similarity of two images.

Implementation of these components may require a lot of effort if not supported by lower level processing functions. We have collected a large set of such lower level processing functions that were implemented in past research activities. This functions are grouped into a toolbox that can be used through the graphical user interface or directly in the Matlab workspace. This toolbox will be constantly updated according to the needs of future image registration procedures.

## 3. RESULTS

We have tested the image registration environment by using it for the development of a rigid registration procedure and demonstrate it on BrainWeb images.

We started by creating a new project on the top of the registration environment. Without any other interference we initialized the graphical user interface wich is shown in figure 4.

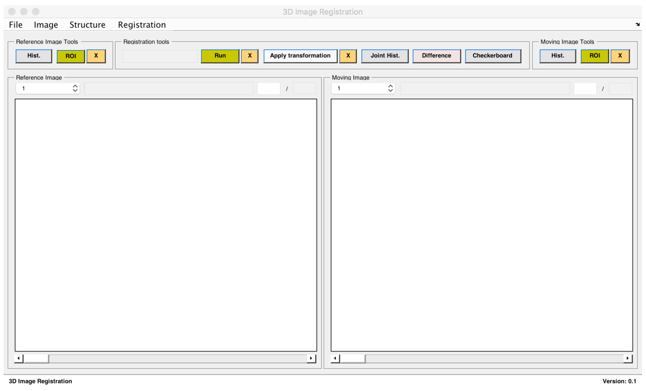


**Figure 4: Graphical user interface.**

After that we loaded the reference and moving image into the registration data structure through the user interface. For the reference image we used a MRI image of brain with $181 \times 217 \times 181$ voxels, 1mm slice thickness and with 3% of noise from the BrainWeb database. To create the moving image we applied a rigid rotation of 10 degrees over the reference image. The user interface displayed images as show in figure 1. Without any additional coding we could slide through all slices of the reference and moving image. In addition to that we analyzed the relation between the two images by displaying the joint intensity distribution histogram or the absolute difference as shown in figure 3.

At this point we only needed to focus on the core implementation of the rigid registration procedure. The step of creating the necessary environment was done in a very short time. To implement the rigid registration procedure we created three components. The first component was designed to controll the overall registration workflow. The second component was designed to compute the similarity between the reference and the moving image. This component was implemented by using low level processing functions from the integrated toolbox. The third component was the optimization process. It was implemented with the free/open-source NLopt [5] library which provides implementation of several optimization methods.

After we implemented the rigid registration procedure on the top of the registration data structure we started testing the implementation. In the graphical user interface we selected the implemented rigid registration method as shown in figure 5. We executed the procedure many times until we solved all issues and got the preferred results as shown in figure 6.

## 4. DISCUSSION AND CONCLUSION

From the described results we can conclude that the registration library has satisfied all the initial requirements. The predefined data structure helped to implement more structured registration methods and to easily display regis-

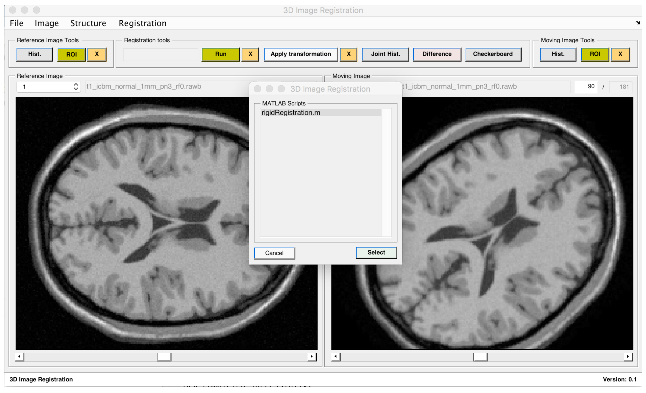


**Figure 5: File selector for selecting image registration procedures.**

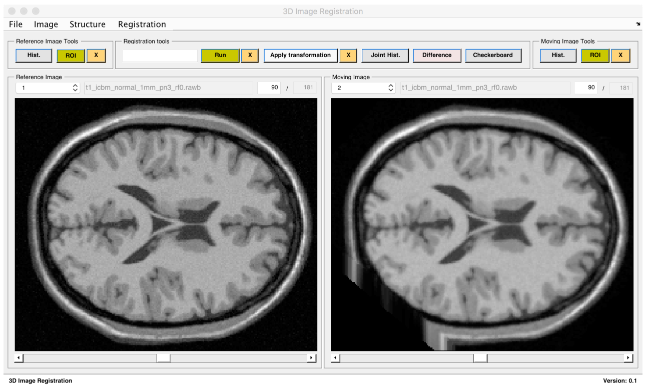


**Figure 6: Graphical user interface after image registration.**

tration results in the graphical user interface. Without any additional coding the graphical user interface displayed the imported medical images and provided all necessary tools for analyzing them. The testing phase was reasonably easier because of the possibility to apply the registration results directly on the loaded moving image inside the user interface. The integrated toolbox helped to speed up the implementation phase of the registration procedure. In the end we only needed to focus on the implementation of three small components. The first one was to controll the overall registration workflow, the second was to compute the similarity between images and the last component was the optimization process. To conclude, we shall sumarize our contribution presented in the paper. We have implemented a new image registration environment for Matlab. It offers support for development of image registration procedures. This environment includes a graphical user interface that simplifies testing and application of image registration procedures as well as their development by incorporating a large set of lower level processing functions. To provide a common way for manipulating images and registration results it includes a well defined data structure. The main goal of this environment is to provide a comprehensive solution to simplify and to speed-up the development of image registration procedures. We have used this library in a new project in which we implemented a rigid registration procedure for medical images. From the described results we can conclude that the development of such procedure within the new image registration environment was much faster and reliable.

## 5. ACKNOWLEDGMENT

We thank Peter Rogelj, professor at University of Primorska, for assistance in the implementation of image registration environment and for the provided set of low level processing functions that were implemented in past research activities.

## 6. REFERENCES

[1] M. D. Abràmoff, P. J. Magalhães, and S. J. Ram. Image processing with imageJ, 2004.

[2] B. B. Avants, N. J. Tustison, M. Stauffer, G. Song, B. Wu, and J. C. Gee. The Insight ToolKit image registration framework. *Frontiers in Neuroinformatics*, 8, 2014.

[3] C. Cocosco, V. Kollokian, R. K. Kwan, G. B. Pike, and A. C. Evans. BrainWeb : Online Interface to a 3D MRI Simulated Brain Database. *3-rd International Conference on Functional Mapping of the Human Brain*, 5(4):S425, 1997.

[4] A. Fedorov, R. Beichel, J. Kalpathy-Cramer, J. Finet, J. C. Fillion-Robin, S. Pujol, C. Bauer, D. Jennings, F. Fennessy, M. Sonka, J. Buatti, S. Aylward, J. V. Miller, S. Pieper, and R. Kikinis. 3D Slicer as an image computing platform for the Quantitative Imaging Network. *Magnetic Resonance Imaging*, 30(9):1323–1341, 2012.

[5] U. Kumar, S. Soman, and Jayadeva. Benchmarking NLopt and state-of-the-art algorithms for continuous global optimization via IACOR. *Swarm and Evolutionary Computation*, 27:116–131, 2016.

[6] C. S. Lent. Learning to Program with MATLAB Building GUI Tools. *John Wiley and Sons Inc.*, page 310, 2013.

[7] P. Mildenberger, M. Eichelberg, and E. Martin. Introduction to the DICOM standard, 2002.

[8] D. Mistry, A. Banerjee, and A. Tatu. Image Similarity based on Joint Entropy (Joint Histogram). *International Conference on Advances in Engineering and Technology*, (1):5, 2013.

[9] G. V. Rossum, P. S. Foundation, U. Swallow, S. Python, P. Software, and F. License. Python ( programming language ). *Flying*, pages 1–14, 2011.

[10] W. J. Schroeder and K. M. Martin. The visualization toolkit. In *Visualization Handbook*, pages 593–614. 2005.

[11] The Mathworks Inc. MATLAB - MathWorks, 2016.

[12] J. West, J. M. Fitzpatrick, M. Y. Wang, B. M. Dawant, C. R. Maurer, R. M. Kessler, R. J. Maciunas, C. Barillot, D. Lemoine, A. Collignon, F. Maes, P. Suetens, D. Vandermeulen, P. A. van den Elsen, S. Napel, T. S. Sumanaweera, B. Harkness, P. F. Hemler, D. L. Hill, D. J. Hawkes, C. Studholme, J. B. Maintz, M. A. Viergever, G. Malandain, and R. P. Woods. Comparison and evaluation of retrospective intermodality image registration techniques. *International Society for Optics and Photonics*, pages 332–347, 1996.

StuCoSReC

University of Primorska Press
www.hippocampus.si
ISBN *978-961-7023-40-4*