Computing and Informatics, Vol. 33, 2014, 410–430

A HYBRID EVOLUTIONARY ALGORITHM FOR EFFICIENT EXPLORATION OF ONLINE SOCIAL NETWORKS

Zorica Stanimirović, Stefan Mišković

Faculty of Mathematics, University of Belgrade
Studentski trg 16/IV, 11 000 Belgrade, Serbia
e-mail: {zoricast, stefan}@matf.bg.ac.rs

Abstract. Online social networks provide large amount of valuable data and may serve as research platforms for various social network analysis tools. In this study, we propose a mathematical model for efficient exploration of an online social network. The goal is to spend minimal amount of time searching for characteristics which define a sub-network of users sharing the same interest or having certain common property. We further develop an efficient hybrid method (HEA), based on the combination of an Evolutionary Algorithm (EA) with Local Search procedure (LS). The proposed mathematical model and hybrid method are benchmarked on real-size data set with up to 10 000 users in a considered social network. We provide optimal solutions obtained by CPLEX solver on problem instances with up to 100 users, while larger instances that were out of reach of the CPLEX were efficiently solved by the proposed hybrid method. Presented computational results show that the HEA approach quickly reaches all optimal solutions obtained by CPLEX solver and gives solutions for the largest considered instance in very short CPU time.

Keywords: Hybrid optimization method, evolutionary algorithm, local search, social network, data flow

Mathematics Subject Classification 2010: 68T20, 90B10, 90B80, 68M10, 90C11

1 INTRODUCTION

The concept of network has been widely used in the literature for describing online friendships and connections between users of online social sites. The studies on Twitter [18], blog sites [17, 25], Facebook and LinkedIn [8, 14, 20] use the model of network to represent the users of a social site, connections and interactions between them. Online social networks offer the opportunity to a user to create its own profile and connect to other users' profiles on different ways: via declaration, acceptance, friendship, appreciation ("like"), depreciation ("dislike"), etc. These networks may be open to everybody, or built around a particular group of people which share the same interest or certain common characteristic [6].

Enormous global popularity of online social network sites has initiated numerous studies and methods investigating different aspects of their use. The development of automated data collection and visualization processes have also contributed to the increased interest in research concerning online social networks. For example, the study by Lewis et al. [20] concerns interests and connections between university students on Facebook. Lerman and Ghosh in [19] investigate how the news spreads along networks of Twitter and Digg users, while Bruns et al. [7] deal with tracking topical discussions on blogs. Several hyperlink analysis methods have been developed, representing a part of webometrics, which measures and analyzes activity in cyberspace [30]. Several softwares have been introduced for exploring online social networks: Gephi – an open source software for exploring and manipulating networks [4], SocSciBot – a software designed to collect data on organizational hyperlink networks on the web [33], IssueCrawler – another software that enables the collection and analysis of hyperlink data, and is popular in the humanities and social sciences [34], etc.

As different events and themes provoke varying interactions and conversations, it is proposed that an analysis would help studies of online social networks by further examining the dynamics of links and information flow. Communications and interactions between users (represented by nodes) can be used to identify a subnetwork of the initial online social network, i.e. the group of users having certain common interest. Similarly, connections among different online social networks sites can be found through hyperlinks. Concepts and models from network-based studies in optimization theory and applications may be adapted for research into online networks, such as facility location models [11, 28], hub networks [3, 9], network flow models [2, 5] and covering models [16, 29].

In this paper, we present a mathematical model for efficient analysis of an online social network, providing new information on linking behaviors and information flow within this network. We consider an online social network with large number of users exchanging some information. The goal is to design an efficient search strategy, which will help to identify nodes that exchange information containing certain keywords, which further may indicate that identified nodes belong to the same interest group. The proposed model may be used in various social behavior studies, market research, political marketing and also in security purposes, such as discov-

ering sexual harassment, child pornography, mobbing, bullying in the cyberspace etc.

Our idea is to focus attention on the nodes in the network which exchange the largest amounts of information flow. Among them, we identify certain number of control points and consider the flow which goes from/to these nodes. The goal is to decide which nodes will be chosen as control ones, such that maximal time needed to search the information flow from/to a chosen control node is minimized. Once we identify the sub-network of control nodes, we consider the flow that departs or is destined to the chosen control nodes and search it by using certain keywords. The choice of a keyword (one or more) depends on the nature and scope of the research on the considered online social network.

The proposed model is benchmarked on three generated data sets containing instances with different number of nodes. The CPLEX 12.1 solver is used to solve instances to optimality, but it only provided optimal solutions for instances of smaller size. In order to solve instances of real-life dimensions, we designed a hybrid evolutionary based method - HEA. The proposed HEA successfully combines an efficient local search heuristic and an evolutionary algorithm approach. The conducted computational experiments show the robustness and efficiency of the HEA approach when solving the instances from all three considered benchmark sets, especially the ones of real-life dimensions.

The remainder of the paper is organized as follows. In Section 2, we formulate the problem and present its mathematical formulation. Section 3 explains in detail all important aspects of the proposed hybrid algorithm HEA. In Section 4, we present and discus computational results on the generated small, medium and large-scale benchmark sets. In Section 4, we present optimal solutions obtained by CPLEX 12.1 solver on the small-size problem instances with up to 100 users. We further provide results of the proposed HEA on all three generated benchmark sets, in which the largest considered instances involve up to 10 000 user nodes. Finally, in Section 5, we draw out some conclusions and propose ideas for the future work.

2 PROBLEM DESCRIPTION AND MATHEMATICAL FORMULATION

We consider the set $I = \{1, \ldots, n\}$ of n users in a network. By $W = [w_{ij}]$ we denote the data flow matrix $n \times n$, where $w_{ij} \geq 0$ is the number of data flow units that originate from user i and are directed to user j. For each $i \in I$, we calculate $G_i = \sum_{j \in I} w_{ij} + \sum_{j \in I, i \neq j} w_{ji}$, representing the the sum of data flow units that are received or sent by a node i. Note that in general case, $w_{ii} \neq 0$ and flow preservation equality $\sum_{j \in I} w_{ij} = \sum_{j \in I} w_{ji}$ does not hold necessarily for each $i \in I$.

User nodes $i \in I$ are sorted in descending order according to the assigned flows G_i. In this way, we obtain the array i_k, $k = 1, \ldots, n$, where $G_{i_1} \geq G_{i_2} \geq \ldots \geq G_{i_n}$ holds. We consider only the first $m \leq n$ members of the sorted array

and obtain the subset $J = \{i_1, \ldots, i_m\}$, $J \subseteq I$. Parameter m depends on n in a pre-determined way ($m = n/2$, $m = n/3$, $m = n/4$, ...) and it may be varied.

In this study, we have chosen $m \leq n$ nodes from I that exchange the largest amount of flow to obtain $J \subseteq I$. However, the subset J may be derived from I regarding some other criterion, such as: incoming or outcoming data flow, geographical position of user nodes, the frequency of exchanging information, etc. A combination of two or more criteria may also be considered, depending on the situation in practice.

We further introduce the following notation:

- p is the number of control nodes to be chosen among the nodes from J;
- $T = [t_{ij}]$ is matrix $n \times n$, where $t_{ij} > 0$ is the time needed to search through one unit of data flow which originates from user i and is directed to user j;
- $\alpha \in (0, 1)$ is the parameter which reflects a faster search through the data flow between chosen control nodes from J;
- b_k, $k \in I$ is the capacity of a node k, i.e. the maximal time that can be spent in searching through the incoming data to a node k.

The goal of the problem is to determine locations for exactly p control nodes, such that the maximal time needed to search the flow through the chosen control nodes is minimized.

The formulation uses binary decision variables:

$$y_j = \begin{cases} 1 & \text{if } j \text{ is chosen as control node,} \\ 0 & \text{otherwise,} \end{cases}$$

for each $j \in J$ and a non-negative decision variable z, representing the upper limit on the time needed to search the flow in the sub-network of control nodes.

Using the notation above, the mathematical formulation of the can be written as:

$$\min z \tag{1}$$

subject to:

$$\sum_{j \in J} y_j = p \tag{2}$$

$$\sum_{i \in I} w_{ik} t_{ik} y_k + (\alpha - 1) \sum_{i \in J} w_{ik} t_{ik} y_k y_i \leq b_k, \quad \forall k \in J \tag{3}$$

$$\sum_{i \in I} (w_{ik} t_{ik} + w_{ki} t_{ki}) y_k + (\alpha - 1) \sum_{l \in J} (w_{kl} t_{kl} + w_{lk} t_{lk}) y_k y_l - \alpha w_{kk} t_{kk} y_k \leq z, \quad \forall k \in J \tag{4}$$

$$y_j \in \{0, 1\}, \quad \forall j \in J \tag{5}$$

$$z \geq 0. \tag{6}$$

By objective function (Equation (1)) we minimize the maximal time needed to search the flow through an established control node. Constraint (Equation (2)) indicates that exactly p control nodes are established. Constraints (Equation (3)) ensure that the time spent in searching through the incoming data to each control node is limited. By constraints (Equation (4)), we impose the lower bounds on the value of objective variable z. Constraints (Equation (5)) reflect the binary nature of decision variables y_j, while (Equation (6)) denotes that continuous variable z takes a non-negative value.

Note that capacity constraint (Equation (3)) will be satisfied if $y_k = 0$, i.e. node $k \in J$ is not chosen as control one. If $y_k = 1$, it means that $k \in J$ is control node and the left side L of constraint (Equation (3)) can be written as:

$$\begin{aligned} L &= \sum_{i\in I\setminus J} w_{ik}t_{ik} + \sum_{i\in J} w_{ik}t_{ik} + (\alpha-1)\sum_{i\in J} w_{ik}t_{ik}y_i \\ &= \sum_{i\in I\setminus J} w_{ik}t_{ik} + \sum_{i\in J,y_i=0} w_{ik}t_{ik} + \sum_{i\in J,y_i=1} w_{ik}t_{ik} + (\alpha-1)\sum_{i\in J,y_i=1} w_{ik}t_{ik} \\ &= \sum_{i\in I\setminus J} w_{ik}t_{ik} + \sum_{i\in J,y_i=0} w_{ik}t_{ik} + \alpha\sum_{i\in J,y_i=1} w_{ik}t_{ik} \end{aligned}$$

Therefore, if $y_k = 1$, the constraint (Equation (3)) reduces to

$$\sum_{i\in I\setminus J} w_{ik}t_{ik} + \sum_{i\in J,y_i=0} w_{ik}t_{ik} + \alpha\sum_{i\in J,y_i=1} w_{ik}t_{ik} \le b_k \tag{7}$$

Example 1. Consider a network I with $n = 10$ user nodes. Flow matrix W and cost matrix T are given by

$$W = \begin{bmatrix} 68 & 1 & 25 & 59 & 65 & 0 & 82 & 62 & 96 & 28 \\ 92 & 0 & 0 & 93 & 22 & 0 & 0 & 48 & 72 & 70 \\ 0 & 100 & 95 & 0 & 0 & 34 & 65 & 12 & 69 & 45 \\ 58 & 60 & 42 & 79 & 0 & 91 & 89 & 0 & 43 & 49 \\ 6 & 30 & 51 & 0 & 0 & 49 & 24 & 55 & 41 & 77 \\ 9 & 40 & 24 & 39 & 0 & 30 & 34 & 0 & 59 & 0 \\ 78 & 0 & 87 & 46 & 73 & 30 & 74 & 13 & 91 & 37 \\ 68 & 75 & 53 & 51 & 25 & 31 & 0 & 8 & 58 & 54 \\ 46 & 0 & 0 & 22 & 23 & 7 & 14 & 1 & 63 & 11 \\ 25 & 49 & 96 & 3 & 92 & 75 & 0 & 22 & 100 & 85 \end{bmatrix}$$

and

$$T = \begin{bmatrix} 0.1 & 0.4 & 0.3 & 0.9 & 0.8 & 0.8 & 0.1 & 0.8 & 0.7 & 0.7 \\ 0.8 & 0.3 & 0.8 & 0.3 & 0.7 & 0.1 & 0.1 & 0.7 & 0.3 & 0.4 \\ 0.9 & 0.6 & 0.5 & 0.1 & 0.1 & 0.9 & 0.9 & 0.6 & 0.8 & 0.3 \\ 0.4 & 0.8 & 0.4 & 0.9 & 0.9 & 0.2 & 0.5 & 0.5 & 0.3 & 0.3 \\ 0.3 & 0.7 & 0.4 & 0.3 & 0.8 & 0.1 & 0.8 & 0.8 & 0.1 & 0.6 \\ 0.8 & 0.1 & 0.9 & 0.8 & 0.9 & 0.7 & 0.2 & 0.2 & 0.8 & 0.2 \\ 0.8 & 0.9 & 0.1 & 0.7 & 0.8 & 0.1 & 0.5 & 0.8 & 0.6 & 0.1 \\ 0.2 & 0.4 & 0.2 & 0.5 & 0.8 & 0.6 & 0.2 & 0.6 & 0.5 & 0.3 \\ 0.9 & 0.2 & 0.4 & 0.6 & 0.1 & 0.8 & 0.2 & 0.1 & 0.1 & 0.9 \\ 0.7 & 0.6 & 0.2 & 0.9 & 0.5 & 0.2 & 0.1 & 0.1 & 0.3 & 0.9 \end{bmatrix}.$$

Exactly $m = 4$ nodes which exchange the largest amounts of flow are to be identified, representing potential locations for $p = 2$ control nodes. Parameter $\alpha = 0.5$ denotes reductions when exploring the data traffic between the chosen control nodes.

For each node $i \in I$ we calculate the amount of flow that goes from/to node i and obtain the array $G = (G_1, G_2, \ldots, G_n) = (936, 752, 893, 903, 633, 582, 911, 644, 879, 1\,003)$ (see Figure 1). The array of nodes sorted according to the assigned amount of flow (in decreasing order) is: 10, 1, 7, 4, 3, 9, 2, 8, 5, 6. Since $m = 4$, the nodes 10, 1, 7 and 4 will be chosen for the subset $J \subset I$.

①	②	③	④	⑤
$G_1 = 936$	$G_2 = 752$	$G_3 = 893$	$G_4 = 903$	$G_5 = 633$
⑥	⑦	⑧	⑨	⑩
$G_6 = 582$	$G_7 = 911$	$G_8 = 644$	$G_9 = 879$	$G_{10} = 1\,003$

Figure 1. The set of nodes I with corresponding amounts of data flow G_i

Capacity constants b_k have direct impact on optimal solution of the problem. Different values for capacities b_k may lead to different optimal solutions on the same problem instance. In some cases, the solution may not exist due to lower value of one or more b_k. Considering the capacity constraint Equation (3) and the inequality Equation (7), we conclude that b_k should be chosen such that $\sum_{i \in I \setminus J} w_{ik} t_{ik} + \alpha \sum_{i \in J} w_{ik} t_{ik} \leq b_k$ holds for every $k = 1, \ldots, n$, in order to provide feasibile solution of the problem, even in the worst case (when all $y_k = 1$, $k \in J$). If b_k, $k = 1, \ldots, n$ are larger than the values on the left side of this inequality, the problem reduces to uncapacitated case.

In order to demonstrate the impact of the capacity constants b_k on the solution of the problem, we have considered different values of b_k in our example and solved the obtained problems to optimality, if possible.

a) Let $b_k = 240$ for every $k = 1, \ldots, n$. In this case, the obtained optimal solution indicates that control nodes are located at 4 and 7 and the corresponding objective function value is 396.85.

b) Let $b_k = 200$ for every $k = 1, \ldots, n$. In optimal solution, nodes 7 and 10 are chosen as control ones, and the objective function value is 408.55.

c) Let $(b_1, \ldots, b_n) = (140, 210, 170, 230, 190, 200, 120, 140, 150, 190)$. In optimal solution for this case, control nodes are located at 10 and 4 and objective value is 426.5.

d) Let $b_k = 180$ for every $k = 1, \ldots, n$. In this case, the problem has no feasible solution.

Optimal solutions for the cases *a*), *b*) and *c*) are presented in Figure 2 (from left to right).

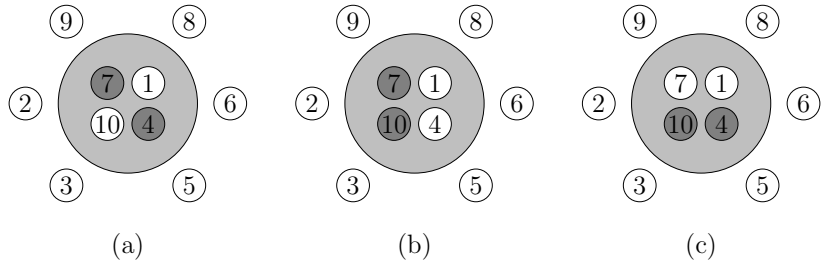

Figure 2. Optimal solution for cases *a*), *b*) and *c*)

3 PROPOSED HYBRID EVOLUTIONARY ALGORITHM

Evolutionary algorithm is an optimization technique based on the imitation of the natural process of evolution [15]. However, there are many situations in which a pure evolutionary algorithm does not perform particulary well. The main drawbacks of a pure EAs are the slow convergence and the possibility of finishing in a local optimum. One of the strategies for speeding up the convergence and avoiding a local optimum trap is to incorporate other exact or heuristic methods into the pure EA. Various hybridizations of the EA with other methods have been proposed in the literature: [1, 10, 12, 22, 26, 27, 32, 31], etc.

In this paper, we propose an efficient hybrid method (HEA) based on combining the evolutionary approach with a local search improvement procedure. The role of the evolutionary part in the proposed HEA is to direct the search to promising regions of a search space. Once promising regions with high quality solutions have been identified, a local search method is applied in order to determine the best solutions in these regions. Solutions generated by the means of evolutionary operators are subject to the local search procedure. Improved solutions will enter a new generation only if they satisfy a fitness quality criteria.

The basic scheme of the proposed HEA approach is presented in Algorithm 1.

Algorithm 1 The basic structure of the HEA

```
Read Input()
Generate initial population()
while not Termination criteria do
    Fitness function calculation()
    Selection()
    Recombination()
    Local search procedure()
end while
Write output()
```

After generating the initial population, a greedy procedure has been used in order to evaluate fitness function of individuals, representing potential solutions of the problem. In each generation of the HEA, the worst 1/3 of the population is replaced, while the remaining 2/3 of the population are directly passed to the next generation. The chromosome and the objective function value of the best individual are saved and updated when an improvement is obtained. A tournament selection operator with different number of tournament participants is used. As a recombination operator, we use a modified crossover operator that is adopted to the problem under consideration. Simple and efficient Local search procedure is applied in order to improve newly-generated individuals. Among the improved individuals we choose the best fitted ones to enter new generation. Several additional strategies have been applied in order to increase the efficiency of the proposed hybrid algorithm. The HEA uses a combination of two termination criteria: the algorithm stops if the maximal number of 50 000 generations is reached or no improvement of the best individual is achieved through 5 000 consecutive generations.

3.1 Evolutionary Part of the HEA

The proposed HEA algorithm uses the binary representation of solutions. Each solution is represented by a binary string (chromosome) of length $m = |J|$. Each bit in a chromosome corresponds to one potential control node from J. If the bit on the k^{th} position in the chromosome takes the value of 1, it means that a control node is located at the node on the k^{th} position in the sorted array of nodes from J regarding the amount of data flow, and 0 otherwise. The values of binary variables y_k, $k = 1, 2, \ldots, m$ are obtained from the chromosome. For example, chromosome chr $= (0, 0, 1, 1)$ corresponds to optimal solution of the problem presented in Example 1 with $n = 8$, $m = 4$ and $p = 2$. Control nodes are located at nodes at positions 3 and 4 in the sorted array of nodes in J, which gives us the values of variables y_k: $y_3 = y_4 = 1$ and $y_1 = y_2 = 0$. Since the sorted array of nodes in J is 10, 1, 7, 4, it means that the nodes with indices 4 and 7 from the initial set of user nodes I are chosen as control ones.

The initial indices of established control nodes are easily obtained by taking the first m members of the sorted vector $(i_1, i_2, \ldots, i_m, i_{m+1}, \ldots, i_n)$ of nodes in I and calculating the product of vector $(i_1, i_2, \ldots, i_m)$ and vector representing a genetic code $(y_1, \ldots, y_m)$, which is defined as $(i_1, i_2, \ldots, i_m) * (y_1, \ldots, y_m) = (i_1 \cdot y_1, i_2 \cdot y_2, \ldots, i_m \cdot y_m)$. In our example, $(10, 1, 7, 4) * (0, 0, 1, 1) = (0, 0, 7, 4)$, which gives us the indices of chosen control nodes.

The fitness function of an individual is equal to its objective function and it is calculated in the following way. From an individual's chromosome, we first obtain the locations of established facilities (and therefore, the values of variables y_k) and then check whether the chromosome is correct or not. A chromosome is labeled as "correct", if the following two conditions are satisfied:

1. there exist exactly p ones in the chromosome;
2. inequality $\sum_{i \in I} w_{ik} t_{ik} y_k + (\alpha - 1) \sum_{i \in J} w_{ik} t_{ik} y_i y_k \leq b_k$ is satisfied for each established control node $k \in J$.

The objective function of a "correct" chromosome chr is then calculated as follows:

$$\max_{k \in J} \sum_{i \in I} (w_{ik} t_{ik} + w_{ki} t_{ki}) \mathrm{chr}(k)$$

$$+ (\alpha - 1) \sum_{l \in J} (w_{kl} t_{kl} + w_{lk} t_{lk}) \mathrm{chr}(k) \mathrm{chr}(l) - \alpha w_{kk} t_{kk} \mathrm{chr}(k) \tag{8}$$

where $\mathrm{chr}(k)$, $\mathrm{chr}(l)$ represent the bit values in chromosome chr at positions k and l, respectively. Chromosomes that are not "correct" are excluded from the population by setting their fitness to ∞. In this way, selection operator will prevent them to enter the new generation.

The initial population, numbering 300 individuals, is generated randomly by uniform distribution, which ensures good diversity of the genetic material. In order to provide better quality of the genetic material in the initial population, individuals in the initial population are created with exactly p ones in the genetic code. For generating an individual's genetic code, we first choose a random value by uniform distribution $i_1 \in \{1, 2, \ldots, m\}$ and set the bit value at position i_1 to 1. In the same way, we randomly choose $i_2 \in \{1, 2, \ldots, m\} \setminus \{i_1\}$, $i_3 \in \{1, 2, \ldots, m\} \setminus \{i_1, i_2\}$, $\ldots$, $i_p \in \{1, 2, \ldots, m\} \setminus \{i_1, \ldots i_{p-1}\}$ by uniform distribution and set bit values at positions $i_2, \ldots, i_p$ to 1. Remaining $m - p$ bits in the individual's genetic code take the value of 0.

A steady-state generation replacement scheme with elitist strategy is used, which means that only 1/3 of individuals are replaced in every generation, while the best 2/3 of individuals are directly passing in the next generation. Elite individuals preserve highly fitted genes and do not need recalculation of fitness function, which provides additional time savings. In order to preserve the diversity of genetic material in the population, we discard duplicate individuals from the population and limit the appearance of individuals with the same fitness but different chromosomes to some constant [35].

As a selection method, we used the Fine Grained Tournament Selection, introduced in [13]. Instead of having an integer tournament size, as in the classic tournament selection, the Fine Grained Tournament Selection operator depends on real parameter F representing the "desired tournament size". In this HEA implementation, F takes the value of 5.4, while the size of each tournament to be performed is chosen from the set $\{\lfloor F \rfloor, \lceil F \rceil\}$.

Individuals that are tournament winners are further subjected to the modified crossover operator. We randomly choose the pairs of parent-chromosomes that will exchange their genetic material and produce two offspring-chromosomes.

The exchange of genetic material of parent-chromosomes chr1 and chr2 is performed by repeating steps 1–3 exactly $m/2$ times:

1. Randomly choose a bit position $i \in \{1, 2, \ldots, m\}$, such that $\text{chr1}(i) = 1$ and $\text{chr2}(i) = 0$;
2. Randomly choose a bit position $j \in \{1, 2, \ldots, m\}$, such that $\text{chr1}(j) = 0$ and $\text{chr2}(j) = 1$;
3. Parent-chromosomes chr1 and chr2 exchange bits on the chosen positions i and j.

The probability that a chosen pair of parent-chromosomes will exchange bits and create two offspring-chromosomes is set to $1/\sqrt[3]{k}$, where k is the current number of HEA iterations. If no bit exchange occurs, offspring-chromosomes remain identical to their parents. Note that the proposed crossover operator preserves exactly p ones in the offspring's genetic codes. Therefore, created offspring will always satisfy the first conditions to be labeled as "correct" individuals (see the definition above). The probability of crossover decreases as the MA progresses, since we want to increase the impact of local search procedure in later stages of the MA.

However, it may happen that one or both offsprings do not satisfy the second condition for "correct" individual, i.e. capacity condition. This directly depends on the chosen values of capacities b_k. If the values of b_k are "tight", the possibility to create infeasible offspring increases, while for "loose" values of b_k the offsprings are more likely to be "correct". Offsprings that are not correct are excluded from the population by setting their fitness to infinity in the fitness function calculation part.

3.2 Local Search Procedure

Evolutionary-based algorithms, as nature-inspired optimization methods, often make use of local search procedures for refining solutions that are generated during the evolutionary exploration of the search space. At initialization part, the EAs generally try to capture a global picture of the search space, and later, during the search process, they successively focus the search on more promising regions of the search space. However, the EAs are usually not so effective concerning the exploitation of the accumulated search experience, that is, finding the best solutions in these

high quality areas. On the other side, the strength of local search lies in the capability of quickly finding better solutions in the neighborhoods of given starting solutions. The success of a hybridization of EA and local search method for solving different combinatorial optimization problems is in the fact that EA is good in identifying promising areas of the search space in which local search methods can then quickly determine the best solutions [23, 21, 24], etc.

In this paper, we exploit this idea and after the evolutionary phase, we apply a local search method that is adopted to the problem under consideration. The local search procedure LS is applied on each individual in each HEA generation, providing additional improvements of solutions and preventing the EA to converge to a local optimum. In the proposed Local search procedure, we first randomly choose two bits on the positions r and s in an individual's chromosome chr, having different values. If, for example, $\mathrm{chr}(s) = 1$ and $\mathrm{chr}(r) = 0$, it means that the s^{th} node i_s in the sorted array of nodes from J is chosen as the control one, while the r^{th} node i_r is not. We try to exchange established control node $i_s \in J$ with non-established one $i_r \in J$, looking for an improvement of the fitness function. This step is repeated as long as we obtain an improvement of the individual's fitness value. Only in the case that the new fitness value is better than the previous one, we perform necessary changes in the individual's genetic code and obtain a new, improved individual. By exchanging two bits with different values in each iteration of the LS procedure, we ensure that the number of bits with the value of 1 remains the same. In this way, we preserve the feasibility of individuals, since the number of established control nodes is fixed to p. The basic scheme of the implemented LS procedure is represented by Algorithm 2.

Algorithm 2 Local search heuristic

```
1: for all chr ∈ population do
2:    while exists improvement for chr do
3:       Randomly choose bit position r in the chr, such that chr(r) = 0
4:       Randomly choose bit position s in the chr, such that chr(s) = 1
5:       Apply procedure Exchange(chr, r, s)
6:    end while
7: end for
```

Note that the fitness evaluations of newly created individuals within LS cycles is the most time-consuming part of the LS and it may significantly affect the total running time of the HEA. Therefore, we apply a strategy for decreasing the computational complexity of calculating fitness values of new individuals. Conducted computational experiments showed that this strategy provides significant running time reductions. The basic concept of the applied strategy is shown in Algorithm 3. The array $\{\sum_{i_k}\}$, $i_k \in J$ stores the values of the sum (6) for each $i_k \in J$, while the procedure *RestoreValues* restores previous values of the parameters in the case of unsuccessful exchange of bits.

Algorithm 3 Exchange bits on positions r and s in chromosome chr

Exchange(chr, r, s)
$\text{chr}(r) = 1$
$\text{chr}(s) = 0$
for all $i_k \in J$, $i_k \neq i_r$ **do**
  if $\text{chr}(k) = 1$ **then**
    $\sum_{i_k} = \sum_{i_k} + (\alpha - 1)(w_{i_r i_k} t_{i_r i_k} + w_{i_k i_r} t_{i_k i_r}) + (1 - \alpha)(w_{i_k i_s} t_{i_k i_s} + w_{i_s i_k} t_{i_s i_k})$
  end if
end for
$\sum_{i_r} = \sum_{k \in I} (w_{k i_r} t_{k i_r} + w_{i_r k} t_{i_r k}) + (\alpha - 1) \sum_{i_l \in J} (w_{i_r i_l} t_{i_r i_l} + w_{i_l i_r} t_{i_l i_r}) \text{chr}(l) - \alpha w_{i_r i_r} t_{i_r i_r}$
$\max = 0$
for all $i_k \in J$ **do**
  if $\sum_{i_k} > b_{i_k}$ **then**
    return
  end if
end for
for all $i_k \in J$ **do**
  if $\text{chr}(k) = 1$ **and** $\sum_{i_k} > \max$ **then**
    $\max = \sum_{i_k}$
  end if
end for
if $\max < \mathit{fitness}(\mathit{chr})$ **then**
  $\mathit{fitness}(\mathit{chr}) = \max$
else
  $\mathit{RestoreValues}()$
  return
end if

4 COMPUTATIONAL RESULTS

All computational experiments were carried out on an Intel Core i5-2430M on 2.4 GHz with 8 GB DDR3 RAM memory under Windows 7 operating system. The HEA implementation was coded in $C\sharp$ programming language. On each instance from the considered data set, the HEA was run 15 times with different random seeds.

In order to verify the quality of the HEA solutions, optimization package CPLEX, version 12.1, is used to solve considered instances to optimality, if possible. We have imposed the time limit of 2h on CPLEX 12.1 runs as an additional stopping criterion (if CPLEX 12.1 does not find optimal solution within this time, it will stop). The CPLEX 12.1 has been run in the same computational environment as the proposed HEA.

We have generated three sets of instances for our computational experiments: small-size S, medium-size M and large-scale L. The values of parameters n, m for each data set are presented in Table 1.

Data set	$n = \|I\|$	$m = \|J\|$
Small-size data set (S)	50, 100	10, 20, 25
Medium-size data set (M)	200, 500, 750	50, 100
Large-scale data set (L)	$1\,000 \leq n \leq 10\,000$	$100 \leq m \leq 1\,000$

Table 1. Data sets used in our computational experiments

Parameters p, α and input data w_{ij}, t_{ij} and b_j are chosen in the following way:

- $p \in [\lfloor m/4 \rfloor, \lceil 3m/4 \rceil]$ and $p \in N$;
- $\alpha \in (0, 1)$;
- $w_{ij} \in U[0, 1\,000]$;
- $t_{ij} \in U[0, 1]$;
- $b_j \in \left[0.95 \sum_{i \in J} w_{ij} t_{ij}, 1.05 \sum_{i \in J} w_{ij} t_{ij}\right]$.

The notation used for column headings in Tables 3–5, presented in the remaining part of this section, is given in Table 2.

Notation	Description
S, M, L	Instance type according to problem dimension (Small, Medium, Large);
n, m, p	Instance parameters $n = \|I\|$, $m = \|J\|$, p =number of control nodes to be located;
Opt.Sol.	Optimal solution obtained by CPLEX 12.1 solver;
t	Total CPLEX 12.1 running time (in seconds);
N	Number of nodes that CPLEX 12.1 used to obtain optimal solution;
Best.Sol.	The best value of the HEA method, with mark *Opt* in cases when the HEA reached optimal solution *Opt.Sol.*;
t_{HEA}	Average running time in which the HEA reaches the best/optimal solution (in seconds);
gen	Average number of the HEA generations;
agap	Average gap of the HEA's solution from the optimal/best one (in percents);
σ	Standard deviation of the HEA's solution from the optimal/best one (in percents).

Table 2. Notation used for presenting computational results

In Table 3, we present optimal solutions for 40 small-size tested instances, obtained by CPLEX 12.1 solver and the best results of the proposed HEA approach. On this small size data set, the average running time for the CPLEX 12.1 is 73.76 seconds, while the average number of nodes is around 7 325. The column *Best.Sol.* in Table 3 shows that the proposed HEA approach reaches all optimal solutions previously obtained by CPLEX 12.1 solver, while values in the next column $t(s)$

Instance				CPLEX			HGA				
S	n	m	p	*Opt.Sol.*	$t(s)$	*Nodes*	*Best.Sol.*	$t(s)$	*gen*	*agap*(%)	σ(%)
1	50	10	4	778.679	0.499	12	Opt	0.001	1.6	0.000	0.000
2	50	10	4	813.398	0.453	24	Opt	0.001	1.6	0.000	0.000
3	50	10	3	839.637	2.730	3	Opt	0.001	2.7	0.000	0.000
4	50	10	3	822.243	1.030	15	Opt	0.001	1.8	0.000	0.000
5	50	10	3	824.441	0.515	22	Opt	0.001	1.5	0.000	0.000
6	50	10	4	787.199	0.374	5	Opt	0.001	1.6	0.000	0.000
7	50	20	5	815.970	2.871	65	Opt	0.001	1.8	0.000	0.000
8	50	20	8	826.540	0.530	0	Opt	0.001	9.2	0.000	0.000
9	50	20	6	807.579	2.657	684	Opt	0.031	29.7	0.000	0.000
10	50	20	5	755.736	1.881	117	Opt	0.001	10.0	0.000	0.000
11	50	20	9	783.302	1.896	217	Opt	0.032	27.2	0.000	0.000
12	50	20	9	698.674	92.124	22 930	Opt	0.237	291.6	0.000	0.000
13	50	20	9	693.408	82.102	21 011	Opt	0.050	66.7	0.000	0.000
14	50	25	7	739.710	5.284	2 371	Opt	0.125	165.3	0.000	0.000
15	50	25	7	770.558	8.772	6 319	Opt	0.015	6.1	0.000	0.000
16	50	25	8	788.215	0.718	0	Opt	0.047	52.2	0.000	0.000
17	50	25	9	800.313	2.137	0	Opt	0.094	114.6	0.000	0.000
18	50	25	13	841.682	1.092	3	Opt	0.125	133.9	0.000	0.000
19	50	25	8	803.185	5.320	67	Opt	0.125	126.3	0.000	0.000
20	50	25	9	725.758	1 667.653	120 779	Opt	0.329	376.1	0.021	0.037
21	100	10	2	1 570.551	0.491	25	Opt	0.001	1.6	0.000	0.000
22	100	10	2	1 572.681	0.358	7	Opt	0.001	1.8	0.000	0.000
23	100	10	3	1 681.155	0.359	1	Opt	0.001	1.4	0.000	0.000
24	100	10	2	1 616.928	1.576	7	Opt	0.001	1.1	0.000	0.000
25	100	10	3	1 699.255	0.483	20	Opt	0.001	1.8	0.000	0.000
26	100	10	4	1 650.381	2.408	24	Opt	0.001	1.3	0.000	0.000
27	100	20	12	1 639.001	2.110	147	Opt	0.040	41.3	0.000	0.000
28	100	20	6	1 589.776	3.572	2 285	Opt	0.062	75.8	0.000	0.000
29	100	20	5	1 612.034	0.826	3	Opt	0.016	15.2	0.000	0.000
30	100	20	9	1 533.836	6.974	2 971	Opt	0.062	63.9	0.000	0.000
31	100	20	9	1 596.712	4.618	2 439	Opt	0.078	92.0	0.000	0.000
32	100	20	6	1 628.994	1.420	18	Opt	0.047	55.4	0.000	0.000
33	100	20	5	1 593.550	6.525	2 830	Opt	0.015	12.7	0.000	0.000
34	100	20	5	1 600.571	3.276	368	Opt	0.025	33.8	0.000	0.000
35	100	25	7	1 556.700	17.289	7 079	Opt	0.012	11.9	0.000	0.000
36	100	25	14	1 643.188	2.699	106	Opt	0.124	134.9	0.000	0.000
37	100	25	9	1 623.482	487.461	25 049	Opt	0.016	8.4	0.000	0.000
38	100	25	13	1 629.059	14.197	8 175	Opt	0.247	249.6	0.016	0.029
39	100	25	11	1 553.270	507.190	68 504	Opt	0.203	197.6	0.009	0.017
40	100	25	9	1 641.288	5.912	442	Opt	0.094	92.7	0.000	0.000
				Average:	73.760	7 325	Opt	0.057	62.8	0.001	0.002

Table 3. Results and comparisons on small-size data set

Instance				HGA				
M	n	m	p	*Best.Sol.*	$t(s)$	*gen*	*agap*(%)	σ(%)
1	200	50	14	3 156.864	3.466	2 916.1	0.011	0.007
2	200	50	14	3 116.857	1.525	1 258.2	0.010	0.007
3	200	50	23	3 263.851	3.077	2 252.0	0.007	0.008
4	200	50	15	3 345.737	1.231	1 094.1	0.000	0.000
5	200	50	23	3 070.048	3.549	2 597.5	0.019	0.012
6	200	50	21	2 988.851	2.818	1 913.4	0.017	0.011
7	200	50	22	3 301.607	3.613	2 203.8	0.013	0.009
8	200	50	16	3 302.545	2.076	1 401.1	0.000	0.000
9	200	100	37	2 784.410	15.702	5 345.2	0.105	0.073
10	200	100	46	3 226.742	21.926	6 752.1	0.003	0.004
11	200	100	43	2 979.639	14.878	4 401.0	0.050	0.043
12	200	100	47	3 084.362	11.859	3 489.6	0.052	0.039
13	200	100	30	2 953.714	11.715	3 997.9	0.102	0.072
14	200	100	29	2 877.726	13.695	4 351.3	0.077	0.061
15	200	100	33	3 202.683	23.886	6 934.1	0.083	0.097
16	500	50	14	8 267.820	2.188	1 555.3	0.001	0.002
17	500	50	14	8 294.660	1.290	902.4	0.000	0.000
18	500	50	13	8 100.883	1.678	1 221.8	0.013	0.010
19	500	50	22	8 244.840	2.898	1 678.2	0.007	0.005
20	500	50	24	8 168.218	3.958	2 243.3	0.022	0.025
21	500	50	23	8 309.416	3.671	2 136.7	0.002	0.001
22	500	50	23	8 071.615	3.797	2 165.3	0.018	0.018
23	500	100	39	8 156.389	40.611	10 826.3	0.069	0.045
24	500	100	26	8 083.815	12.657	4 521.0	0.027	0.042
25	500	100	33	7 707.815	25.287	6 657.1	0.077	0.050
26	500	100	30	8 112.250	13.616	3 605.1	0.019	0.016
27	500	100	30	8 077.388	17.730	5 217.1	0.036	0.032
28	500	100	42	7 982.338	23.616	5 351.2	0.040	0.028
29	500	100	32	8 149.878	28.015	7 502.1	0.001	0.001
30	500	100	32	8 134.182	16.475	5 604.6	0.107	0.062
31	750	50	14	12 289.425	2.608	1 880.7	0.000	0.000
32	750	50	19	12 495.038	2.403	1 304.1	0.000	0.000
33	750	50	17	12 396.373	3.716	1 474.8	0.001	0.000
34	750	50	20	12 349.432	3.329	1 355.8	0.001	0.001
35	750	50	14	12 299.174	3.656	1 878.8	0.002	0.002
36	750	100	35	12 168.721	15.591	4 020.6	0.061	0.042
37	750	100	35	12 216.957	33.509	8 518.7	0.002	0.003
38	750	100	47	11 965.588	25.638	5 583.5	0.047	0.029
39	750	100	38	12 076.371	16.906	4 912.4	0.042	0.031
40	750	100	49	12 069.514	28.816	5 523.7	0.030	0.024
				Average:	11.717	3 713.7	0.029	0.023

Table 4. Results on medium-size data set

Instance				HGA				
L	n	m	p	*Best.Sol.*	$t(s)$	*gen*	*agap*(%)	σ(%)
1	1 000	200	91	16 017.265	38.383	4 708.7	0.019	0.015
2	1 000	200	79	15 956.093	21.642	3 046.8	0.059	0.047
3	1 000	200	75	15 593.404	20.157	2 923.9	0.001	0.001
4	1 000	200	57	16 030.064	24.108	3 996.9	0.138	0.110
5	1 000	200	55	15 872.746	17.578	2 990.9	0.109	0.087
6	1 000	200	93	15 565.459	16.097	2 088.9	0.032	0.026
7	1 500	200	54	24 008.552	33.034	5 348.5	0.029	0.023
8	1 500	200	99	24 520.548	23.500	2 658.8	0.005	0.004
9	1 500	200	73	23 758.690	39.017	5 407.4	0.039	0.032
10	1 500	200	84	23 709.244	32.491	4 091.2	0.049	0.040
11	1 500	300	88	24 651.022	115.382	11 071.6	0.009	0.007
12	1 500	300	105	24 439.681	75.675	6 533.0	0.075	0.060
13	2 000	300	88	32 302.890	91.877	8 443.5	0.022	0.017
14	2 000	300	103	32 880.564	160.157	13 335.5	0.046	0.037
15	2 000	300	112	32 314.458	61.211	4 756.9	0.053	0.042
16	2 000	300	131	31 812.982	45.326	3 159.7	0.026	0.021
17	2 000	300	99	32 600.049	91.841	7 766.4	0.052	0.042
18	2 000	300	134	32 943.928	105.467	7 284.6	0.136	0.109
19	3 000	300	134	47 729.246	63.281	4 096.8	0.064	0.051
20	3 000	300	101	48 909.602	99.736	7 891.7	0.067	0.054
21	3 000	500	202	46 657.849	158.655	6 201.5	0.015	0.012
22	3 000	500	223	46 824.343	232.612	8 502.1	0.046	0.037
23	3 000	500	178	48 055.138	198.754	8 480.8	0.025	0.020
24	3 000	500	225	48 052.535	238.323	8 675.3	0.030	0.024
25	5 000	500	215	79 883.412	236.939	8 406.5	0.033	0.026
26	5 000	500	173	80 022.878	114.822	4 706.5	0.037	0.029
27	5 000	500	142	81 317.030	143.221	6 748.7	0.028	0.022
28	5 000	500	194	82 172.654	272.711	10 201.8	0.022	0.018
29	5 000	500	211	82 503.239	381.215	11 265.1	0.004	0.003
30	5 000	500	214	80 363.777	255.135	8 999.3	0.049	0.039
31	7 500	1 000	291	123 298.725	1 414.616	29 200.4	0.009	0.007
32	7 500	1 000	374	122 422.884	896.066	15 020.2	0.013	0.010
33	7 500	1 000	310	118 628.789	454.943	8 391.4	0.041	0.033
34	7 500	1 000	412	121 882.770	2 151.378	33 211.7	0.034	0.028
35	7 500	1 000	358	122 463.333	1 502.836	26 304.5	0.064	0.055
36	10 000	1 000	447	159 862.069	1 018.791	14 059.3	0.103	0.127
37	10 000	1 000	443	160 085.880	881.016	12 486.9	0.069	0.089
38	10 000	1 000	426	160 990.792	2 679.971	33 520.7	0.077	0.093
39	10 000	1 000	445	159 899.115	1 054.745	15 064.5	0.024	0.035
40	10 000	1 000	322	161 158.793	1 039.416	19 190.0	0.114	0.137
				Average:	412.553	10 005.973	0.047	0.042

Table 5. Results on large-scale data set

indicate that corresponding HEA running times are extremely short. The average running time of the HEA for solving small-size instances presented in Table 3 is 0.057 seconds, which is significantly shorter (around 1 294 times) compared to the CPLEX 12.1. The average gap and standard deviation values are 0.001 % and 0.002 %, which indicates that the proposed HEA provides optimal solutions almost every time within 15 runs with different random seeds.

For the medium size and large-scale data sets mentioned above, no optimal solution is found due to memory or time limit imposed on CPLEX 12.1. In cases when no optimal solution is obtained by CPLEX 12.1, the proposed HEA method provides solutions that are presented in Table 4 (medium size instances) and Table 5 (large-scale instances).

As can be seen from Table 4, the HEA quickly reaches its best solutions for medium-size instances. The average CPU time of the HEA for the medium-size data set is 11.717 s, while the average number of HEA generations is less than 3 714. The average gap and standard deviation are 0.029 % and 0.023 %, respectively, which indicates good stability of the algorithm within all HEA runs.

As expected, for L-type instances involving up to $n = 10\,000$ user nodes, the HEA needed longer execution time, compared to M-type and S-type instances. The reason is in the fact that the Exchange procedure within Local search method is applied more often, due to significantly larger number of nodes in J for large-scale instances. The values in the last row of Table 5 show that the average HEA running time over all L-type instances was 412.553 s. while the average gap of the HEA solution from the best-known one is $agap = 0.047\,\%$ and the standard deviation is $\sigma = 0.042\,\%$. The longest HEA run on the largest considered instance with 10 000 nodes is under 45 min and the average running time on all considered instances with 10 000 nodes is around 22 min, while the values of $agap$ and σ are still low.

The presented results of computational experiments on all three data sets clearly demonstrate the robustness of the proposed hybrid evolutionary-based algorithm with respect to both solutions' quality and running times. The results on L-type instances indicate obvious potential of the HEA for real-life applications.

5 CONCLUSIONS

In this study, we deal with the problem of efficient exploration of data flow within an online social network. We propose the mathematical model of the problem and benchmark it on three data sets containing instances with up to 10 000 user nodes. Since instances with more than 100 user nodes could not be solved to optimality by CPLEX solver, we have designed a hybrid metaheuristic method HEA in order to tackle real-life problem dimensions. In the EA part, binary representation of solutions, an efficient fitness function and fine-grained tournament selection operator are used. A modified crossover operator adopted to the problem under consideration is implemented. An efficient Local search heuristic is proposed to improve the solutions generated by the EA and help in preventing the convergence to a local

optimum. Execution time of the LS heuristic is decreased by implemented strategy for exchanging bits in solution's chromosome. The results of exhaustive computational tests on small, medium and large data set show efficiency and stability of the proposed HEA method. Small and medium-size instances were solved in less than 0.6 and 12 seconds respectively (in average), while average gap and standard deviation were almost 0 for small instances and under 0.3 % for medium ones. In average, the HEA needed less than 413 seconds of CPU time to solve large-size instances, while low values of average gap 0.047 % and standard deviation 0.042 % indicate algorithm's stability. Presented experimental results clearly indicate that the proposed HEA represents a promising metaheuristic method for this and similar problems related to online social networks involving large number of users.

Acknowledgement

This research was partially supported by Serbian Ministry of Education and Science under the grants No. 174010 and 47017.

REFERENCES

[1] Abdella, M.—Marwala, T.: The Use of Genetic Algorithms and Neural Networks to Approximate Missing Data in Database. Computing and Informatics, Vol. 24, 2005, pp. 577–589.

[2] Ahuja, R. K.—Magnanti, T. L.—Orlin, J. B.: Network Flows. Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1993.

[3] Alumur, S.—Kara, B. Y.: Network hub Location Problems: The State of the Art. European Journal of Operational Research, Vol. 190, 2008, No. 1, pp. 1–21.

[4] Bastian, M.—Heymann, S.—Jacomy, M.: Gephi: An Open Source Software for Exploring and Manipulating Networks. 3rd International AAAI Conference on Weblogs and Social Media, 2009. Retrieved from `http://www.aaai.org/ocs/index.php/ICWSM/09/paper/view/154`.

[5] Ball, M. O.—Magnanti, T. L.—Monma, C. L.—Nemhauser, G. L.: Network Models. Handbooks in OR & MS, Vol. 7, Elsevier Science BV, 1995.

[6] Boyd, D.—Ellison, N. B.: Social Network Sites: Definition, History, and Scholarship. Journal of Computer-Mediated Communication, Vol. 13, 2008, pp. 210–230.

[7] Bruns, A.—Kirchhoff, L.—Nicolai, T.—Wilson, J. A.—Saunders, B. J.—Highfield, T.: Network and Concept Maps for the Blogosphere. Discussion paper, 2008. Retrieved from `http://eprints.qut.edu.au/archive/00013580/01/13580.pdf`.

[8] Caers, R.—Castelyns, V.: LinkedIn and Facebook in Belgium: The Influences and Biases of Social Network Sites in Recruitment and Selection Procedures. Social Science Computer Review, Vol. 29, 2011, pp. 437–448.

[9] Campbell, J. F.—Ernst, A.—Krishnamoorthy, M.: Hub Location Problems. In: Hamacher, H.—Drezner, Z. (eds.) Location Theory: Applications and Theory. Springer Verlag Berlin Heidelberg, pp. 373–407.

[10] Chen, Y. W.—Lu, Y. Z.—Yang, G. K.: Hybrid Evolutionary Algorithm With Marriage of Genetic Algorithm and Extremal Optimization for Production Scheduling. The International Journal of Advanced Manufacturing Technology, Vol. 36, 2008, No. 9-10, pp. 959–968.

[11] Drezner, Z.—Hamacher, H. W.: Facility Location Applications and Theory. Springer Verlag, New York, 2002.

[12] Fan, S. K. S.—Liang, Y. C.—Zahara, E.: A Genetic Algorithm and a Particle Swarm Optimizer Hybridized With Nelder Mead Simplex Search. Computers & Industrial Engineering, Vol. 50, 2006, No. 4, pp. 401–425.

[13] Filipović, V.: Fine Grained Tournament Selection Operator in Genetic Algorithms. Computing and Informatics, Vol. 22, 2003, pp. 143–161.

[14] Gaines, B. J.—Mondak, J. J.: Typing Together? Clustering of Ideological Types in Online Social Networks. Journal of Information Technology & Politics, Vol. 6, 2009, pp. 216–231.

[15] Goldberg, D.: Genetic Algorithms in Search, Optimization and Machine Learning. Addison Wesley Press, M. A., 1989.

[16] Fallah, H.—Sadigh, A. N.—Aslanzadeh, M.: Covering Problem. In: Farahani, R. Z.—Hekmatfar, M. (eds.): Facility Location: Concepts, Models, Algorithms and Case Studies. Springer Dordrecht Heidelberg London, New York, 2009, pp. 145–176.

[17] Highfield, T.—Kirchhoff, L.—Nicolai, T.: Challenges of Tracking Topical Discussion Networks Online. Social Science Computer Review, Vol. 29, 2011, No. 3, pp. 340–353.

[18] Huberman, B. A.—Romero, D. M.—Wu, F.: Social Networks That Matter: Twitter Under the Microscope. First Monday, Vol. 14, 2009.

[19] Lerman, K.—Ghosh, R.: Information Contagion: An Empirical Study of the Spread of News on Digg and Twitter Social Networks. 4th International Conference on Weblogs and Social Media (ICWSM 2010). Retrieved from `http://arxiv.org/abs/1003.2664`.

[20] Lewis, K.—Kaufman, J.—Gonzalez, M.—Wimmer, A.—Christakis, N.: Tastes, Ties, and Time: A New Social Network Dataset Using Facebook.com. Social Networks, Vol. 30, 2008, No. 4, pp. 330–342.

[21] Mala, D. J.—Ruby, E.—Mohan, E. V.: A Hybrid Test Optimization Framework-Coupling Genetic Algorithm with Local Search Technique. Computing and Informatics, Vol. 29, 2010, pp. 133–164.

[22] Misevičus, A.—Rubliauskas, D.: Testing of Hybrid Genetic Algorithms for Structured Quadratic Assignment Problems. Informatica, Vol. 20, 2009, No. 2, pp. 255–272.

[23] Moscato, P. A.—Cotta, C.: A Gentle Introduction to Memetic Algorithms. In: Glover, F.—Kochenberger, G. (eds.): Handbook of Metaheuristics. Kluwer Academic Press, Boston, MA, 2003, pp. 105–144.

[24] Neri, F.—Cotta, C.: Memetic Algorithms and Memetic Computing Optimization: A Literature Review. Swarm and Evolutionary Computation, Vol. 2, 2012, pp. 1–14.

[25] Park, H. W.—Thelwall, M.: Developing Network Indicators for Ideological Landscapes from the Political Blogosphere in South Korea. Journal of Computer Mediated Communication, Vol. 13, 2009, pp. 856–879.

[26] Prestwich, S.—Tarim, S.—Rossi, R.—Hnich, B.: Evolving Parameterised Policies for Stochastic Constraint Programming. In: Gent, I. (ed.): Principles and Practice of Constraint Programming CP 2009, Lecture Notes in Computer Science, Springer Verlag Berlin Heidelberg, Germany, Vol. 5732, 2009, pp. 684–691.

[27] Puljić, K.—Manger, R.: A Distributed Evolutionary Algorithm with a Superlinear Speedup for Solving the Vehicle Routing Problem. Computing and Informatics, Vol. 31, 2012, No. 3, pp. 675–692.

[28] ReVelle, C. S.—Eiselt, H. A.: Location Analysis: A Synthesis and Survey. European Journal of Operational Research, Vol. 16, 2005, pp. 1–19.

[29] Schilling, D. A.—Jayaraman, V.—Barkhi, R.: A Review of Covering Problems in Facility Location. Location Science, Vol. 1, 1993, No. 1, pp. 25–55.

[30] Thelwall, M.: Introduction to Webometrics: Quantitative Web Research for the Social Sciences. San Rafael, CA: Morgan & Claypool Publishers, 2009.

[31] Vivekanandan, P.—Rajalakshmi, M.—Nedunchezhian, R.: An Intelligent Genetic Algorithm for Mining Classification Rules in Large Datasets. Computing and Informatics, Vol. 32, 2013, No. 1, pp. 1–22.

[32] Wei, J.: A Hybrid Particle Swarm Evolutionary Algorithm for Constrained Multi-Objective Optimization. Computing and Informatics, Vol. 29, 2010, pp. 701–718.

[33] SocSciBot: `http://socscibot.wlv.ac.uk/`.

[34] IssueCrawler: `http://www.issuecrawler.net/`.

[35] Stanimirović, Z.: A Genetic Algorithm Approach for the Capacitated Single Allocation p-Hub Median Problem. Computing and Informatics, Vol. 29, 2010, pp. 117–132.

Zorica Stanimirović is Assistant Professor at the Department for Numerical Mathematics and Optimization and Vice-Dean for Science and Research at Faculty of Mathematics, University of Belgrade. She received her Ph. D. in optimization from Faculty of Mathematics, Belgrade in 2007. She is a part-time researcher at two scientific projects supported by Serbian Ministry for Education and Science. Her research interests are: combinatorial optimization, location problems, metaheuristic methods, and mathematical modeling.

Stefan Mišković is a Ph. D. student at the Department for Informatics and Computer Science, Faculty of Mathematics, University of Belgrade. At the same faculty, he graduated as a Bachelor of mathematics and computer science in 2010. He defended his Master thesis in 2011. He is working as a teaching assistant at Faculty of Mathematics, University of Belgrade. His research is mainly focused on optimization, evolutionary algorithms and other metaheuristic methods.