Yugoslav Journal of Operations Research
25 (2015), Number 3, 361–378
DOI: 10.2298/YJOR130702032M

METAHEURISTIC APPROACHES TO SOLVING LARGE-SCALE BILEVEL UNCAPACITATED FACILITY LOCATION PROBLEM WITH CLIENTS' PREFERENCES

Miroslav MARIĆ
Faculty of Mathematics, University of Belgrade
maricm@matf.bg.ac.rs

Zorica STANIMIROVIĆ
Faculty of Mathematics, University of Belgrade
zoricast@matf.bg.ac.rs

Nikola MILENKOVIĆ
Faculty of Mathematics, University of Belgrade
nikola.milenkovic@live.com

Aleksandar DJENIĆ
Faculty of Mathematics, University of Belgrade
djenic@matf.bg.ac.rs

Received: July 2013 / Accepted: September 2014

Abstract: In this study, we consider a variant of the Bilevel Uncapacitated Facility Location Problem (BLUFLP), in which the clients choose suppliers based on their own preferences. We propose and compare three metaheuristic approaches for solving this problem: Particle Swarm Optimization (PSO), Simulated Annealing (SA), and a combination of Reduced and Basic Variable Neighborhood Search Method (VNS). We used the representation of solutions and objective function calculation that are adequate for all three proposed methods. Additional strategy is implemented in order to provide significant time savings when evaluating small changes of solution's code in improvement parts. Constructive elements of each of the proposed algorithms are adapted to the problem under consideration. The results of broad computational tests on modified problem instances from the literature show good performance of all three proposed methods, even on large

problem dimensions. However, the obtained results indicate that the proposed VNS-based has significantly better performance compared to SA and PSO approaches, especially when solving large-scale problem instances. Computational experiments on large scale benchmarks demonstrate that the VNS-based method is fast, competitive, and able to find high-quality solutions, even for large-scale problem instances with up to 2000 clients and 2000 potential facilities within reasonable CPU times.

Keywords: Location problems, Discrete optimization, Particle Swarm Optimization, Simulated Annealing, Variable Neighborhood Search.

MSC: 68T20, 90B80, 90B06.

1. INTRODUCTION

In this study, we consider the bilevel uncapacitated facility location problem (BLUFLP), introduced by Hanjoul et al. in [11]. We start from a given set I of M potential sites for locating facilities, and a set J of N clients. The costs of assigning clients to facilities are given by a cost matrix C. The clients' preferences to be served by facilities are defined by a matrix G. Opening a facility at certain location assumes certain additional cost. The problem is to choose the facilities that are to be opened and to assign the clients to the opened facilities so that the sum of fixed costs for opening facilities and the total cost for servicing the clients is minimized. The problem involves two stages of decision making:

- at the upper level, a set of facilities to be opened is chosen,
- at the lower level, the clients are assigned to these facilities by taking into account the clients' preferences.

Several reformulations of the BLUFLP as a single-level location problem with some additional constraints are proposed in Gorbachevskaya [10]. In Hansen et al. [15], the authors introduced another reformulation of the problem, and showed that it dominates the three previous ones from the literature, regarding their linear programming relaxations.

Cánovas et al. [4] developed several valid inequalities for the BLUFLP. They combined them and got preprocessing rules that were applied to the model to obtain a reduced and tighter formulation. Computational experiments were carried out on a modified subset of standard ORLIB instances [2] for the uncapacitated facility location problem, involving up to 100 customers and 75 potential facility locations. The results of the computational study indicate that the proposed approach may be successfully incorporated in a more general framework, such as branch-and-bound algorithms, in order to improve lower bounds and the overall efficiency of the algorithm.

Alekseeva et al. in [1] considered a variant of the problem with a fixed number of facilities to be opened, named the p-median problem with clients' preferences. The authors also designed a genetic algorithm for solving this problem [1]. The

proposed method was benchmarked on a specific set of test instances, which include some essentially sparse matrices of transportation costs, meaning that transportation is not possible for each pair facility–client.

Vasilyev et al. [34] presented a new formulation of the BLUFLP, based on a family of valid inequalities that are related to the problem on a pair of matrices and the set packing problem. A cutting plane method is implemented for calculating the corresponding lower bounds. The proposed cutting plane algorithm was further incorporated in two versions of a branch-and-cut method in order to find an optimal solution. The simulated annealing method is proposed for obtaining the upper bounds of the optimal solution used in the proposed branch-and-cut methods. Numerical experiments are conducted on the same benchmark set as in [4], including test instances of relatively small dimensions ($M \leq 75$, $N \leq 100$). Computational results from [34] approve the efficiency of the implemented branch-and-cut methods when solving the considered BLUFLP test instances.

The purpose of this study is to develop metaheuristic methods that are able to solve BLUFLP instances of real-life dimensions. Inspired by the results presented in [25], we design and implement advanced variants of three metaheuristics: Particle Swarm Optimization, Simulated Annealing and Variable-Neighborhood Search, which are tailored to the BLUFLP. The proposed metaheuristic methods are benchmarked on the generated data set involving up to 2000 clients and 2000 potential facility locations, and the obtained results are analyzed and compared.

The remainder of the paper is organized as follows. Section 2 provides mathematical formulation of the BLUFLP. Section 3 gives a description of the implementation of the proposed metaheuristics for solving the BLUFLP. Results of the computational experiments and comparisons of the algorithms' performance are presented in Section 4. In Section 5, we draw out some conclusions, and give directions for a future work.

2. MATHEMATICAL FORMULATION

The formulation involves the following notation:

$I = \{1, ..., M\}$ is the set of potential facilities;

$J = \{1, ..., N\}$ is the set of clients;

$f_i \geq 0$ is the cost of opening the facility $i \in I$;

$C = [c_{ij}]$, $c_{ij} \geq 0$, $i \in I$, $j \in J$, is the matrix of the production and delivery costs for servicing the clients;

$G = [g_{ij}]$, $g_{ij} \geq 0$, $i \in I$, $j \in J$ is the clients' preference matrix $M \times N$. More precisely, if $g_{i_1 j} < g_{i_2 j}$, then the client j prefers facility i_1 to facility i_2.

Two sets of binary decision variables are used: $x_{ij} \in \{0, 1\}$, $i \in I$, $j \in J$ and y_i, $i \in I$. Variable $x_{ij} \in \{0, 1\}$ takes the value of 1 if the client j is served by facility i,

and 0 otherwise. Variable y_i, $i \in I$ is equal to 1 if the facility i is opened, and 0 otherwise.

Using the notation given above, the problem is formulated as in the study of Hansen et al. [15]:

$$min_y \sum_{i \in I} \sum_{j \in J} c_{ij} x^*_{ij}(y) + \sum_{i \in I} f_i y_i \tag{1}$$

subject to:

$$y_i \in \{0, 1\} \quad \text{for every} \quad i \in I \tag{2}$$

where $x^*_{ij}(y)$ is the optimal solution of the client problem:

$$min_x \sum_{i \in I} \sum_{j \in J} g_{ij} x_{ij} \tag{3}$$

subject to:

$$\sum_{i \in I} x_{ij} = 1 \quad \text{for every} \quad j \in J, \tag{4}$$

$$x_{ij} \leq y_j \quad \text{for every} \quad i \in I, \ j \in J, \tag{5}$$

$$x_{ij} \in \{0, 1\} \quad \text{for every} \quad i \in I, \ j \in J. \tag{6}$$

The objective (1) of the upper level problem minimizes the cost of servicing the clients and opening the facilities. Objective function (3) of the lower level problem guarantees that the clients are served in conformity with their preferences. Constraints (4) ensure that each client is served by exactly one facility. Inequalities (5) indicate that a client can be served only by an open facility. Finally, constraints (2) and (6) indicate binary nature of variables y_i and x_{ij}, respectively.

The BLUFLP is NP-hard in the strong sense. In the case that clients' preferences at the lower level properly correspond to the transportation costs at the upper level, the BLUFLP is equivalent to the well known uncapacitated facility location problem [5]. If there are no fixed costs for opening facilities, i.e. $f_i = 0$ for all $i \in I$, this case of the BLUFLP is still NP-hard in the strong sense. If $c_{ij} = -g_{ij}$ for all $i \in I, j \in J$, the problem can be solved in polynomial time, as it was proved by Hansen et al. in [15].

3. PROPOSED METAHEURISTIC METHODS

We have employed three well-known metaheuristics for solving the BLUFLP: Particle Swarm Optimization, Simulated Annealing, and a Variable-Neighborhood Search based method. All three approaches are adopted to the problem under consideration. In the following subsections, the proposed algorithms will be explained in more details.

3.1. Solution's encoding and objective function calculation

Although these algorithms use different strategies to explore the search space and to look for global minimum, they have two common aspects: solution encoding, and objective function calculation.

All three metaheuristic methods use a binary encoding. Each solution is represented by a binary string of the length $M = |I|$. Each bit in the solution encoding corresponds to one potential facility location. If the bit on the i-th position in the solution encoding takes the value of 1, it means that a facility is located at the i-th location. Zero on the i-th position in the solution encoding indicates that the $i-$th location is not chosen for establishing a facility.

Indices of established facilities are easily obtained from the solution encoding. For each client $j \in J$, we sort the array of potential facilities according to preferences of the client j. This information is stored in the matrix of sorted preferences G_s, where each row corresponds to one client and stores the array of sorted facilities, ordered from the most preferred to the least preferred. By using the matrix of sorted clients' preferences G_s, we speed up the objective function calculation since it is called many times during the run of all three algorithms.

For each client j, we are searching through the j-th row of the matrix G_s, looking for the first established facility in the sorted array assigned to client j. This will obviously be the established facility that client j prefers the most. Corresponding transportation costs are added to the objective function value. In this way, the assignment procedure is performed and the sum of transportation costs is calculated. Finally, the objective function value is obtained by adding opening costs for each established facility to total transportation costs.

The proposed methaheuristic methods often perform the inversion of a bit value in a solution encoding, looking for possible improvements. Since inversion procedure is applied many times, recalculation of the objective function value for every newly created solution will significantly prolong the total running time. For this reason, we have implemented an efficient strategy, which quickly returns objective function value of a new solution, obtained by opening/closing a single facility in the current solution. This strategy performs only partial recalculations of objective function of the current solution and therefore, provides valuable time savings. Similar ideas for accelerating the swap procedure when solving location problems with fixed number of facilities can be found in [1], [12], and [30].

The function *Invert* starts from a given solution x, its objective function $f(x)$ (previously calculated) and the index of facility i to be opened/closed. If the i-th bit in the solution encoding is changed from 0 to 1, this is equivalent to opening facility at location i. Instead of recalculating objective function value, we go through the array of clients and check if the newly opened facility i is more preferable for a client $j \in J$, compared to the one currently assigned. If the answer is yes, we assign the client j to facility i and update transportation costs in the objective function value. Finally, we need to add fixed cost for opening facility at position i.

Similar procedure is performed if the facility at position i is closed by inverting bit value $x(i)$ from 1 to 0. We search the array of clients and identify all clients $j \in J$

that were previously assigned to facility i, which is now closed. For each such a client j, we go through the j-th row of sorted matrix of clients' preferences G_s and find an opened facility i' that the client j prefers the most. We further assign j to i' and update transportation costs. At the end, we subtract fixed cost for opening facility at location i from the objective function value.

The output of the *Invert* function is a new solution x' with inverted bits on position i and its objective function value. The pseudocode of *Invert* procedure is shown in Algorithm 1.

Invert function KwEacheach InputinputOutputoutput Solution x;
Objective value *cost* of the solution x;
Index of facility i to be opened/closed; New solution x';
Objective value of the solution x'; $f_{count} \leftarrow$ the number of active facilities in x $x' \leftarrow x$ $x'(i) = 1$ $x'(i) \leftarrow 0$ $f_{count} \leftarrow f_{count} - 1$ $x'(i) \leftarrow 1$ $f_{count} \leftarrow f_{count} + 1$ f_{count} is 0 $x, cost$

facility i is open

client $j \in J$ j prefers newly opened facility i more than the previously assigned one assign client j to facility i update *cost* with C_{ij}

cost $\leftarrow$ *cost*+f_i facility i is closed client $j \in J$ j was assigned to facility i find the best preferred open facility i' for client j
by using matrix G_s update *cost* with $C_{i'j}$

cost $\leftarrow$ *cost*-f_i

$x', cost$

3.2. Particle Swarm Optimization (PSO)

Particle Swarm Optimization method (PSO) is a nature inspired, population-based metaheuristic that uses the idea of natural swarm intelligence, see [17], [19]. Particles in a swarm behave as simple and non-sophisticated agents that cooperate by an indirect communication medium, and perform movements in the decision space. In the literature, the PSOs have been successfully designed for both continuous and discrete optimization problems [18], [28] and [31].

In the proposed PSO for solving the BLUFLP, a swarm consists of $S = 60$ particles moving around in a M-dimensional search space. The number of particles S is a parameter that may be varied.

Particles are first randomly generated by setting the bit values (0 or 1) of the corresponding particle vectors with certain probability. Each particle p_k, $k = 1, 2, ..., S$ has assigned information on its current position Y_k, the best visited position B_k, and velocity v_k. A position of a particle is represented by a binary vector Y_k of length M and corresponds to a candidate solution in the search space. The velocity of a particle v_k is also a vector of length M with elements that take real values between $[-v_{max}, v_{max}]$, where v_{max} is a pre-determined parameter. If an element of the velocity vector v_k exceeds v_{max} or $-v_{max}$, it will be reset to v_{max} or $-v_{max}$, respectively.

In each iteration of the PSO, each particle successively adjusts its position Y_k in respect to the best position B_k visited by itself and the best position visited by the whole swarm. The flying direction of a particle also depends on a cognitive

learning parameter φ_p, and a social learning parameter φ_g. Parameters φ_p and φ_g represent the attraction that a particle will fly to, either toward its own success or towards the success of the best-positioned particle, respectively. Parameter ω (denoted as *inertia weight*) controls the impact of the previous velocity of a particle on its current velocity. For large values of ω, the impact of the previous velocities will be much higher, while smaller values encourage search around current particle position. The *inertia weight* parameter represents a trade-off between global exploration and local exploitation. The values of variables F_p and F_g are randomly chosen from the interval $(0, 1)$.

Since we are dealing with a discrete problem with binary variables, the velocity of a particle is associated with the probability that a particle's bit will take the value of 1. Therefore, a *sigmoid function* $S(v) = 1/(1 + e^{-v})$ is used to normalize the velocities values into the interval $[0, 1]$ (see paper by Kennedy et al. [18]). A random number u is generated in the same range. If the generated number u is less than the normalized velocity value of a particle $S(v_k[d])$, the decision variable $Y_k[d]$ is initialized to 1, otherwise the value 0 is assigned to $Y_k[d]$, for each dimension $d = 1, 2, .., M$.

If a particle p_k has moved to better position compared to its best local solution, the best local position B_k is updated. Moreover, if the new best local position is better than the best global one, the best global position G of the swarm is updated. The algorithm stops if the best global solution remains unchanged through the maximal number of PSO iterations (stopping criterion). In this implementation, the maximal number of PSO non-improving iterations is set to 10 000.

The basic scheme of the proposed PSO method is presented in Algorithm 2.

Particle Swarm Optimization FloorFloor SigmoidSigmoid particle $p_k, k = 1, \ldots, S$ randomly initialize a particle position vector:
$Y_k[d] \leftarrow U(0, 1) + 0.5$, for each dimension $d = 1, ..., M$ set local best known position to the initial position: $B_k \leftarrow Y_k$ initialize particle velocity: $v_k[d] \leftarrow U(-v_{max}, v_{max})$, for each dimension $d = 1, ..., M$ particle p_k is better than the global best particle g $g \leftarrow p_k$ maximum non improving iterations reached particle $p_k, k = 1, \ldots, S$ dimension $d = 1, \ldots, M$ $F_p \leftarrow U(0, 1)$ $F_g \leftarrow U(0, 1)$ Update particle velocity: $v_k[d] \leftarrow \omega v_k[d] + \varphi_p F_p(B_k[d] - Y_k[d]) + \varphi_g F_g(G[d] - Y_k[d])$ $v_k[d] > v_{max}$ $v_k[d] \leftarrow v_{max}$ $v_k[d] < -v_{max}$ $v_k[d] \leftarrow -v_{max}$ $u \leftarrow U(0, 1)$ $u < v_k[d]$ $Y_k[d] \leftarrow 1$ $Y_k[d] \leftarrow 0$

Y_k better than local best position B_k $B_k \leftarrow Y_k$ local best position B_k better than global G $G \leftarrow B_k$

3.3. Simulated Annealing (SA)

Simulated annealing SA is a single-solution, local-search based optimization technique, inspired by the physical process of the cooling of a metal until it reaches a minimum energy crystalline structure, see [8], [16], and [20]. The key property of the SA method is that it allows hill-climbing moves (which worsen the objective function value) as a mechanism to decrease the probability of converging to a local optimum. It showed to be a simple and efficient method for solving different combinatorial and continuous optimization problems, as in [6], [21], and [33].

In the initialization phase of the proposed SA for the BLUFLP, the number of facilities M, initial temperature $T = t_0$, $t_0 \geq 0$, and a temperature cooling schedule t_k are defined. We also select a repetition schedule, R_k, that defines the number of iterations executed at each temperature t_k. The proposed SA starts with a randomly generated solution x that belongs to the search space. The best found solution of the SA is denoted by x_{global}. Initially, the solution x_{global} is equal to the starting solution x.

The SA further proceeds in several iterations. In the main SA loop, the current solution x is set to the best found solution x_{global}, and repetition counter r is set to 0. In the inner SA loop, we randomly generate a solution x' belonging to a neighborhood of the current solution x. The neighbor solution x' is obtained by inverting bit value on a randomly chosen position in the encoding of the solution x. The procedure *Invert* is used to calculate the objective function value of the new solution x' in an efficient way (see Subsection 3.1). We further calculate the difference in objective function values between the current solution and the generated neighboring solution $\Delta_{x,x'}$. Moves that improve the objective function are always accepted, and the best found solution x_{global} is updated. Otherwise, the chosen neighbor x' is selected with a given probability p, which depends on the current temperature and the amount of degradation $\Delta_{x,x'}$ of the objective function, i.e. $p = exp(-\Delta_{x,x'}/t_k)$. As the algorithm progresses, the probability that we will accept the solution of a lower quality decreases. In this way, we help the algorithm to avoid a local optimum trap. The temperature t_k decreases after every k generation, or after the solution has been improved. The algorithm stops if the solution has not been improved through 200 000 consecutive SA iterations. The basic concept of the proposed SA method is shown in Algorithm 3.

Simulated Annealing InvertInvert RandomRandom RandomInitializeRandomInitialize

select number of facilities M select the temperature change counter k select a temperature cooling schedule t_k select an initial temperature $T = t_0 \geq 0$ select a repetition schedule, R_k, that defines the number of iterations executed at each temperature, t_k

$x \leftarrow M$ $x_{global} \leftarrow x$ maximum non improving iterations reached set repetition counter $r = 0$ $x \leftarrow x_{global}$ $r = R_k$ $i \leftarrow 1, M$ $x' \leftarrow x, i$ x' better than x $x \leftarrow x'$ $x \leftarrow x'$ with probability $exp(-\Delta_{x,x'}/t_k)$ x better than x_{global} $x_{global} \leftarrow x$ break $r \leftarrow r + 1$

$k \leftarrow k + 1$

3.4. Proposed VNS-based method

In this section, we present the combination of the Basic Variable Neighborhood Search Method - VNS [13] and its reduced variant - RVNS [14] for solving the BLUFLP. The VNS metaheuristic showed to be successful in solving various discrete location problems, see for example [3], [9], [22], [26], [27], [32], etc. In our VNS-based approach, the RVNS method is employed to provide a good-quality solution, later used as a starting point of the basic VNS algorithm. Throughout VNS loops, we tend to move to a solution that is relatively far from the current one and to perform a systematic search in the neighborhood of the new solution.

Neighborhoods of a given solution x are defined by taking into account the applied binary encoding of solutions. If the encoding of a solution x' differs from the encoding of a solution x in exactly k bits ($k = 1, 2, ..$), we say that x' belongs to the k-th neighborhood of the solution x, and vice versa.

An initial solution of the RVNS is generated randomly. In the RVNS phase, we try to find an improvement of the current solution in its k-neighborhood ($k = 1, 2, ...$) by inverting the bit on a randomly chosen position in the solution encoding. The procedure *Invert* is used to calculate the objective function value of a newly created solution, such that additional time savings are provided (see Subsection 3.1). The RVNS algorithm runs until we reach the maximal number of iterations without any improvement. The best solution obtained by the RVNS is used as the initial solution of the VNS phase.

In the main VNS loop, we first randomly move to a solution in the k-th neighborhood of the current solution (*Shaking phase*). We apply Local Search procedure on the new solution, trying to find a new local extremum (*Local Search phase*). If the best solution is improved, we perform *Shake* and *Local search* in the k-th neighborhood of this new, improved solution. Otherwise, we keep the current best solution, and change the size of the neighborhood from k to $k + 1$. At the beginning of the VNS part, the value of parameter k_{max} is set to $min(\lfloor M/3 \rfloor, 30)$, where $M = |I|$. The main VNS loop is repeated until the maximal number of VNS iterations without improvement of the best solution is reached (500 in this VNS implementation). The pseudocode of the proposed VNS-based method for the BLUFLP problem is presented in Algorithm 6.

RandomInitializeRandomInitialize InvertInvert RandomRandom MinMin Shake-Shake LocalSearchLocalSearch VNS-based method

RVNS $x \leftarrow M$ $k_{max} \leftarrow 2$

maximum non improving iterations reached $k \leftarrow 1$ $k = k_{max}$ $i \leftarrow 1, M$ $x' \leftarrow x, i$ x' better than x $x \leftarrow x'$ break $k \leftarrow k + 1$ VNS $k_{max} \leftarrow \lfloor M/3 \rfloor, 30$ maximum non improving iterations reached $k \leftarrow 2$ $k = k_{max}$ $x' \leftarrow x, k$ $x'' \leftarrow x'$ x'' better than x $x \leftarrow x''$ break $k \leftarrow k + 1$

4. COMPUTATIONAL RESULTS

In this section, computational results and comparisons of the proposed metaheuristic methods are presented. All three methods were implemented by using .NET Framework. We used CPLEX 12.4 solver to find optimal solutions (if possible). We imposed no time limit on the run of CPLEX solver. All tests were run on an Intel Core i7-860 2.8 GHz (quad-core processor) with 8GB RAM memory under Windows 7 Professional operating system.

Computational experiments are carried out on four data sets from the literature, adapted for the BLUFLP, and three randomly generated data sets. Each proposed metaheuristic method was run 20 times on each tested instance. Parameter values used in metaheuristics were tuned experimentally in order to provide best performance of an algorithm in the sense of solution quality.

We have first considered test instances from the literature, originally introduced for single and multiple level uncapacitated facility location problem, which involve up to $N = 2000$ clients $M = 2000$ potential facilities:

- **Data set 1:** Standard ORLIB data set from [2], initially designed for the UFLP. This data set contains small and medium size test problems with $50 \leq N \leq 1000$ clients, and $16 \leq M \leq 100$ potential facilities;
- **Data set 2:** Modified UFLP instances, generated from standard ORLIB data set. These instances are proposed in [23], and include $50 \leq N \leq 1000$ clients and potential facilities;
- **Data set 3:** M^* data set introduced in [29]. It contains large scale instances with $300 \leq N \leq 2000$ and $300 \leq M \leq 2000$;
- **Data set 4:** Modified M^* instances from [23]. This is a challenging large-scale data set involving $300 \leq n \leq 2000$ clients, and $300 \leq m \leq 2000$ potential facilities.

Instances in the used Data sets 1-4 contain fixed costs f_i for establishing a facility at a position $i \in I$. Transportation costs are used as the costs c_{ij} for servicing a client j by facility i for each pair $i \in I$, $j \in J$, while clients' preference matrix is created in respect to distance matrix. For each client $j \in J$, the array of distances d_{ij}, $i \in I$ is sorted in ascending order, and then small perturbations of the sorted array are performed, i.e. 5-10% of the elements in the array exchange positions. In this way, we obtain the array $d_{i_1 j}, \ldots d_{i_M j}$ which defines client's preferences $g_{p_1 j} = 1$,...,$g_{p_M j} = M$. This procedure is performed for each client $j \in J$. For some instances from the data sets, different perturbation levels are applied, resulting in several modifications of the original instance for the BLUFLP.

Note that data sets used for computational studies in [4] and [34] were obtained by modifying the ORLIB instances [2] for the UFLP of smaller dimensions, while the customers' preferences are generated by using uniform distribution [4]. These instances were separated into three blocks: the first block contains the problems with $N = 50$ customers and $M = 50$ potential facility locations; the second block consists of the problems of size $N = 75$ and $M = 50$; and the third block involves instances with $N = 100$ and $M = 75$. Unfortunately, these test instances were unavailable to us, and therefore we could not perform exact comparisons. Since the dimensions of instances used in [4] and [34] are relatively small, these test problems do not represent a challenge for the metaheuristic method proposed in this paper.

In Table 1, we provide computational results and comparisons of the considered methods on instances from Data sets 1-4. For each instance, we present optimal solution obtained by CPLEX 12.4 solver and its running time ("-" stands if no optimal solution was found). For each of the three proposed metaheuristics, the results are presented as follows.

(i) The best solution *Best.Sol* on the current instance. If *Best.Sol* is equal to optimal solution obtained by CPLEX, *opt* is written. Note that the optimal solutions and the best results of all three metaheuristics are bolded in Tables 1-4.

(ii) Average running time $t(s)$, in which the algorithm reaches optimal solution or obtains its best solution (in seconds).

(iii) Average gap *agap*(%) from the optimal/best solution through 20 runs. Average gap is calculated as $agap = \frac{1}{20}\sum_{i=1}^{20} gap_i$, where $gap_i = 100 \cdot \frac{|sol_i - Best.Sol|}{|Best.Sol|}$, $i = 1, .., 20$ and *Best.Sol* is the optimal/best solution on the current instance.

(iv) Standard deviation σ(%) from the optimal/best solution through 20 runs, obtained as $\sigma = \sqrt{\frac{1}{20}\sum_{i=1}^{20}(gap_i - agap)^2}$.

As it can be seen from Table 1, the CPLEX 12.4 was able to solve only instances with up to 50 clients, and 50 potential facilities to optimality. The proposed VNS-based approach outperformed the PSO and the SA in the sense of solutions quality: it reached all optimal solutions and provided best solutions for all test instances. The values in *agap*(%) column indicate its high reliability in providing all optimal/best solutions. As the $t(s)$ column shows, in some cases the VNS method is slower compared to PSO or SA approaches, but its running times are still relatively short, even for the largest tested instances with 2000 nodes. The proposed PSO shows the worst performance regarding solutions quality, but its running times are short, while the SA showed slightly better performance compared to PSO.

We have further benchmarked our methods on three newly generated sets of instances, which include up to $N = 2000$ clients and $M = 2000$ facilities. The instances are created in a similar way as done in a study by Hansen et al. [15]. In a square $[0, N] \times [0, N]$ in two-dimensional plane, we randomly choose N points for locations of clients and M points for locations of facilities. Fixed costs for establishing a facility $j \in J$ is equal to $f_j = \frac{\sqrt{N}}{10}$, while the elements of cost matrix are obtained as $c_{ij} = d_{ij}$. Following the idea from [15], clients' preferences are initially sorted according to their corresponding distances, and then we perform some random changes by using three different strategies, producing three new benchmark sets.

- **Data set 5:** In this data set, for each client $j \in J$, we randomly choose $\lfloor \frac{M}{8} \rfloor$ pairs of facilities and exchange their preferences for the client j. In this way, around 25% of preferences are changed for each client.

- **Data set 6:** In this group of instances, for each client $j \in J$, we randomly choose the subset of $\lfloor \frac{M}{10} \rfloor$ facilities and assign them randomly generated preference values that are between minimal and maximal preference value for this client.

Table 1: Results and comparisons on Data sets 1-4

Inst-N-M	CPLEX		PSO				SA				VNS			
	Opt.Sol.	t(s)	Best.Sol.	t(s)	$agap$(%)	σ(%)	Best.Sol.	t(s)	$agap$(%)	σ(%)	Best.Sol.	t(s)	$agap$(%)	σ(%)
cap-50-16	1248142.9	0.172	**opt**	0.014	3.450	1.992	**opt**	0.007	0.000	0.000	**opt**	0.008	0.000	0.000
cap-50-25	1248142.9	0.828	1270249.925	0.026	1.891	0.523	**opt**	0.047	0.000	0.000	**opt**	0.020	0.443	0.767
cap-50-50	1248142.9	0.409	**opt**	0.093	13.959	12.624	**opt**	0.047	0.000	0.000	**opt**	0.014	0.000	0.000
capa-1000-100	-	-	**24147833.770**	0.284	0.000	0.000	**24147833.770**	0.162	0.000	0.000	**24147833.770**	0.036	0.000	0.000
capb-1000-100	-	-	**22705728.388**	0.339	0.205	0.614	**22705728.388**	0.194	0.000	0.000	**22705728.388**	0.031	0.000	0.000
capc-1000-100	-	-	**22204075.489**	0.321	0.331	0.752	**22204075.489**	0.343	0.000	0.000	**22204075.489**	0.057	0.000	0.000
mq1-300-300	-	-	**4820.990**	0.748	0.117	0.335	**4820.990**	10.633	0.000	0.000	**4820.990**	0.114	0.000	0.000
mr1-500-500	-	-	**3686.929**	2.178	0.408	0.467	**3686.929**	103.019	0.139	0.253	**3686.929**	0.186	0.024	0.103
ms1-1000-100	-	-	**7159.554**	9.186	0.586	0.611	**7159.554**	391.812	0.530	0.526	**7159.554**	0.571	0.024	0.071
mt1-2000-2000	-	-	14822.496	60.105	25.891	22.114	**14796.344**	1039.134	0.893	0.452	**14796.344**	6.416	0.062	0.084
md0-100-100	-	-	**559418.823**	0.148	2.917	1.960	**559418.823**	0.124	1.504	1.280	**559418.823**	0.016	0.000	0.000
md1-100-100	-	-	561768.823	0.153	3.024	1.260	**557000.568**	0.046	1.094	0.804	**557000.568**	0.025	0.259	0.348
md2-100-100	-	-	562082.736	0.110	2.904	0.747	554583.354	0.069	1.522	0.798	**553797.540**	0.042	0.034	0.103
md0-200-200	-	-	1149737.938	0.459	4.056	1.049	1127644.647	0.271	2.574	1.016	**1122632.460**	1.046	0.687	0.504
md1-200-200	-	-	1138336.196	0.594	4.798	1.211	1115991.054	0.216	3.003	1.522	**1110665.699**	0.903	0.567	0.422
md2-200-200	-	-	1124097.708	0.650	3.919	1.060	1122353.595	0.288	2.874	0.874	**1102880.789**	0.473	0.879	0.603
md0-300-300	-	-	1706456.503	1.329	3.525	0.969	1673291.934	0.508	2.966	1.327	**1661657.633**	0.199	0.699	0.570
md1-300-300	-	-	1689891.715	1.017	3.360	1.177	1690701.489	0.386	2.290	0.579	**1667518.851**	1.317	0.999	0.599
md2-300-300	-	-	1661295.583	1.377	2.573	1.114	1658392.177	0.704	2.104	0.798	**1648977.107**	2.211	0.531	0.424
md0-500-500	-	-	2827363.965	3.235	2.728	0.949	2810201.942	1.207	2.127	0.768	**2787568.582**	11.995	0.935	0.644
md1-500-500	-	-	2800942.512	3.937	3.819	0.970	2773501.349	2.289	2.723	1.104	**2753812.000**	8.212	1.415	0.615
md2-500-500	-	-	2824298.465	3.316	3.573	0.890	2803407.896	2.146	2.366	0.904	**2782127.139**	2.317	1.408	0.793
md0-1000-1000	-	-	5608973.680	21.912	2.946	0.643	5601373.986	8.750	2.107	0.519	**5526194.156**	89.119	0.940	0.376
md1-1000-1000	-	-	5594718.762	18.381	2.916	0.660	5548414.188	7.408	1.996	0.662	**5501359.273**	21.405	1.082	0.454
md2-1000-1000	-	-	5667948.877	18.134	2.310	0.624	5611827.823	7.356	1.335	0.617	**5605783.436**	71.544	0.735	0.440
md0-2000-2000	-	-	11395587.703	220.616	2.596	0.533	11296896.489	48.508	1.941	0.440	**11201602.43**	238.201	0.874	0.427
md1-2000-2000	-	-	11457550.357	164.476	2.337	0.438	11343445.599	46.449	1.753	0.388	**11245868.168**	459.582	0.648	0.361
md2-2000-2000	-	-	11268173.593	227.294	2.403	0.634	11254432.882	42.346	1.859	0.433	**11128587.883**	172.829	0.721	0.365

- **Data set 7:** In this data set, preferences are multiplied with the values obtained by normal distribution $N(0, 0.2)$ on the interval $[-1, 1]$, i.e. $g_{ij} = d_{ij}f(r_{ij}; 0, 0.2)$, where r_{ij} is randomly chosen value from $[-1, 1]$, and f is corresponding Gaussian function.

Tables 2-4 show the results of conducted computational experiments on Data sets 5-7. As it can be seen from Tables 2-4, all three metaheuristics reached optimal solutions, previously obtained by CPLEX solver in significantly shorter CPU time. The exception is PSO method, which failed to obtain optimal solution for instance $N = 100$, $M = 50$ from Data set 7 (see Table 4). On other instances from Data sets 5-7 that were out of reach for CPLEX, all three proposed methods provided solutions in relatively short CPU time, considering problem dimensions, but the quality of the obtained solutions is different. From columns *Best.Sol* in Tables 2-4, it can be seen that the VNS outperforms both PSO and SA in the sense of solution quality. On all considered instances from Data sets 5-7, the VNS produced the best solutions compared to two other methods, with exception of several cases in which the PSO and SA produced the same solution as the VNS. The percentage of the obtained best solutions of PSO, SA, and VNS methods on Data sets 5-7 is as follows.

- **Data set 5:** PSO 50% , SA 75% and VNS 100%;
- **Data set 6:** PSO 45% , SA 65% and VNS 100%;
- **Data set 7:** PSO 47% , SA 53% and VNS 100%.

From the results presented in Tables 2-4, we can see that the VNS showed good stability through all 20 runs when solving instances from Data sets 5-7. Although clients' preferences in these data sets were generated by different strategies, which ensure various perturbation levels of initial preferences, the VNS showed excellent performance in all three cases. Low values of average gap *agap*(%) and standard deviation σ(%) presented in Tables 2-4 clearly indicate good stability and reliability of the proposed VNS approach for solving the BLUFLP.

Regarding columns $t(s)$ in Tables 2-4, we may notice that on smaller size instances, the VNS obtains the best solutions in shortest CPU time compared to both SA and PSO. On larger problem dimensions, the proposed PSO method has, generally, the shortest running times (but the worst solutions' quality), while the VNS performs faster compared to the SA approach. In general, we may conclude that the running times of all three proposed metaheuristics are reasonably short, having in mind the dimensions of considered problem instances.

Table 2: Results and comparisons on Data set 5

N	M	CPLEX		PSO				SA				VNS			
		Opt.Sol.	t(s)	Best.Sol.	t(s)	$agap$(%)	σ(%)	Best.Sol.	t(s)	$agap$(%)	σ(%)	Best.Sol.	t(s)	$agap$(%)	σ(%)
50	16	489.062	0.155	**opt**	0.006	0.000	0.000	**opt**	0.021	0.000	0.000	**opt**	0.008	0.000	0.000
50	25	382.334	0.216	**opt**	0.012	0.000	0.000	**opt**	0.039	0.000	0.000	**opt**	0.009	0.000	0.000
100	33	1446.460	0.466	**opt**	0.027	0.000	0.000	**opt**	0.075	0.000	0.000	**opt**	0.009	0.000	0.000
50	50	299.290	0.346	**opt**	0.044	0.000	0.000	**opt**	0.090	0.000	0.000	**opt**	0.012	0.000	0.000
100	50	1045.440	0.755	**opt**	0.055	1.704	2.060	**opt**	0.092	0.000	0.000	**opt**	0.013	0.000	0.000
100	100	681.343	2.232	**opt**	0.180	1.365	0.829	**opt**	2.964	0.064	0.126	**opt**	0.016	0.000	0.000
500	100	24627.296	62.915	**opt**	0.297	0.618	0.671	**opt**	0.808	0.191	0.572	**opt**	0.103	0.000	0.000
500	160	-	-	18655.531	18.844	2.387	1.037	**18510.343**	47.256	1.612	1.040	**18510.343**	2.178	0.000	0.000
1000	200	-	-	83071.905	69.863	2.322	1.118	82611.959	120.244	1.864	0.930	**82385.578**	5.697	0.413	0.369
500	250	-	-	12682.472	32.177	3.141	1.844	**12675.028**	52.331	1.715	0.982	**12675.028**	4.246	0.248	0.590
1000	330	-	-	58809.450	144.446	2.992	1.160	58951.865	60.797	2.804	1.384	**58357.067**	95.31	0.375	0.356
500	500	-	-	8764.589	169.026	3.947	1.573	8696.023	48.427	4.022	0.997	**8592.339**	12.599	2.150	0.974
1000	500	-	-	45214.891	68.394	6.387	2.621	46086.846	372.246	7.004	1.985	**44511.336**	391.811	0.608	0.641
2000	500	-	-	248049.522	251.534	4.551	1.660	249465.893	1235.423	4.816	1.589	**242959.314**	666.921	0.705	0.535
1000	1000	-	-	24852.396	480.221	4.552	1.429	24243.553	752.734	4.029	1.604	**24231.653**	803.082	1.862	1.353
2000	1000	-	-	132320.259	316.826	8.466	2.500	135726.286	1748.054	7.392	1.561	**128934.553**	6820.102	1.915	1.021
2000	2000	-	-	72613.203	1334.132	9.901	1.935	70846.735	2283.885	9.599	3.576	**68171.714**	3880.871	4.907	2.749

Table 3: Results and comparisons on Data set 6

N	M	CPLEX		PSO				SA				VNS			
		Opt.Sol.	$t(s)$	Best.Sol.	$t(s)$	$agap(\%)$	$\sigma(\%)$	Best.Sol.	$t(s)$	$agap(\%)$	$\sigma(\%)$	Best.Sol.	$t(s)$	$agap(\%)$	$\sigma(\%)$
50	16	383.219	0.184	**opt**	0.008	0.000	0.000	**opt**	0.011	0.000	0.000	**opt**	0.008	0.000	0.000
50	25	308.805	0.246	**opt**	0.016	0.000	0.000	**opt**	0.045	0.000	0.000	**opt**	0.009	0.000	0.000
100	33	1072.120	0.425	**opt**	0.034	0.000	0.000	**opt**	0.126	0.000	0.000	**opt**	0.009	0.000	0.000
50	50	200.371	0.353	**opt**	0.055	0.087	0.119	**opt**	0.213	0.000	0.000	**opt**	0.011	0.000	0.000
100	50	756.914	0.766	**opt**	0.056	0.000	0.000	**opt**	0.495	0.000	0.000	**opt**	0.011	0.000	0.000
100	100	602.470	2.185	**opt**	0.181	0.000	0.000	**opt**	0.816	0.000	0.000	**opt**	0.018	0.000	0.000
500	100	15739.108	22.183	**opt**	0.337	0.000	0.000	**opt**	4.762	0.009	0.027	**opt**	0.033	0.000	0.000
500	160	12478.805	45.917	**opt**	6.181	3.256	1.900	**opt**	15.759	2.084	0.963	**opt**	0.729	0.549	0.950
1000	200	-	-	**46245.730**	1.101	0.000	0.000	**46245.730**	1.472	0.000	0.000	**46245.730**	0.121	0.000	0.000
500	250	-	-	9445.287	8.692	1.678	0.394	**9445.020**	10.272	1.658	0.380	**9445.020**	2.035	0.785	0.868
1000	330	-	-	**35647.710**	48.820	0.934	0.595	**35647.710**	108.828	1.168	0.874	**35647.710**	7.760	0.016	0.072
500	500	-	-	7149.432	64.587	6.631	2.426	7147.914	248.784	5.073	1.592	**6956.374**	48.749	2.900	1.112
1000	500	-	-	29051.094	73.348	2.762	1.220	29069.179	222.355	3.073	1.059	**28906.628**	731.573	1.119	0.608
2000	500	-	-	137455.552	103.227	1.839	0.722	137213.970	474.519	1.478	0.559	**136697.672**	89.115	0.674	0.397
1000	1000	-	-	21508.861	446.732	6.227	1.547	21577.499	1105.461	6.379	1.846	**20757.159**	1225.038	4.264	2.071
2000	1000	-	-	97308.115	222.576	4.692	1.263	97721.838	788.968	5.068	1.369	**95177.507**	4262.441	2.835	1.272
2000	2000	-	-	84828.559	610.593	6.963	2.101	83060.457	2496.242	6.510	2.476	**81593.345**	4688.081	5.548	2.697

Table 4: Results and comparisons on Data set 7

N	M	CPLEX		PSO				SA				VNS			
		Opt.Sol.	$t(s)$	Best.Sol.	$t(s)$	$agap(\%)$	$\sigma(\%)$	Best.Sol.	$t(s)$	$agap(\%)$	$\sigma(\%)$	Best.Sol.	$t(s)$	$agap(\%)$	$\sigma(\%)$
50	16	648.697	0.381	**opt**	0.010	0.000	0.000	**opt**	0.016	0.000	0.000	**opt**	0.010	0.000	0.000
50	25	438.800	0.572	**opt**	0.018	0.112	0.275	**opt**	0.025	0.000	0.000	**opt**	0.020	0.000	0.000
100	33	1601.799	1.517	**opt**	0.026	0.000	0.000	**opt**	0.044	0.000	0.000	**opt**	0.009	0.000	0.000
50	50	312.208	0.697	**opt**	0.081	1.395	1.342	**opt**	0.077	0.000	0.000	**opt**	0.014	0.000	0.000
100	50	1260.401	4.958	1263.813	0.062	0.816	0.916	**opt**	0.084	0.000	0.000	**opt**	0.011	0.000	0.000
100	100	885.697	6.631	**opt**	0.248	0.410	0.173	**opt**	0.635	0.260	0.060	**opt**	0.159	0.000	0.000
500	100	24580.796	242.911	**opt**	4.045	0.056	0.041	**opt**	5.168	0.001	0.005	**opt**	0.057	0.000	0.000
500	160	-	-	**18211.576**	46.407	0.115	0.075	**18211.576**	2.992	0.050	0.042	**18211.576**	0.881	0.003	0.011
1000	200	-	-	**63644.060**	41.304	0.139	0.117	**63644.060**	2.847	0.088	0.115	**63644.060**	1.265	0.000	0.000
500	250	-	-	13791.713	46.555	0.856	0.379	13791.713	28.756	0.774	0.390	**13786.470**	17.554	0.049	0.137
1000	330	-	-	48908.837	57.080	0.282	0.103	48882.177	114.904	0.264	0.121	**48856.078**	59.315	0.032	0.026
200	500	-	-	157220.468	193.478	0.162	0.073	157146.871	619.103	0.173	0.084	**157120.856**	139.351	0.006	0.005
500	500	-	-	9379.777	248.546	1.018	0.325	9366.054	194.565	1.074	0.335	**9315.699**	127.354	0.215	0.147
1000	500	-	-	38188.045	239.389	0.418	0.177	38229.335	303.063	0.414	0.182	**38158.741**	120.820	0.038	0.032
1000	1000	-	-	26854.283	907.240	1.053	0.274	26857.898	845.434	1.051	0.219	**26678.550**	1650.404	0.163	0.121
2000	1000	-	-	110784.751	858.785	0.840	0.190	110693.008	1482.923	0.667	0.206	**110352.541**	4641.535	0.160	0.114
2000	2000	-	-	75741.422	1566.246	1.665	0.317	75521.000	5024.576	1.366	0.289	**74886.285**	15132.015	0.339	0.217

5. CONCLUSIONS

In this paper, we proposed the Particle Swarm Optimization, Simulated Annealing, and a combination of Reduced and Basic Variable neighborhood search method for solving the Bilevel Uncapacitated Facility Location Problem - BLUFLP. Binary solution encoding was used in all three methods, and an efficient strategy for calculating objective function was implemented. These common elements represent a good basis for possible future hybridization of these methods. Other constructive elements of the proposed metaheuristic approaches were adapted to the problem under consideration. All three metaheuristics were subject to broad computational experiments on several modified data sets from the literature and newly generated data sets with up to 2000 clients and 2000 potential facilities. The obtained results indicate that the combination of Reduced and Basic VNS outperforms both PSO and SA methods, especially in the cases of large-scale problem instances. The VNS-based method shows excellent performance regarding solution quality, stability and running times. It obviously represents a promising metaheuristic for solving the BLUFLP, and has a potential to be applied to solving similar facility location problems.

Acknowledgements: This research was partially supported by Serbian Ministry of Education, Science and Technological Development under the grants no. 174010 and 47017.

REFERENCES

[1] Alekseeva, E.V., and Kochetov, Y.A., "Genetic Local Search for the p-Median Problem with Clients' Preferences", *Diskret. Anal. Issled. Oper.*, 14 (2007) 3–31.
[2] Beasley, J.E., "Obtaining test problems via internet", *Journal of Global Optimization*, 8 (1996) 429–433.
[3] Brimberg, J., Mladenović, N., Urošević, D., and Ngai, E., "Variable neighborhood search for the heaviest k-subgraph", *Computers and Operations Research*, 36 (2009) 2885–2891.
[4] Cánovas, L., García, S., Labbé, M., and Marín, A., "A Strengthened Formulation for the Simple Plant Location Problem with Order", *Operations Research Letters*, 35 (2007) 141–150.
[5] Cornuejols, G., Nemhauser, G.L., and Wolsey, L.A., "The uncapacitated facility location problem", In: Mirchandani, P.B., and Francis, R.L. (eds.), *Discrete Location Theory*, New York: Wiley-Interscience, 1990, 119–171.
[6] Dekkers, A., and Aarts, E., "Global optimization and simulated annealing", *Mathematical Programming*, 50 (1991) 367–393.
[7] Dempe, S., *Foundations of bilevel programming*, Dordrecht: Kluwer Academic Publishers, 2002.
[8] Eglese, R.W., "Simulated annealing: a tool for operational research", *European Journal of Operational Research*, 46 (1990) 271–281.
[9] García-López, F., Melián-Batista, B., Moreno-Pérez, J.A., and Moreno-Vega, J.M., "The parallel variable neighborhood search for the p-median problem", *Journal of Heuristics*, 8 (2002) 375–388.
[10] Gorbachevskaya, L.E., *Polynomially solvable and NP-hard bilevel standardization problems*, PhD Thesis, Sobolev Institute of Mathematics, Novosibirsk, (in Russian), 1998.
[11] Hanjoul, P., and Peeters, D., "A Facility Location Problem with Clients' Preference Ordering", *Regional Science and Urban Economies*, 17 (1987) 451–473.
[12] Hansen, P., and Mladenović, N., "Variable neighborhood search for the p-median", *Location Science*, 5 (1997) 207–226.
[13] Hansen, P., and Mladenović, N., "An Introduction to Variable Neighborhood Search", In: Voss S., Martello, S., Osman, I.H., and Roucairol, C. (eds.), *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, Kluwer Academic Publishers, 1999, 433–458.

[14] Hansen, P., and Mladenović, N., "Variable neighborhood search: Principles and applications", *European Journal of Operational Research*, 130 (2001) 449–467.
[15] Hansen, H., Kochetov, Y.A., and Mladenović, N., "Lower bounds for the uncapacitated facility location problem with user preferences", *Preprint G – 2004 – 24, GERAD-HEC, Montreal*, 2004.
[16] Henderson, D., Sheldon, H., Jacobson, H., and Johnson, A. W, "The Theory and Practice of Simulated Annealing", In: Glover F., and Kochenberger, G.A., (eds.), *Handbook of metaheuristics*, New York, Boston, Dordrecht, London, Moscow: Kluwer Academic Publishers, (2003) 287–321.
[17] Kennedy, J., and Eberhart, R.C., "Particle swarm optimization", *Proceedings of the IEEE International Conference on Neural Networks, Perth, Australia*, (1995) 1942–1948.
[18] Kennedy, J., and Eberhart, R.C., "A discrete binary version of the particle swarm algorithm", *Proceedings of the IEEE International Conference Systems, Man and Cybernetic*, (1997) 4104–4108.
[19] Kennedy, J., and Eberhart, R.C. *Swarm Intelligence*, San Francisco, CA: Morgan-Kaufmann, 2001.
[20] Kirkpatrick, S., Gelatt, C.D., and Vecchi, M.P., "Optimization by simulated annealing", *Science*, 220 (1983) 671–680.
[21] Koulamas, C., Anthony, S.R., and Jaen, R., "A survey of simulated annealing application to operations-research problems", *OMEGA International Journal of Management*, 22 (1994) 41–56.
[22] Liang, Y.C., Lo, M.H., and Chen, Y.C., "Variable neighbourhood search for redundancy allocation problems", *IMA Journal of Management Mathematics*, 18 (2007) 135–155.
[23] Marić, M., "An efficient genetic algorithm for solving the multi-level uncapacitated facility location problem", *Computing and Informatics*, 29 (2010) 183–201.
[24] Marić M., Stanimirović, Z., and Stanojević, P., "An efficient memetic algorithm for the uncapacitated single allocation hub location problem", *Soft Computing*, 17 (2013) 445–466.
[25] Marić M., Stanimirović, Z., and Milenković, N., "Metaheuristic Methods for Solving the Bilevel Uncapacitated Facility Location Problem with Clients Preferences", *Electronic Notes in Discrete Mathematics*, 39 (2012) 43–50.
[26] Mladenović N., Tododijević, R., and Urošević D., "An efficient General Variable Neighborhood Search for large Travelling Salesman Problem with Time Windows", *The Yugoslav Journal of Operations Research*, 23 (2013), *DOI* : 10.2298/*YJOR*120530015*M*.
[27] Pérez Pérez, M., Almeida Rodríguez, F., and Moreno-Vega, J.M., "A hybrid VNS–path relinking for the p-hub median problem", *IMA Journal of Management Mathematics*, 18 (2007) 157–171.
[28] Pulido, G.T., Coello, C.A.C., and Lechuga, M.S., "Handling multiple objectives with particle swarm optimization", *IEEE Transactions on Evolutionary Computation*, 8 (2004) 256–279.
[29] Raidl, G.R., and Gottlieb, J., "Empirical analysis of locality, heritability and heuristic bias in evolutionary algorithms: A case study for the multidimensional knapsack problem", *Evolutionary Computation*, 13 (2005) 441–475.
[30] Resende, M.G.C., and Werneck, R.F., "A fast swap-based local search procedure for location problems", *Annals of Operations Research*, 150 (2007) 205–230.
[31] Reyes-Sierra, C., and Coello, C.A.C., "Multi-objective particle swarm optimizers: A survey of the state-of-the-art", *International Journal of Computational Intelligence Research*, 2 (2006) 287–308.
[32] Schwengerer, M., Pirkwieser, S., and Raidl, G., "A variable neighborhood search approach for the two-echelon location-routing problem", *Evolutionary Computation in Combinatorial Optimization*, (2012) 13–24.
[33] Suman, B., and Kumar, P., "A survey of simulated annealing as a tool for single and multiobjective optimization", *Journal of the Operational Research Society*, 57 (2006) 1143–1160.
[34] Vasilev, I.L., and Klimentova, K.B., "The Branch and Cut Method for the Facility Location Problem with Clients' Preferences", *Journal of Applied and Industrial Mathematics*, 4 (2010) 441–454.
[35] Vasilev, I.L., Klimentova, K.B., and Kochetov, Y.A., "New Lower Bounds for the Facility Location Problem with Clients' Preferences", *Computational Mathematics and Mathematical Physics*, 49 (2009) 1010–1020.