INFORMATICA, 2014, Vol. 25, No. 3, 439–466

DOI: http://dx.doi.org/10.15388/Informatica.2014.23

Memetic Algorithm for Solving the Multilevel Uncapacitated Facility Location Problem

Miroslav MARIĆ, Zorica STANIMIROVIĆ*, Aleksandar DJENIĆ, Predrag STANOJEVIĆ
Faculty of Mathematics, University of Belgrade
Studentski trg 16/IV, 11 000 Belgrade, Serbia
e-mail: maricm@matf.bg.ac.rs, zoricast@matf.bg.ac.rs, djenic@matf.bg.ac.rs, djapedjape@gmail.com

Received: August 2012; accepted: March 2014

Abstract. We consider the Multilevel Uncapacitated Facility Location Problem (MLUFLP) and propose a new efficient integer programming formulation of the problem that provides optimal solutions for the MLUFLP test instances unsolved to optimality up to now. Further, we design a parallel Memetic Algorithm (MA) with a new strategy for applying the local search improvement within the MA frame. The conducted computational experiments show that the proposed MA quickly reaches all known optimal and best known solutions from the literature and additionally improves several solutions for large-scale MLUFLP test problems.

Key words: facility location, network design, hierarchical problems, memetic algorithm, parallelization.

1. Introduction

There are many papers in the literature dealing with multilevel location problems, due to numerous areas of their applications. Multilevel facility location models are adequate for networks in which facilities to be located have certain common properties, but also some important differentiating feature (for example, a different kind of service that they offer), which allows us to group them into levels. These facilities interact with each other, so that it is not possible to locate facilities in each level independently from other levels.

Multilevel location problems often arise when modeling supply chains, transportation networks, postal or other delivery networks, energy distribution networks, etc. In these problems, it is necessary to design a hierarchical distribution network, i.e. to locate warehouses, suppliers or distribution centers on different network levels and to assign a supply path to each user, such that the network's efficiency is maximized and transportation (or other) costs reduced to a minimum. Therefore, these problems are often named hierarchical location problems in the literature. Multilevel networks also appear in the public sector, for example in education system, health service system, multilevel organization of

*Corresponding author.

bank units, etc. Telecommunications are another important application area of hierarchical location problems, i.e. designing mobile communication networks, computer networks, Internet and satellite communication, etc. In these networks, different sets of facilities are required, equipped with different devices and carrying out different tasks. Traffic (e.g. data, signals) is routed via facilities located on different network levels in order to reach an access node.

The Multilevel Uncapacitated Facility Location Problem is a generalization of the well-known simple plant location problem, which is one of the fundamental and most studied models in facility location theory (Drezner and Hamacher, 2002; ReVelle and Eiselt, 2005).

The MLUFLP considers a set of facilities F ($|F| = m$) partitioned into k levels $F_1, \ldots, F_k$ and a set of clients D ($|D| = n$). Transportation costs c_{ij} for each $(i, j) \in \bigcup_{l=1}^{k-1}(F_{l+1} \times F_l) \cup (D \times F_k)$ are given and fixed costs f_i for establishing a facility $i \in F$ are assumed. A feasible solution is evaluated as a sum of the fixed costs of the located facilities, plus the clients' transportation costs. A client's transportation costs are calculated as the sum of the transportation cost from the client itself to the first assigned facility i, $i \in F_k$, and the transportation costs between successive facilities in the sequence of facilities assigned to the client. The objective of the MLUFLP is to minimize the sum of the total transportation costs and the fixed costs for establishing the facilities.

The MLUFLP is NP-hard, since it represents a generalization of the simple plant location problem that is proven to be NP-hard (Krarup and Pruzan, 1983). Improved inapproximability results and hardness factor for the MLUFLP were recently presented in Krishnaswamy and Sviridenko (2012).

Most of the papers that deal with different variants of MLUFLP consider theoretical analysis, such as the works of Aardal *et al.* (1999), Bumb and Kern (2001), Ageev (2002), Ageev *et al.* (2005) and Zhang (2006). These papers contain mainly theoretical aspects of the problem and provide no computational results. In the paper of Edwards (2001), the authors construct a shortest path-based algorithm (SP) for solving the MLUFLP and implement it. They also benchmark several previously proposed approaches for the MLUFLP: a linear program solution rounding 3-approximation algorithm MLRR in Aardal *et al.* (1999), a path reduction of the k-level facility location problem to a single-level problem (PR-RR) in Chudak and Shmoys (1999) and a local improvement 3-approximation algorithm for the path reduction PR-LI, in Charikar and Guha (1999). In the paper of Espejo *et al.* (2003), the authors treat the maximal covering two-level location problem. In Galvão *et al.* (2002) a 3-level facility location model is considered, with an upper bound on the maximum number of facilities to locate at each level; two heuristic methods are proposed for solving the problem.

Capacitated variants of the two-level facility location problem are extensively studied in the literature (Bloemhof *et al.*, 1996; Tragantalerngsak, 1997; Pirkul and Jayaraman, 1998; Charikar and Guha, 1999; Klose, 1999; Tragantalerngsak *et al.*, 2000; Klose, 2000). In the paper (Eitan *et al.*, 1991), the authors propose a mixed integer linear programming model of the capacitated variant of the MLUFLP with different hierarchical relationships between the nodes and assume both fixed and variable costs in the model.

Table 1
Fixed costs for establishing facilities.

f1	f2	f3	f4	f5	f6	f7	f8
3	2	2	4	4	1	1	1

Table 2
Transportation costs from facilities on the first level to facilities on the second level.

	f1	f2	f3
f4	9	6	8
f5	8	6	5
f6	9	5	9
f7	7	7	7
f8	8	9	5

The two-level location problem with modular node capacities has been recently studied in Addis *et al.* (2012), where a new formulation and an exact branch-and price algorithm are proposed for solving this problem. Several dynamic capacitated and uncapacitated variants are considered in Hinojosa *et al.* (2000), Melachrinoudis and Min (2000), Canel *et al.* (2001), Dias *et al.* (2008) and Lunday and Sherali (2010).

In the paper of Marić (2010), an evolutionary-based approach for solving the MLUFLP is presented. A binary encoding scheme is used with a corresponding objective function that implements a dynamic programming approach for finding the sequence of located facilities on each level to satisfy clients' demands. The proposed genetic algorithm (GA) reached all known optimal solutions for smaller size test instances and provided solutions for large-scale problem dimensions with up to $n = 2000$ clients and $m = 2000$ facilities. Moreover, all optimal/best known solutions are obtained by the GA for a single-level variant of the problem (simple plant location problem).

The MLUFLP is also studied in Gabor *et al.* (2010). The authors propose a new integer programming formulation for the multilevel uncapacitated facility location problem and a novel 3-approximation algorithm based on LP-rounding. In the case of a one-level problem ($k = 1$), this algorithm reduces to the 3-approximation algorithm described in Chudak and Shmoys (2003). For multiple levels, the algorithm must provide a path of open facilities for each demand node. It exploits the level structure preserved by the integer program: if one knows which facilities should be opened at the lowest r levels ($r \geqslant 1$) in order to ensure optimality, the problem is reduced to a facility level problem on $k - r$ levels. On each level, the facilities are opened according to a procedure similar to the one used in Chudak and Shmoys (2003) for the one-level problem. However, the authors provide no computational results in order to compare the effectiveness of the proposed formulation and the performance of the proposed 3-approximation algorithm.

Let us consider an example of the MLUFLP network with $n = 10$ clients, $m = 8$ potential facilities located on $k = 2$ levels. The first level F_1 contains 3 potential facilities, while the second level F_2 includes 5 potential facilities. The fixed costs for establishing facilities $\mathrm{f}1, \ldots, \mathrm{f}8$ are given in Table 1, while Table 2 shows transportation costs from facilities f1–f3 on the first level to facilities f4–f8 on the second level. The transportation costs from facilities on the second level to clients c1–c10 are presented in Table 3.

Table 3
Transportation costs from facilities on the second level to clients.

	f4	f5	f6	f7	f8
c1	5	6	6	8	7
c2	5	9	9	7	9
c3	5	6	8	9	7
c4	7	6	5	6	8
c5	6	5	6	7	7
c6	7	8	5	8	8
c7	9	6	6	7	7
c8	6	7	7	8	5
c9	9	5	8	8	6
c10	8	8	9	9	8

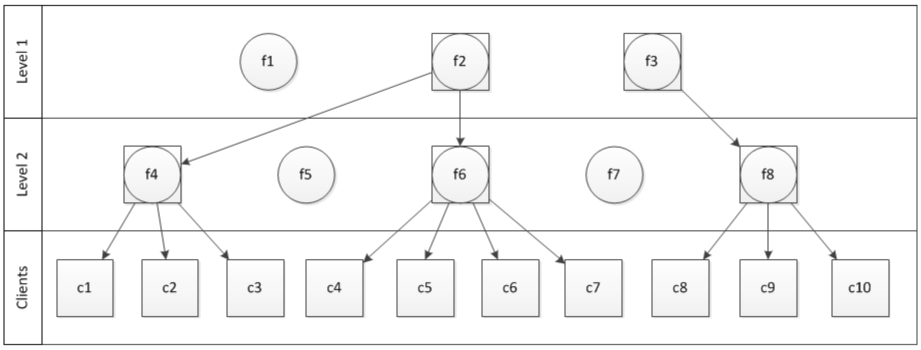

Fig. 1. Optimal solution for a network with 2 levels, 8 facilities and 10 clients.

The optimal solution is presented in Fig. 1, which shows that the established facilities are f2 and f3 on Level 1 and f4, f6 and f8 on Level 2. The sequences of facilities (r, s), $r \in F_2$, $s \in F_1$, assigned to each client can be seen from Fig. 1. The objective function value of the optimal solution is 119.

2. Mathematical Formulations of the MLUFLP

2.1. *Previous Mathematical Formulations*

In this section we first present the standard integer programming formulation of the MLUFLP from Edwards (2001). This formulation, denoted as the MLUFLP-1, considers the assignment of a client $j \in D$ to a valid sequence $p \in P$ of facilities, where the set of all valid sequences of facilities is defined by $P = F_k \times \cdots \times F_1$. The transportation cost of the assignment of a client $j \in D$ to a sequence $p = (i_k, \ldots, i_1)$, $p \in P$ is equal to $c_{pj} = c_{ji_k} + c_{i_k i_{k-1}} + \cdots + c_{i_2 i_1}$.

The MLUFLP-1 involves the following binary decision variables:

$$y_i = \begin{cases} 1, & \text{if } i \in F \text{ is open,} \\ 0, & \text{otherwise,} \end{cases} \qquad i \in F,$$

$$x_{pj} = \begin{cases} 1, & \text{if client } j \text{ is assigned to the sequence } p, \\ 0, & \text{otherwise,} \end{cases} \qquad p \in P,\ j \in D.$$

Using the notation mentioned above, the problem can be written as (formulation MLUFLP-1):

$$\min \sum_{i \in F} f_i y_i + \sum_{p \in P} \sum_{j \in D} c_{pj} x_{pj}, \tag{1}$$

$$\sum_{p \in P} x_{pj} = 1, \quad \text{for each } j \in D, \tag{2}$$

$$\sum_{\forall p: i \in p} x_{pj} \leqslant y_i, \quad \text{for each } i \in F,\ j \in D, \tag{3}$$

$$x_{pj} \in \{0, 1\}, \quad \text{for each } p \in P,\ j \in D, \tag{4}$$

$$y_i \in \{0, 1\}, \quad \text{for each } i \in F. \tag{5}$$

The objective function (1) minimizes the sum of overall transportation costs and the fixed costs for establishing facilities. Constraints (2) ensure that every client is assigned to a sequence of facilities, while constraints (3) guarantee that any facility in a sequence used by some client is paid for. Constraints (4) and (5) reflect the binary nature of variables x_{pj} and y_i.

Another mathematical formulation of the MLUFLP was proposed in Gabor *et al.* (2010). We will refer to it as the MLUFLP-2. In Gabor *et al.* (2010), authors use a notation slightly different from the MLUFLP-1. In the original formulation of MLUFLP-2, which we will also use, the facility levels are reversed, so that a client $d \in D$ is assigned to a facility $i \in F_1$ and then successively to facilities from $F_2, F_3, \ldots, F_k$. Accordingly, the transportation costs c_{ij} are given for each $(i, j) \in \bigcup_{l=1}^{k-1} (F_l \times F_{l+1}) \cup (F_1 \times D)$. Another minor difference is that the MLUFLP-2 involves the demands for each customer $k \in D$, denoted as w_k. Therefore, in the objective function calculation, the transportation costs per unit of flow c_{ij} are multiplied with these demands. Since our research is focussed on the variant of the MLUFLP with no clients' demands, by putting $w_k = 1$ for all $k \in D$ in the MLUFLP-2, this formulation becomes equivalent to the MLUFLP-1, which was confirmed by computational experiments in Section 5.1. Note that in our testings of the MLUFLP-2 formulation, we needed to re-numerate the levels to ensure the consistency, i.e. F_1 becomes F_k, F_2 becomes $F_{k-1}, \ldots, F_k$ becomes F_1.

The binary decision variables of the MLUFLP-2 are:

$$y_i = \begin{cases} 1, & \text{if facility } i \text{ is open,} \\ 0, & \text{otherwise,} \end{cases} \qquad i \in F,$$

$$x_{di}=\begin{cases}1, & \text{if demand point } d \text{ is assigned to facility } i,\\ 0, & \text{otherwise,}\end{cases} \qquad i\in F_1,\ d\in D,$$

$$z_{dij}=\begin{cases}1, & \text{if demand point } d \text{ uses edge } (i,j),\\ 0, & \text{otherwise,}\end{cases}$$

$$(i,j)\in F_l\times F_{l+1},\ l=1,\dots,k-1,\ d\in D.$$

The mathematical formulation MLUFLP-2 is as follows:

$$\min\sum_{i\in F} f_i y_i+\sum_{j\in D}\sum_{i\in F_1} w_j c_{ij} x_{ji}+\sum_{d\in D}\sum_{l=1}^{k-1}\sum_{i\in F_l}\sum_{j\in F_{l+1}} w_d c_{ij} z_{dij}, \tag{6}$$

$$\sum_{i\in F_1} x_{di}=1, \quad \text{for each } d\in D, \tag{7}$$

$$\sum_{j\in F_2} z_{dij}\leqslant x_{di}, \quad \text{for each } i\in F_1,\ d\in D, \tag{8}$$

$$\sum_{j\in F_{l+1}} z_{dij}\leqslant \sum_{j'\in F_{l-1}} z_{dj'i}, \quad \text{for each } i\in F_l,\ d\in D,\ l=2,\dots,k-1, \tag{9}$$

$$x_{di}\leqslant y_i, \quad \text{for each } i\in F_1,\ d\in D, \tag{10}$$

$$\sum_{j\in F_{l-1}} z_{dji}\leqslant y_i, \quad \text{for each } i\in F_l,\ d\in D,\ l=2,\dots,k, \tag{11}$$

$$x_{di}\in\{0,1\}, \quad \text{for each } d\in D,\ i\in F_1, \tag{12}$$

$$z_{dij}\in\{0,1\}, \quad \text{for each } (i,j)\in F_l\times F_{l+1},\ l=2,\dots,k,\ d\in D, \tag{13}$$

$$y_i\in\{0,1\}, \quad \text{for each } i\in F. \tag{14}$$

Constraints (7) ensure that each demand point $d\in D$ is connected to exactly one facility on the first level. Constraints (8) say that a demand point d uses an edge $(i,j)\in F_1\times F_2$ only if d is assigned to facility $i\in F_1$. Constraints (9) ensure that a demand point d uses an edge $(i,j)\in F_l\times F_{l+1}$, $l=2,\dots,k-1$ only if d uses an edge (j',i), for some $j'\in F_{l-1}$, but for the same i. Finally, constraints (10) and (11) respectively indicate that a demand point d will be assigned to a facility $i\in F_1$ and will use an edge $(j,i)\in F_{l-1}\times F_l$, $l=2,\dots,k$, only if facility i is open. All variables used in this model are binary by constraints (12)–(14).

Note that the number of variables in the MLUFLP-2 has decreased from an exponential one in the MLUFLP-1

$$|D|\cdot|F_1|\cdot|F_2|\cdots|F_k|+|F|$$

to a polynomial one:

$$|F|+|D|\cdot|F_1|+|D|\cdot\sum_{l=1}^{k-1}|F_l|\cdot|F_{l+1}|.$$

The number of constraints in the MLUFLP-2 is polynomial. However, it is greater than the number of constraints in the MLUFLP-1:

$$|D| + 2|D| \cdot \sum_{l=1}^{k-1} |F_l| + |D| \cdot |F_k|$$

constraints versus $|D| + |F| \cdot |D|$ in the MLUFLP-1 from Edwards (2001).

2.2. *New Mathematical Formulation*

A new mathematical formulation, named MLUFLP-3 uses the same notation as the MLUFLP-1. Let's observe the set of clients D as a new level, a level of the $k+1$-th order, i.e. let's introduce the identity $D \equiv F_{k+1}$. Instead of the binary variables x_{pj}, the new formulation uses integer variables $z_{is}^l \geqslant 0$ representing the number of clients from D that are supplied via link (i, s), where $i \in F_l$ and $s \in F_{l-1}$ belong to two adjacent levels l and $l-1$ and $l = 2, \dots, k+1$. Since we defined $D \equiv F_{k+1}$, we allow a client's node to be the left side node of the considered link, i.e. $i \in F_{l+1}$. The binary variables $y_i \in \{0, 1\}$, $i \in F$ remain, indicating whether a facility is established at a location i or not. The transportation costs c_{ij} are given as in MLUFLP-1.

The new integer programming formulation of the MLUFLP-3 is as follows:

$$\min \sum_{i=1}^{m} f_i y_i + \sum_{l=2}^{k+1} \sum_{i \in F_l} \sum_{s \in F_{l-1}} c_{is} z_{is}^l, \tag{15}$$

$$\sum_{i \in F_k} z_{ji}^k = 1, \quad \text{for each } j \in D \equiv F_{k+1}, \tag{16}$$

$$\sum_{s \in F_{l-1}} z_{is}^l = \sum_{r \in F_{l+1}} z_{ri}^{l+1}, \quad \text{for each } i \in F_l,\ l = 2, \dots, k, \tag{17}$$

$$z_{ri}^{l+1} \leqslant n y_i, \quad \text{for each } i \in F_l,\ r \in F_{l+1},\ l = 1, \dots, k, \tag{18}$$

$$z_{is}^l \in \mathbb{N} \cup \{0\}, \quad \text{for each } l = 2, \dots, k+1,\ i \in F_l,\ s \in F_{l-1}, \tag{19}$$

$$y_i \in \{0, 1\}, \quad \text{for each } i \in F. \tag{20}$$

The objective function (15) minimizes the sum of overall transportation costs and fixed costs for establishing facilities. Constraints (16) ensure that every client (at the level $F_{k+1} = D$) is supplied from exactly one facility at the level k. Constraints (17) guarantee that for each facility i on each level l, the number of "incoming" client assignments in the node i is equal to the number of "outgoing" assignments. By constraints (18) we make sure that the clients are supplied via the established facilities only and that the number of clients that use facilities on two subsequent levels does not exceed the total number of clients. Constraints (19) imply that the variables z_{is}^l take non-negative integer values, while constraints (20) reflect the binary nature of the variables y_i.

The number of variables of the MLUFLP-3 is also polynomial, as in the MLUFLP-2:

$$\begin{aligned}|F| + \sum_{l=2}^{k+1} |F_l| \cdot |F_{l-1}| &= |F| + |F_{k+1}| \cdot |F_k| + \sum_{l=2}^{k} |F_l| \cdot |F_{l-1}| \\ &= |F| + |F_k| \cdot |D| + \sum_{l=1}^{k-1} |F_{l+1}| \cdot |F_l|.\end{aligned}$$

By comparing the expressions for the number of variables in the MLUFLP-2 and MLUFLP-3, we can notice that the sum that occurs in the expression for the MLUFLP-2 is additionally multiplied by $|D|$. Therefore, we conclude that the number number smaller compared to the number of variables in the MLUFLP-2.

The number of constraints in the MLUFLP-3 is:

$$\begin{aligned}&|D| + \sum_{l=2}^{k} |F_l| + \sum_{l=1}^{k} |F_l| \cdot |F_{l+1}| \\ &= |D| + \sum_{l=2}^{k} |F_l| + \sum_{l=1}^{k-1} |F_l| \cdot |F_{l+1}| + |F_{k+1}| \cdot |F_k| \\ &= |D| - |F_1| + |F_k| + \sum_{l=1}^{k-1} |F_l| + \sum_{l=1}^{k-1} |F_l| \cdot |F_{l+1}| + |D| \cdot |F_k| \\ &= |D| + |D| \cdot |F_k| - |F_1| + |F_k| + \sum_{l=1}^{k-1} (1 + |F_l|) \cdot |F_{l+1}|.\end{aligned}$$

This number of constraints in the MLUFLP-3 is smaller compared to the the MLUFLP-2, as long as $2|D| > 1 + |F_l|$ and the number of facilities on the level F_k is not significantly greater than on F_1.

The conducted computational experiments confirmed that formulations MLUFLP-1 and MLUFLP-3 are equivalent (see Section 5.1) and the formal mathematical proof of the equivalence is given in Appendix.

3. A Proposed Memetic Algorithm for Solving the MLUFLP

Evolutionary algorithms have proved to be robust and effective heuristics for solving various optimization problems. However, there are many situations in which a pure evolutionary algorithm does not perform particulary well and various hybridizations of the EA with other methods have been proposed (Misevičus, 2006; Fan *et al.*, 2006; Chen *et al.*, 2008; Misevičus and Rubliauskas, 2009; Prestwich *et al.*, 2009). Hybridizations of evolutionary algorithms with local search methods are denoted as memetic algorithms in the literature (Moscato and Cotta, 2003, 2007; Neri *et al.*, 2012).

The role of the evolutionary part in a memetic algorithm (MA) is to focus the search on promising regions of the search space. Once the promising areas with high quality solutions have been identified, local search methods are applied in order to determine the best solutions in these areas. This type of hybridization showed to be very successful for many optimization problems in the literature. The main idea of a basic memetic algorithm (Moscato and Cotta, 2007) is to apply a local search method on the newly-generated individuals that are candidates for entering a new generation. Individuals that are improved by the means of the applied local search will enter a new generation if they satisfy certain criteria. However, it turns that for our problem, a basic MA approach is not so efficient in finding good-quality solutions, especially for large-scale problem dimensions.

In this paper, we propose a modification in the strategy of combining the evolutionary approach with a local search method. The idea is to apply the local search occasionally, only in situations when the evolutionary method gives no improvement of the objective function value through a significant number of iterations or when it converges extremely slowly. Another difference from a basic MA method is that we do not apply the local search on the whole population – we search for improvements in the neighborhoods of a certain number of individuals only. In this way, we give a chance to individuals with different quality of their genetic material to be improved. Even small improvements of individuals with worse objective function values may direct the evolutionary method to the global optimum through further applications of evolutionary operators and local search procedures in the successive MA iterations.

The purpose of the described strategy is to give an impulse to the evolutionary algorithm in cases when it would most probably converge to a local optimum trap. The local search is applied only to some, not all, individuals in the population, which has a twofold effect: it does not significantly decrease the diversity of the genetic material in the population and it does not excessively increase the running time. The applied strategy showed to be adequate for solving the MLUFLP efficiently, especially large-scale problem instances. Preliminary computational experiments show that the improvements of the MLUFLP solutions by involving this strategy become more obvious as the problem dimension increases. In addition, we applied several parallelization techniques in order to achieve speed ups, which are most conspicuous when solving larger MLUFLP instances.

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

An initial population, containing 150 individuals (chromosomes), is sequentially generated pseudo-randomly, thus providing a good initial diversity of the genetic material. From the initial random seed for the current MA run, we generate 150 different random seeds and assign one seed to each chromosome in the population. This step is performed in order to ensure the same behavior of individuals, no matter if the algorithm is run in parallel or sequentially.

Differently from a basic MA approach, we do not apply a local search in order to improve our randomly generated initial population. For our problem, the use of a local search method in this stage of the MA showed to be time-expensive, especially for larger problem dimensions. Our computational experiments show that the absence of a local search method for improving the initial population doesn't affect solutions' quality.

Algorithm 1: Memetic algorithm.

```
randomly generate initial population;
while stopping condition not reached do
    if in previous N generations there were no improvements then
        select chromosomes for local search;
        apply parallel local search on selected chromosomes;
    end
    apply evolutionary operators:
        do parallel fitness calculation;
        do selection;
        do parallel two point crossover;
        do parallel mutation;
    parallel calculation of the objective function value for the current generation;
end
```

In each generation of the MA, the worst $1/3$ of the population is replaced, while the remaining $2/3$ of the population is directly passed to the next generation. The chromosome and the objective function value of the best individual are kept and updated every time an improvement is achieved.

In each iteration, the algorithm checks whether or not there was some improvement of the best individual in the previous N consecutive MA generations. If not, we apply the local search procedure, but only on selected individuals from the MA population. Evolutionary operators are further applied in each MA iteration: parallel fitness calculation, selection, parallel crossover and parallel mutation. Finally, the objective function of newly created individuals is calculated in parallel.

The proposed MA uses a combination of three stopping criteria, i.e. the algorithm stops if one of the following conditions is satisfied:

- maximal number of 5000 generations;
- no improvement of the best individual is achieved through 2000 consecutive MA generations;
- the MA reaches the time limit of 1 hour.

The encoding of the solutions and evolutionary operators used in the proposed memetic algorithm are customized for the problem under consideration. Parallel programming techniques are implemented in order to improve the efficiency of the algorithm. Several strategies are applied to increase the diversity of individuals and keep the algorithm away from the local optimal trap. In the following sections, all aspects of the proposed memetic algorithm will be explained in details.

3.1. *Encoding and Objective Function Calculation*

The proposed algorithm uses a binary encoding, i.e. each solution is represented by a binary string (chromosome) of the length $m = |F|$. Each bit in the chromosome corresponds to one potential facility in the network. If the bit on the k-th position in the chromosome takes the value of 1, it means that a facility is located at the k-th node. Zero on the k-th position in the chromosome indicates that the k-th node is not chosen for establishing a facility.

For example, chromosome 01110101 corresponds to the optimal solution presented in Example 1. Facilities are established at nodes 2, 3, 4, 6 and 8 which gives us the values of variables y_i: $y_2 = y_3 = y_4 = y_6 = y_8 = 1$ and $y_1 = y_5 = y_7 = 0$.

The objective function is calculated in the following way. From the chromosome we obtain the locations of the established facilities, and therefore, the values of the variables y_i, $i = 1, \dots, m$. A chromosome is labeled "correct", if it corresponds to a feasible solution of the MLUFLP, i.e. if there exists at least one established facility at each level l, $l = 1, \dots, k$. "Incorrect" chromosomes are corrected by randomly locating one facility at each level with no previously established facilities. Since crossover and mutation operators may produce "incorrect" chromosome, the procedure of correcting newly generated infeasible individuals is done in each MA generation. For good performance of the MA it is important to preserve the property of "correctness" of chromosomes (i.e. feasibility of solutions) in the MA population.

The main idea in calculating the objective function value for a given chromosome is to decrease the number of potential paths. Considering all possible paths is time and memory consuming, and in this case problems of practical size would remain unsolved. The objective function is calculated by solving the subproblem of the MLUFLP, named the FixedMLUFLP, obtained from the MLUFLP by fixing the established facilities, with an additional condition that each level contains at least one located facility. Since each (corrected) chromosome in every MA generation has this property, it may be used as a starting point for the FixedMLUFLP. The FixedMLUFLP has polynomial complexity and may be solved by using the dynamic programming approach, proposed in Marić (2010). In the same paper, it was proven that the FixedMLUFLP has optimal substructure of solutions, which was stated by the following theorem:

Theorem 1. *The FixedMLUFLP can be polynomially reduced to the shortest path problem in a directed acyclic graph* (*DAG*).

For the proof of Theorem 1, we refer to paper of Marić (2010).

After solving the FixedMLUFLP to optimality, the solutions obtained from the dynamic programming give us the corresponding allocations and the objective function value of the MLUFLP for the considered feasible solution.

3.2. *Evolutionary Part of the MA*

For evaluating the quality of individuals in the population, we apply parallel calculation of the fitness function. A fitness value is assigned to every individual in the population and it represents its chances to take part in producing the next generation. Therefore, it is important that a fitness value reflects the quality of the individual in some "real" way.

Fitted individuals in the population succeed in creating offspring, while the unfit ones are removed from the population.

In the literature there are different ways to define a fitness of a individual, such as direct mapping from the objective function, fitness ranking, fitness remapping, etc. Fitness ranking and fitness remapping are often used in evolutionary algorithms, since they are performed within a chosen selection operator in an efficient manner. In practice, fitness ranking is realized within a rang-based selection operator, while fitness remapping is implemented within a tournament-based selection.

In our algorithm, we use the strategy of fitness remapping within a tournament-based selection operator. This strategy is not time consuming and it works well for both minimization and maximization problems. The parallel fitness remapping used in our EA is applied as follows. Chromosomes are searched in parallel in order to obtain the maximum and the minimum objective function values: $\max_{obj}$ and $\min_{obj}$ respectively. In the second parallel pass through the array of chromosomes, we calculate the fitness of each chromosome $\mathit{fitness}(chr)$ in the following way:

$$\mathit{fitness}(chr) = \frac{\max_{obj} - obj(chr)}{\max_{obj} - \min_{obj}},$$

where $obj(chr)$ is the objective function value of the current chromosome.

The applied strategy of fitness calculation follows the nature of evolutionary based algorithms and gives good results in practice. By implemented fitness remapping, we increase the effects of tournament-based selection as the algorithm progresses, which leads the algorithm to high-quality solutions.

In order to ensure the diversity of individuals and prevent a premature convergence, duplicate individuals are removed by setting their fitness to 0. Further, the number of individuals with the same objective (fitness) value, but different chromosomes, is limited to some constant parameter. This is also done by setting their fitness value to 0.

As a selection method, we used the fine grained tournament selection, described in Filipović (2003). Instead of having an integer tournament size, as in the classic tournament selection, this selection operator is performed over 50 tournaments of different sizes that are executed sequentially. Each of 150 individuals from the population may be randomly chosen to participate in one of the tournaments. Individuals that are tournament winners are further subjected to the parallel crossover and parallel mutation operators. In our implementation, in 60% of the cases the tournament size is 5, and 6 otherwise.

In the initialization part of the MA, an array of two-point crossover operators is created and random seeds for those operators are generated and assigned, one to each operator. In each generation of the MA, pairs of parent-chromosomes that will exchange their genetic material are chosen using the selection method described above. A different crossover operator is applied to each pair of parent-chromosomes, producing two offspring-chromosomes. The parents exchange genes after the crossover points, which are chosen using the random seed of the operator. For these reasons the crossover step will produce the same results, regardless of if it is performed in parallel or sequentially.

In our MA implementation, the probability that a chosen pair of parent-chromosomes will exchange their genes and create new chromosomes is set to 0.8. Otherwise, if no gene exchange occurs, the offspring-chromosomes remain identical to their parents.

The main purpose of mutation is to counteract premature convergence and to maintain enough diversity in the population. The diversity of genetic material is usually large at the beginning of a run and decreases with time. The appearance of frozen bits (Stanimirović *et al.*, 2007) in later MA stages rapidly increases the possibility of premature convergence. Therefore, we have used the concept of a mutation operator with frozen bits (Stanimirović *et al.*, 2007), which has been parallelized in our MA approach.

In the initialization part of the MA, we create an array of mutation operators. For each individual in the population, we apply one mutation operator from the created array. Each mutation operator works by changing a randomly selected bit in the chromosome (0 to 1, 1 to 0) with certain probability (mutation rate). Mutation rate depends only on length of the chromosome m, and whether a bit to be mutated is frozen or not. The probability of changing a non-frozen bit (basic mutation rate) is set to $\frac{0.5}{m}$, which means that one bit (out of m) will be changed with the probability of 0.5. Frozen bits are mutated with 4 times higher probability, which helps in maintaining the diversity of genetic material in the population.

The Parallel Frozen Bits Detector is applied in order to obtain the positions of frozen bits in chromosomes. This procedure constructs two masks of the same length as an individual's genetic code. One mask stores the positions of frozen zeros, and the other the positions of frozen ones in the genetic code.

The conducted computational experiments showed that the parallelization techniques, implemented in the proposed MA, ensure significant improvements in the sense of CPU time. A comparison of the sequential and the enhanced, parallelized version of the MA on a chosen subset of the challenging MLUFLP instances is presented in Section 5.3.

4. Local Search Improvement Procedure of the MA

An important question when designing a memetic algorithms is how to incorporate a local search method so that a good balance between the global and a local search is achieved. If the effect of local search is too strong, the algorithm may quickly converge to local optima and the algorithm is likely to rediscover the same local optimum over and over again. In addition, an excessive local search quickly leads to a loss of diversity within the population (Neri *et al.*, 2012).

The importance of this problem has been recognized by Hart in Hart (1994), who posed four basic questions regarding the usage of a local search within MA frame:

- How often should a local search be applied? (local search frequency).
- On which solutions should a local search be used? (local search probability).
- How long should a local search be run? (local search depth).
- How efficient does a local search need to be? (local search efficiency).

These four questions are directed to determining four local search parameters: local search frequency, local search probability, local search depth and local search efficiency, respectively. In concrete implementations of memetic algorithms, one can find different combinations and different values of these parameters related to the use of local search, which define various mechanisms for balancing global and local search. The list of mechanisms used in the literature is by far not complete, but the combination of local search frequency and the local search depth are considered as the most typical ones (Neri *et al.*, 2012). In basic MA implementations, a local search is usually applied with a fixed frequency on the whole population of individuals. Another strategy is to apply a local search probabilistically with certain value of local search probability parameter, which is usually fixed in basic MA implementations. Regarding the third question posed by Hart, the running time of a local search is often considered as the local search depth parameter. Other balancing mechanisms define the local search depth parameter as the size of the neighborhood of a solution that is subject to a local search. The local search depth parameter may vary through MA iterations, but it is more often set to some fixed value. The quality of the obtained improvement is usually used as a local search efficiency parameter, which is related to the fourth question.

Differently from a basic MA concept (see Moscato and Cotta, 2003, 2007; Neri *et al.*, 2012), we use a variable local search frequency in designing our MA approach. More precisely, we apply local search only in cases when there are strong indications that the evolutionary algorithm would converge to a local optimum. Further, local search is applied only to a portion of the search space, giving a chance to individuals with different fitness values to be improved. The local search depth is restricted to an individual's neighborhood of size 1 in one local search iteration, i.e. we try to invert one bit in the chromosome at a time, trying to obtain an individual with a better fitness value. Even small occasional improvements provide a good impulse for the evolutionary algorithm to perform better and escape from a local optimum trap. The results of preliminary experiments showed the efficiency of the MA when only 10% of the population is subjected to the local search procedure in cases when no improvement is obtained after 150 consecutive generations. The conducted computational results confirm that such combination of a variable local search frequency, fixed local search probability and fixed local search depth represent a good mechanism for balancing the evolutionary (global) and the local search when solving the MLUFLP.

Algorithm 2 describes the functioning scheme of the Local search procedure within the memetic algorithm.

Local search procedure goes through a chromosome and inverts a single bit value (0 to 1, 1 to 0), starting at the bit position 1, which corresponds to the binary variable y_1, i.e., opening/closing the facility 1. If the new chromosome is valid and its objective function is improved, the initial chromosome is replaced with the new one. The described procedure is repeated until no further improvement is achieved. There is no time limit imposed on the local search of an individual.

We have further implemented parallelization in the local search procedure. If the MA obtained no improvement within N consecutive generations, we apply the parallel lo-

Algorithm 2: Local search.

```
foreach R-th chromosome chr in population do
    obj ← objective value of chr;
    improvement ← true;
    while improvement do
        improvement ← false;
        k ← 1;
        while k ⩽ chromosme length and not improvement do
            if inverting the k-th bit on chr gives a valid chromosome then
                new_chr ← chr with inverted k-th bit;
                new_obj ← objective value of new_chr;
                if new_obj < obj then
                    obj ← new_obj;
                    chr ← new_chr;
                    improvement ← true;
                end
            end
            k ← k + 1;
        end
    end
end
```

cal search procedure on every R-th individual in the MA population looking for an improvement in the neighborhood of one of the selected individuals. The local search procedure is completely deterministic and affects only the chromosome on which it is performed.

5. Computational Results

In this section, computational results for the proposed MA and comparisons with existing methods for solving the MLUFLP are presented. All three integer programming formulations are implemented and tested by using CPLEX 12.1 solver in order to obtain optimal solutions for considered MLUFLP instances and compare their effectiveness.

Computational experiments were carried out on an Intel Core i7-860 2.8 GHz (quad-core processor) with 8 GB RAM memory under Windows 7 Professional operating system. The MA method was implemented by using the .NET Framework.

In implementing parallelization elements in the MA, we followed the idea of threads that are being executed on different processors. We adopted the concept of threads with shared memory architecture and used it in our experiments. For creating threads on different processors, we used the .NET Task Parallel Library.

Table 4
Data sets used in our computational experiments.

Data set	Number of levels	Dimension	Description
ORLIB instances	$k = 1$	$50 \leqslant n \leqslant 1000$ $16 \leqslant m \leqslant 100$	Standard ORLIB data set (Beasley, 1996); Initially designed for the UFLP; Contains small and medium size test problems.
M* instances	$k = 1$	$300 \leqslant n \leqslant 2000$ $300 \leqslant m \leqslant 2000$	Large scale instances; Introduced in Raidl and Gottlieb (2005).
Modified UFLP instances	$2 \leqslant k \leqslant 4$	$50 \leqslant n \leqslant 1000$ $16 \leqslant m \leqslant 100$	Based on standard UFLP instances; Generated from ORLIB instances (Beasley, 1996) for the UFLP; Introduced in Marić (2010).
Large-scale MLUFLP instances	$2 \leqslant k \leqslant 5$	$300 \leqslant n \leqslant 2000$ $300 \leqslant m \leqslant 2000$	Challenging large-scale data set for the MLUFLP; Derived from the M* data set from (Raidl and Gottlieb, 2005); Introduced in Marić (2010).

Computational experiments in this study were carried out on a large number of the MLUFLP instances from the literature. A brief description of data sets used in our tests is given in Table 4.

For more detailed explanation on the benchmark data set for the MLUFLP we refer to Marić (2010). Note that the instances from Edwards (2001) are not available and they are, in most cases, based on standard ORLIB instances.

5.1. *Experimental Comparisons of the MLUFLP Formulations*

In Table 5 we present computational results of the three MLUFLP formulations on instances from data sets described above. In the first column, an instance's name is given, which includes the information on the original instance, the number of levels, the facilities on each level and the clients respectively. For example, the instance $capb\_3L\_12\_25\_63.1000$ is created by modifying the ORLIB instance *capb* and involves 3 levels with 12, 25 and 63 facilities respectively, and 1000 clients.

The following two columns contain the results of the MLUFLP-3 model: solutions on the tested data set obtained by the CPLEX 12.1 solver and the corresponding total CPU times. The remaining columns show the results of the MLUFLP-2 and MLUFLP-1 models, presented in the same way as for the MLUFLP-3. In cases when a tested model gave no solution, a dash "–" is written.

From the results given in Table 5, it is obvious that the MLUFLP-3 outperforms both MLUFLP-1 and MLUFLP-2 formulations. It gives optimal solution for 27 tested instances, providing optimal solution for 13 instances that were out of reach of previous two MLUFLP formulations. The average running time for the MLUFLP-3 is 1762.750 seconds, which is less than half an hour. For the remaining instances from the MLUFLP data sets mentioned above, no optimal solution is found due to time or memory limits. Comparing the average running times of the three formulations on the first 14 instances in Table 5, we notice that the MLUFLP-3 is slightly faster than the MLUFLP-1, while the MLUFLP-2 is around three times slower compared to both MLUFLP-3 and MLUFLP-1.

Table 5
Comparison of the MLUFLP formulations

Instance name	Opt_{sol}	MLUFLP-3	MLUFLP-2	MLUFLP-1
		$t(s)$	$t(s)$	$t(s)$
cap71_1L_16.50	932615.750	0.025	0.020	0.010
cap71_2L_6_10.50	1813375.513	0.041	0.039	0.024
cap71_3L_2_5_9.50	4703216.306	0.030	0.054	0.041
cap101_1L_25.50	796648.438	0.028	0.029	0.031
cap101_2L_8_17.50	1581551.394	0.065	0.049	0.046
cap101_3L_3_7_15.50	3227179.813	0.048	0.100	0.123
cap131_1L_50.50	793439.563	0.027	0.105	0.041
cap131_2L_13_37.50	1592548.450	0.150	0.202	0.222
cap131_3L_6_14_30.50	3201970.463	0.131	0.363	1.262
cap131_4L_3_7_15_25.50	3630297.669	0.109	0.302	5.199
capa_1L_100.1000	17156454.478	1.866	5.787	4.043
capb_1L_100.1000	12979071.581	2.000	5.694	3.759
capc_1L_100.1000	11505594.329	29.796	43.477	29.956
mq1_1L_300.300	3591.273	275.684	876.944	284.198
Average		22.143	66.655	23.497
capa_2L_30_70.1000	31524957.410	2407.986	–	–
capa_3L_15_30_55.1000	40725103.254	41.427	–	–
capa_4L_6_12_24_58.1000	54643362.801	169.598	–	–
capb_2L_35_65.1000	25224163.283	1197.128	–	–
capb_3L_12_25_63.1000	34978486.506	34.164	–	–
capb_4L_6_13_31_50.1000	53034149.833	191.909	–	–
capc_2L_32_68.1000	22762468.838	548.607	–	–
capc_3L_13_27_60.1000	35540649.433	332.641	–	–
capc_4L_4_9_27_60.1000	57017358.038	322.868	–	–
mq1_2L_100_200.300	8341.287	9627.119	–	–
mq1_3L_30_80_190.300	12994.871	9943.663	–	–
mq1_4L_20_40_80_160.300	17648.010	5236.609	–	–
mq1_4L_18_39_81_162.300	18048.031	17230.518	–	–
Average		1762.750	–	–

5.2. *Parameter Sensitivity Analysis*

Before we ran computational tests for the MA, we had experimented with different values of parameters N and R, which denote the number of MA generations with no improvement and the portion of the MA population on which we apply local search heuristic respectively. We had varied N and R in order to determine the most preferable combination for further experiments. The computational experiments were first carried out on the instance $mr1\_3l\_55\_120\_325.500$ with 500 clients and 500 potential facilities located on 3 levels. We ran the parallel MA 20 times for each combination of parameters, but with different random seeds. The same combination of stopping criteria defined in Section 3 was used in these experiments.

In Table 6, for each combination of the parameters we present:

- best solution of the MA;
- running time (in seconds);

Table 6
MA results on instance $mr1\_3l\_55\_120\_325.500$ for different values of N and R.

N	R	*Best*	$t(s)$	gen	$agap(\%)$	$\sigma(\%)$
50	3	10911.319	49.610	991.2	0.460	0.579
50	5	10911.319	29.040	959.65	0.733	0.726
50	8	10911.319	23.020	818.75	1.485	1.221
50	10	10911.319	17.890	873.3	1.378	1.117
100	3	10911.319	7.500	1015.45	0.950	0.781
100	5	10911.319	9.440	1043.3	0.966	0.799
100	8	10911.319	7.380	954.3	1.627	1.210
100	10	10911.319	7.540	980.6	1.472	1.118
150	3	10911.319	7.420	1094	0.866	0.759
150	5	10911.319	7.300	1078.2	0.969	0.781
150	8	10911.319	7.360	990.85	1.176	1.015
150	10	10911.319	7.370	978	1.336	1.121

Table 7
MA results on instance $ms1\_5l\_25\_55\_120\_250\_550.1000$ for different values of N and R.

N	R	*Best*	$t(s)$	$agap(\%)$	$\sigma(\%)$
100	3	40070.037	311.770	0.636	0.536
100	5	40070.037	424.290	0.484	0.407
100	8	40070.037	290.560	0.844	0.470
100	10	40171.410	301.030	0.723	0.518
100	15	40070.037	432.260	0.869	0.542
100	20	40171.410	340.000	0.910	0.722
150	3	40070.037	283.200	0.611	0.517
150	5	40070.037	225.720	0.631	0.448
150	8	40171.410	170.990	0.813	0.562
150	10	40070.037	139.070	1.049	0.659
150	15	40070.037	134.000	0.543	0.514
150	20	40171.410	98.570	0.948	0.752

- number of MA generations;
- average gap from the best solution (in percents);
- standard deviation σ (in percents).

As it can be seen from Table 6, for each of the 12 tested combinations, the MA quickly reached best solutions, which indicates good stability of the algorithm. As it was expected, the combinations with $N = 50$ generations produce significantly longer MA runs on the considered medium size instance, since we apply local search more often. Therefore, we omitted these combinations from further experiments on larger data set.

In Table 7, we present the results of the MA with different parameter values on a large-scale instance $ms1\_5l\_25\_55\_120\_250\_550.1000$ with 1000 clients and 1000 potential facilities on 5 levels. We kept the values of $N = 100$ and $N = 150$ for the first parameter, and for these values, we varied the second parameter R. The results are presented in the same way as in Table 6.

The column "Best" in Table 7 shows that there is a difference in the quality of the obtained solutions for different parameter combinations. The MA reached best-known solution for combinations $N = 100$, $R = 3, 5, 8, 15$ and $N = 150$, $R = 3, 5, 10, 15$. Regarding the columns $avg(\%)$ and $\sigma(\%)$ related to these parameter combinations, we notice that the lowest values of the average gap and standard deviation were obtained for $N = 100$, $R = 5$ and $N = 150$, $R = 15$. Regarding the running times for these two combinations, we notice that the MA is more than three times faster using the parameters $N = 150$, $R = 15$. Therefore, based on short CPU times and good solution quality, we decided to use these parameter values $N = 150$, $R = 15$ in further experiments.

5.3. *The Impact of the Implemented Parallelization Techniques in the MA*

In order to investigate the effects of the parallelization techniques implemented in the proposed MA, we performed additional computational experiments. We benchmarked the parallel MA and the sequential MA (without any parallelization) on all available MLUFLP test instances, described above. In order to obtain fair comparisons, we ran both variants of the MA with the same random seed. The stopping criterion was maximum number of 2000 MA generations. All experiments were carried out on the same platform.

In Table 8, we first give the MLUFLP instance's name and the optimal solution (if it is known). We further present the best solution obtained by both variants of the MA and the number of MA generations used as the stopping criterion. Finally, we give the running times of the parallel and the sequential variants of the MA in which they achieve the best solution.

Regarding the way of implementing parallelization techniques in the MA (see Section 3), and the fact that we used the same random seed for both the parallel and the sequential MA, it is clear that for each considered test instance, the best solutions obtained by both variants of the MA will be the same. Therefore, we compare the running times needed for the parallel and the sequential MA to obtain the best solutions. Note that for both variants of the MA we used the same number of generations as the only stopping criterion.

As it can be seen from Table 8, the maximal running times are produced in the case of instance $mt1\_4L\_120\_250\_520\_1110.2000$: the sequential MA needed 8120.320 s to obtain the best solution, while the parallel MA needed only 2717.830 s. Regarding the average running times over the whole MLUFLP data set, we notice that the average running time for the sequential MA is 742.970 s and 252.392 s for the parallel MA. Therefore, we may conclude that the implemented parallelization techniques give significant contribution to MA's computational efficiency. The running times presented in Table 8 show that the parallel MA has almost 3 times better performance compared to sequential MA. Note that all experiments are carried out on the same processor with four cores.

5.4. *Computational Results of the MA on the MLUFLP Benchmark Set*

In this subsection, we present the results of the MA on the MLUFLP instances introduced in the literature so far. The MA results are compared with the results of the genetic algorithm approach (GA) from Marić (2010), which was run on Intel 1.8 GHz with 512 MB

Table 8
Comparisons of the parallelized and sequential MA.

Instance name	Opt_{sol}	MA		Parallel MA	Sequential MA
		best	*gen*	$t(s)$	$t(s)$
cap71_1L_16.50	932615.750	opt	2000	0.040	0.030
cap71_2L_6_10.50	1813375.513	opt	2000	0.040	0.030
cap71_3L_2_5_9.50	4703216.306	opt	2000	0.040	0.030
cap101_1L_25.50	796648.438	opt	2000	0.040	0.050
cap101_2L_8_17.50	1581551.394	opt	2000	0.040	0.040
cap101_3L_3_7_15.50	3227179.813	opt	2000	0.050	0.040
cap131_1L_50.50	793439.563	opt	2000	0.160	0.290
cap131_2L_13_37.50	1592548.450	opt	2000	0.070	0.110
cap131_3L_6_14_30.50	3201970.463	opt	2000	0.470	0.880
cap131_4L_3_7_15_25.50	3630297.669	opt	2000	0.070	0.110
mq1_1L_300.300	3591.273	opt	2000	7.820	23.800
mr1_1L_500.500	–	2349.856	2000	22.400	70.200
ms1_1L_1000.1000	–	4378.632	2000	240.170	737.820
mt1_1L_2000.2000	–	9176.509	2000	1197.490	3210.580
capa_1L_100.1000	17156454.478	opt	2000	6.200	17.840
capa_2L_30_70.1000	31524957.410	opt	2000	6.940	20.700
capa_3L_15_30_55.1000	40725103.254	opt	2000	3.730	10.520
capa_4L_6_12_24_58.1000	54643362.801	opt	2000	10.210	29.700
capb_1L_100.1000	12979071.581	opt	2000	14.650	42.930
capb_2L_35_65.1000	25224163.283	opt	2000	33.690	101.630
capb_3L_12_25_63.1000	34978486.506	opt	2000	1.050	2.920
capb_4L_6_13_31_50.1000	53034149.833	opt	2000	2.030	5.990
capc_1L_100.1000	11505594.329	opt	2000	33.380	100.040
capc_2L_32_68.1000	22762468.838	opt	2000	22.390	67.230
capc_3L_13_27_60.1000	35540649.433	opt	2000	5.260	14.710
capc_4L_4_9_27_60.1000	57017358.038	opt	2000	7.540	23.490
mq1_2L_100_200.300	8341.287	opt	2000	7.040	21.230
mq1_3L_30_80_190.300	12994.871	opt	2000	4.140	12.810
mq1_4L_20_40_80_160.300	17648.010	opt	2000	15.540	44.020
mq1_4L_18_39_81_162.300	18048.031	opt	2000	13.890	39.920
mr1_2L_160_340.500	–	6707.505	2000	76.400	203.680
mr1_3L_55_120_325.500	–	11113.620	2000	7.540	23.770
mr1_4L_30_65_140_265.500	–	15399.713	2000	82.330	237.970
ms1_2L_320_680.1000	–	13438.520	2000	22.840	74.110
ms1_3L_120_250_630.1000	–	22457.108	2000	332.990	963.130
ms1_4L_64_128_256_552.1000	–	31221.559	2000	535.470	1439.460
ms1_5L_25_55_120_250_550.1000	–	40171.410	2000	135.760	369.580
mt1_2L_650_1350.2000	–	27733.057	2000	1480.790	4646.470
mt1_3L_256_600_1144.2000	–	46828.626	2000	1181.110	3747.810
mt1_4L_120_250_520_1110.2000	–	65735.982	2000	2717.830	8120.320
mt1_5L_60_120_250_500_1070.2000	–	84263.579	2000	2118.430	6035.790
Average			2000	252.392	742.970

memory. To our knowledge, this is the most recent metaheuristic approach proposed in the literature for solving the MLUFLP that provided solutions for large-scale MLUFLP instances.

The first column of Table 9 contains the MLUFLP instance's name. The optimal solution of the current instance Opt_{sol}, obtained by CPLEX 12.1 solver, is given in the second

Table 9
Comparisons of the GA and MA approaches.

Instance	Opt_{sol}	GA					MA				
		best	$t\,(s)$	gen	*agap*(%)	$\sigma\,(\%)$	*best*	$t\,(s)$	gen	*agap*(%)	$\sigma\,(\%)$
cap71_1L_16.50	932615.750	opt	0.012	2010	0.000	0.000	opt	0	2009.5	0.000	0.000
cap71_2L_6_10.50	1813375.513	opt	0.006	2010	0.000	0.000	opt	0	2009.4	0.000	0.000
cap71_3L_2_5_9.50	4703216.306	opt	0.005	2010	0.000	0.000	opt	0	2006.9	0.000	0.000
cap101_1L_25.50	796648.438	opt	0.030	2026	0.000	0.000	opt	0.01	2035.3	0.000	0.000
cap101_2L_8_17.50	1581551.394	opt	0.018	2017	0.000	0.000	opt	0.01	2014.8	0.000	0.000
cap101_3L_3_7_15.50	3227179.813	opt	0.019	2018	0.000	0.000	opt	0.01	2013.9	0.000	0.000
cap131_1L_50.50	793439.563	opt	0.199	2140	0.000	0.000	opt	0.03	2137.1	0.000	0.000
cap131_2L_13_37.50	1592548.450	opt	0.118	2078	0.000	0.000	opt	0.03	2073.7	0.000	0.000
cap131_3L_6_14_30.50	3201970.463	opt	0.105	2108	0.125	0.084	opt	0.03	2257.1	0.071	0.087
cap131_4L_3_7_15_25.50	3630297.669	opt	0.071	2049	0.000	0.000	opt	0.03	2038.8	0.000	0.000
capa_1L_100.1000	17156454.478	opt	13.409	2319	0.000	0.000	opt	0.97	2352.9	0.000	0.000
capa_2L_30_70.1000	31524957.410	opt	8.380	2308	0.106	0.063	opt	1.07	2728.4	0.021	0.050
capa_3L_15_30_55.1000	40725103.254	opt	3.076	2120	0.000	0.000	opt	0.92	2342.5	0.000	0.000
capa_4L_6_12_24_58.1000	54643362.801	opt	4.688	2195	0.881	0.904	opt	1	2736.2	0.617	0.840
capb_1L_100.1000	12979071.581	opt	43.642	3047	0.060	0.186	opt	10.14	3317.9	0.170	0.286
capb_2L_35_65.1000	25224163.283	opt	18.414	2781	0.860	1.406	opt	5.26	3515.1	0.631	1.297
capb_3L_12_25_63.1000	34978486.506	opt	3.137	2090	0.000	0.000	opt	0.57	2111.0	0.000	0.000
capb_4L_6_13_31_50.1000	53034149.833	opt	4.652	2222	0.110	0.057	opt	1.1	2317.0	0.117	0.049
capc_1L_100.1000	11505594.329	opt	34.542	2907	0.073	0.135	opt	15.08	3054.1	0.026	0.013
capc_2L_32_68.1000	22762468.838	opt	22.625	3021	0.770	0.717	opt	2.56	3172.8	0.290	0.691
capc_3L_13_27_60.1000	35540649.433	opt	8.539	2406	0.675	1.176	opt	0.74	2340.1	0.182	0.793
capc_4L_4_9_27_60.1000	57017358.038	opt	4.601	2166	0.150	0.210	opt	1.55	2318.0	0.279	0.205
mq1_1L_300.300	3591.273	opt	14.288	2307	0.000	0.000	opt	2.22	2255.9	0.000	0.000
mq1_2L_100_200.300	8341.287	opt	19.313	2670	0.000	0.000	opt	0.61	2270.5	0.000	0.000
mq1_3L_30_80_190.300	12994.871	opt	16.980	2616	2.273	1.369	opt	0.87	2803.8	0.389	0.953
mq1_4L_20_40_80_160.300	17648.010	opt	17.423	2699	1.764	2.124	opt	1.92	3433.0	0.871	1.523
mq1_4L_18_39_81_162.300	18048.031	opt	14.876	2620	0.736	0.876	opt	3.68	3246.3	0.440	0.553
mr1_1L_500.500	–	2349.856	74.852	2595	0.000	0.000	2349.856	11.73	2452.3	0.000	0.000
mr1_2L_160_340.500	–	6707.505	83.116	2918	0.611	0.988	6707.505	5.73	2664.9	0.000	0.000
mr1_3L_55_120_325.500	–	10911.319	76.009	2858	1.341	0.814	10911.319	7.25	3245.6	0.734	0.788
mr1_4L_30_65_140_265.500	–	15311.469	61.234	2773	1.544	0.879	**15237.2605**	23	3819.5	0.871	0.418
ms1_1L_1000.1000	–	4378.632	534.888	2980	0.000	0.000	4378.632	29.4	2511.4	0.000	0.000
ms1_2L_320_680.1000	–	13416.805	540.534	3323	0.510	0.485	**13361.3895**	22.86	3226.1	0.303	0.346
ms1_3L_120_250_630.1000	–	21881.384	501.034	3228	2.260	1.312	21881.384	117.23	3376.8	1.128	0.871
ms1_4L_64_128_256_552.1000	–	30936.585	418.521	3225	2.258	1.019	**30902.742**	119.21	3438.6	1.355	1.024
ms1_5L_25_55_120_250_550.1000	–	40191.231	396.686	3105	1.738	1.118	**40070.0365**	136.01	4114.6	0.543	0.514
mt1_1L_2000.2000	–	9176.509	4134.325	3871	0.000	0.000	9176.509	350.32	1300.6	0.000	0.000
mt1_2L_650_1350.2000	–	27733.057	3949.573	4309	0.331	0.704	27733.057	819.98	1712.1	0.012	0.021
mt1_3L_256_600_1144.2000	–	46095.089	3247.064	4170	2.021	1.105	46095.09	858.48	1567.7	1.266	0.770
mt1_4L_120_250_520_1110.2000	–	65044.003	3084.885	4154	1.126	0.649	**64953.253**	1108.49	1847.1	0.667	0.400
mt1_5L_60_120_250_500_1070.2000	–	83523.753	2944.080	4144	1.678	0.830	**83404.332**	916.77	2034.8	0.997	0.692
Average			507.499	2715.1	0.600	0.480		114.422	2555.3	0.300	0.330

column. A "–" sign in the column Opt_{sol} means that no optimal solution was obtained due to memory or time limits. The remaining columns contain the results of the GA and MA approaches respectively. On each considered instance, the GA and MA method were run 20 times. For each method we present:

- the best objective function value *best*, marked *opt* in cases when the method reached optimal solution;
- average running time t (in seconds);
- average number of generations *gen*;
- average gap *agap* (in percents) of the obtained solution from the optimal Opt_{sol} or the best solution *best* (in cases when optimal solution is not known);
- standard deviation σ (in percents) of the obtained solution from the optimal Opt_{sol} or the best solution *best*;

From the results presented in Table 9, it can be seen that both GA and MA approaches reach optimal solutions previously obtained by CPLEX solver in short CPU time. In cases when no optimal solution was obtained by CPLEX, the GA and MA methods provide the same solutions in 8 cases, while in 6 cases the MA outperforms the GA in the sense of the solution quality and improves the best solutions previously obtained by the GA.

Over all tested MLUFLP instances, the average gap of the MA solution from the optimal/best-known one is $agap = 0.300\%$ and the standard deviation is $\sigma = 0.330\%$. Considering the corresponding average values for the GA approach ($agap = 0.600\%$ and $\sigma = 0.480\%$), and taking into account that the MA improved several GA's best solutions, we may conclude that the MA method achieves better average solution quality compared to the GA.

From the columns $t(s)$ in Table 9, we can see that the average MA running time over all MLUFLP instances was $t = 114.422$ s, while the average GA time was $t = 507.499$ s, which indicates the efficiency of both proposed approaches. The maximal MA's running time was 1108.49 s.

The presented computational experiments clearly demonstrate the robustness of the proposed memetic algorithm with respect to both solutions' quality and running times, even on large-scale MLUFLP instances.

6. Conclusions

This paper considers the Multilevel Uncapacitated Facility Location Problem (MLUFLP), a well-known NP-hard combinatorial optimization problem from the literature. We propose a new integer programming formulation of the problem, which uses fewer variables and constraints and which showed to be more efficient compared to existing MLUFLP formulations. Further, we propose a memetic algorithm for solving the MLUFLP, based on a new concept of applying a local search method for improving the solutions obtained by the evolutionary algorithm. Several parallelization techniques are incorporated into the proposed memetic algorithm in order to improve the CPU times of the MA runs. Fitness function calculation, the crossover operator, the mutation operator and the objective function calculation are realized in parallel. In cases when the evolutionary algorithm has run through many generations without improvement, the possibility of a premature convergence significantly increases. In these situations, the evolutionary algorithm needs an impulse in order to turn away from a local optimum trap.

The proposed memetic algorithm was subject to comparative tests including benchmark problems with up to 2000 clients and 5 levels. The MA quickly reached all known optimal and best known solutions from the literature and in case of 6 large-scale instances, the MA produced new, improved solutions. Comparing the efficiency of the parallel and the sequential variants of the proposed MA, we conclude that the parallel MA achieved considerable gains in terms of the computing time required to reach high-quality solutions. The advantages of the parallel variant of the MA become more obvious when solving large-scale problems.

The proposed memetic algorithm with parallelization strategy showed to be a suitable concept for solving the MLUFLP, especially large-scale problem instances. The obtained numerical results show that the developed MA approach represents a valuable addition to existing methods for solving the MLUFLP.

Further work will be directed to adopting the proposed memetic algorithm for solving the capacitated and other variants of the Multilevel Uncapacitated Facility Location Problem.

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

Appendix

Theorem 2. *The formulations MLUFLP-1 and MLUFLP-3 are equivalent, i.e. MLUFLP-1 $\Leftrightarrow$ MLUFLP-3.*

Proof. $\Rightarrow$: We will first prove that the MLUFLP-3 follows from the MLUFLP-1. Suppose that conditions (2)–(3) of the MLUFLP-1 formulation hold and objective (1) is considered.

(1), (2)–(3) $\Rightarrow$ (15):

Since $|F| = m$, then $\sum_{i\in F} f_i y_i = \sum_{i=1}^{m} f_i y_i$. For the second member of the objective function. i.e. the multiple sum we have:

$$\sum_{p\in P}\sum_{j\in D} c_{pj}x_{pj} = \sum_{p\in P}\sum_{j\in D: x_{pj}=1} c_{ji_k} + c_{i_k i_{k-1}} + \cdots + c_{i_2 i_1} \tag{21}$$

because, if $x_{pj} = 1$, then there is unique path $p = (i_k, i_{k-1}, \ldots, i_l, i_{l-1}, \ldots, i_2, i_1)$ which is assigned to j. Let's fix $l \in \{2, \ldots, k+1\}$ and consider the member (i_l, i_{l-1}). Look through all $p \in P$ and $j \in D$ such that $x_{pj} = 1$ and seek for the paths p that contain the member (i_l, i_{l-1}), i.e. the vertices i_l, i_{l-1} belong to p, and they are situated on the l-th and $l-1$-th places in the k-tuple that defines p (the counting starts from left to right of the k-tuple). The number of paths p that satisfy this condition will be exactly the number of clients that are supplied from the facilities i_l and i_{l-1}, situated on the l-th and $l-1$-th levels respectively. This number of clients is exactly z_{is}^{l} by the definition. The sum (21) becomes exactly

$$\sum_{l=2}^{k+1} c_{i_l i_{l-1}} z_{i_l i_{l-1}}^{l}. \tag{22}$$

If we further perform the re-numeration, i.e. for the fixed l, let's denote i_l by i and i_{l-1} by s, we obtain (15)

$$\sum_{l=2}^{k+1} c_{i_l i_{l-1}} z_{i_l i_{l-1}}^{l} = \sum_{l=2}^{k+1}\sum_{i\in F_l}\sum_{s\in F_{l-1}} c_{is} z_{is}^{l}.$$

(2)–(3) $\Rightarrow$ (16):

Fix an arbitrary $j \in D = F_{k+1}$. From (2) we have $1 = \sum_{p \in P} x_{pj}$, which means that there is exactly one path $p = (i_k, i_{k-1}, \dots, i_l, i_{l-1}, \dots, i_2, i_1)$ which is assigned to $j \in F_{k+1}$. Therefore,

$$\sum_{i_k \in F_k} z^k_{j i_k} = \sum_{i \in F_k} z^k_{ji} = 1, \quad \text{which is (16).}$$

(2)–(3) $\Rightarrow$ (17):

Let us fix an l and $i \in F_l$. By the definition of z^l_{is} we have

$$\sum_{s \in F_{l-1}} z^l_{is} = \sum_{j \in D} \sum_{p \in P : i, s \in p, s \in F_{l-1}, i \in F_l} x_{pj}.$$

According to (2)–(3), for every $j \in D$ there is unique established path $p \in P$ such that $x_{pj} = 1$. For every such path p that (in addition) contains fixed $i \in F_l$ and $s \in F_{l-1}$, there is unique $r \in p$ such that $r \in F_{l+1}$. It means that the flow originating from $j \in D$, which is shipped via facilities $s \in F_{l-1}$ and $i \in F_l$, it must be further distributed from $i \in F_l$ to some $r \in F_{l+1}$. Since there is only and only one path $p \in P$ that is assigned to client $j \in D$, the facility $r \in p, r \in F_{l+1}$ is unique. Therefore, we obtain:

$$\sum_{j \in D} \sum_{p \in P : i, r \in p, i \in F_l, r \in F_{l+1}} x_{pj} = \sum_{r \in F_{l+1}} z^{l+1}_{ri},$$

and (17) is proven.

(2)–(3) $\Rightarrow$ (18):

Let's fix $l \in \{1, \dots, k\}$, $r \in F_{l+1}$, $i \in F_l$. If $l = k$, it means that $r \in F_{k+1} = D$. There are two possibilities for $i \in F_k$: it is established ($y_i = 1$) or not ($y_i = 0$). If $y_i = 0$, it is not possible to assign client r to the unestablished facility (condition (3)). It means that $z^{k+1}_{ri} = 0$, and hence, $0 = z^{k+1}_{ri} \leqslant n y_i = 0$ holds. If $y_i = 1$, and the flow goes from facility r to $i \in F_k$, then, regarding (2) there is unique path $p \in P$ (containing $i \in F_k$) such that $x_{pj} = 1$, it follows that $1 = z^{k+1}_{ri}$ and $1 = z^{k+1}_{ri} \leqslant n y_i = n$ is true.

If $l \leqslant k$, for fixed $r \in F_{l+1}$, $i \in F_l$ we again consider two possibilities for y_i.

If $y_i = 0$, again by (3), it is not possible to construct a path $p \in P$, that will conduct the flow originating from some client $j \in D$ via unestablished facility $i \in F_l$ and facilities $r \in F_{l+1}$ (whether r is established or not). It means that $z^{l+1}_{ri} = 0$, and hence, $0 = z^{l+1}_{ri} \leqslant n y_i = n$ holds.

If $y_i = 1$, then we consider two possibilities for $r \in F_{l+1}$:

If $y_r = 0$, there is no $p \in P$, that will conduct the flow originating from some client $j \in D$ via facility $i \in F_l$ and unestablished facility $r \in F_{l+1}$. Hence, $z^{l+1}_{ri} = 0$, and, $0 = z^{l+1}_{ri} \leqslant n y_i = n$ holds.

If $y_r = 1$, then both $i \in F_l$ and $r \in F_{l+1}$ are established, and hence, it is possible to construct a path $p \in P$, that conducts the flow originating from some client $j \in D$

via facility $r \in F_{l+1}$ and $i \in F_l$. Since, there are most $|D| = n$ clients, in the best case they are all assigned to (possibly different) paths containing $r \in F_{l+1}$ and $i \in F_l$. Hence, $z_{ri}^{l+1} \leqslant n = ny_i$ holds.

$\Leftarrow$: Let's now prove that the MLUFLP-1 follows from the MLUFLP-3. Suppose that conditions (16)–(18) of the MLUFLP-3 formulation hold and objective (15) is considered.

(16)–(18) $\Rightarrow$ (2):

Let's fix $j \in D$. From (16) we have:

$$1 = \sum_{i \in F_k} z_{ji}^k = \sum_{p \in P: i \in p,\, i \in F_k} x_{pj},$$

which means that exists exactly one path $p = (i, \ldots)$ that is assigned to j which "starts" with $i \in F_k$. By using (17) we obtain:

$$\sum_{i \in F_k} z_{ji}^k = \sum_{s \in F_{k-1}} z_{is}^{k-1} = \sum_{t \in F_{k-2}} z_{ti}^{k-2} = \ldots = \sum_{i \in F_1} z_{ji}^1,$$

which means that there is exactly one facility at each level that defines a path $p = (i, \ldots)$ assigned to j. Therefore, this path is unique, i.e.

$$1 = \sum_{p \in P} x_{pj}.$$

(15), (16)–(18) $\Rightarrow$ (1):

Since $|F| = m$, then $\sum_{i=1}^m f_i y_i = \sum_{i \in F} f_i y_i$. For the second member of the objective function. i.e. the multiple sum we have:

$$\begin{aligned}\sum_{l=2}^{k+1} \sum_{i \in F_l} \sum_{s \in F_{l-1}} c_{is} z_{is}^l &= \sum_{l=2}^{k+1} \sum_{p \in P: i, s \in P,\, i \in F_l,\, s \in F_{l-1}} \sum_{j \in D} c_{is} x_{pj} \\ &= \sum_{j \in D} \sum_{l=2}^{k+1} \sum_{p \in P: i,\, s \in P,\, i \in F_l,\, s \in F_{l-1}} c_{is} x_{pj}.\end{aligned}$$

For every $j \in D$ we have a path p assigned to j (by the definition of feasible solution). According to (2), that is already proven above, this path p is unique. If facilities i, s belong to p ($(i, s) \in p$), such that $i \in F_l$, $s \in F_{l-1}$, we add the cost c_{is}. By going through all levels $l = 2, \ldots, k+1$ we construct the sum c_{pj}. Note that for $l = k+1$, $j \equiv i$. Hence:

$$\sum_{j \in D} \sum_{l=2}^{k+1} \sum_{p \in P: i, s \in P,\, i \in F_l,\, s \in F_{l-1}} c_{is} x_{pj} = \sum_{j \in D} \sum_{p \in P} c_{pj} x_{pj},$$

that is exactly the second member of (1).

(16)–(18) $\Rightarrow$ (3):

For every $i \in F$ we have two possibilities for $i \in F$.

If facility $i \in F$ is not established, then, according to (18) for every client $j \in D$ we have no paths assigned to j that contain i. Hence, $x_{pj} = 0$ for all $p \in P$ such that $i \in p$ and all $j \in D$. Therefore, $0 = \sum_{p \in P: i \in p} x_{pj} \leqslant y_i = 0$ holds for every $j \in D$.

If facility $i \in F$ is established, then $y_i = 1$. Let us assume the opposite, i.e. there exists $j \in D$, such that $\sum_{p \in i} x_{pj} > y_i = 1$. Then there exist at least two paths $p_1, p_2 \in P$ that are assigned to $j \in D$ and both contain facility i: $i \in p_1, p_2$. That is a contradiction with (2), which is already proven above.

Therefore, (3) holds for every $i \in F$, $j \in D$. □

References

Addis, B., Carello, G., Ceselli, A. (2102). Exactly solving a two-level location problem with modular node capacities. *Networks*, 59(1), 161–180.

Aardal, K., Labbé, M., Leung, J., Queyranne, M. (1996). On the two-level uncapacitated facility location problem. *INFORMS J. Comput.*, 8, 289–301.

Aardal, K., Chudak, F., Shmoys, D.B. (1999). A 3-approximation algorithm for the k-level uncapacitated facility location problem. *Information Processing Letters*, 72, 161–167.

Ageev, A. (2002). Improved approximation algorithms for multilevel facility location problems. *Operations Research Letters*, 30, 327–332.

Ageev, A., Ye, Y., Zhang, J. (2005) Improved combinatorial approximation algorithms for the k-level facility location problem. *SIAM Journal on Discrete Mathematics*, 18, 207–217.

Beasley, J.E. (1990). OR Library: distributing test problems by electronic mail. *Journal of the Operational Research Society*, 41, 1069–1072.

Beasley, J.E. (1996). Obtaining test problems via internet. *Journal of Global Optimization*, 8, 429–433.

Bloemhof-Ruwaard, J., Salomon, M., Van Wassenhove, L.N. (1996). The capacitated distribution and waste disposal problem. *European Journal of Operational Research*, 88, 490–503.

Bumb, A.F., Kern, W. (2001). A simple dual ascent algorithm for the multilevel facility location problem. *Electronic Notes in Discrete Mathematics*, 8, 14–17.

Blum, C., Puchinger, J., Raidl G.R. (2011). Hybrid metaheuristics in combinatorial optimization: a survey. *Applied Soft Computing*, 11(6), 4135–4151.

Canel, C, Khumawala, B., Law, J., Loh, A. (2001). An algorithm for the capacitated, multicommodity, multiperiod facility location problem. *Computers and Operations Research*, 28, 411–427.

Charikar, M., Guha, S. (1999). Improved combinatorial algorithms for the facility location and k-median problems. In: *Proceedings of the 40th Annual Symposioum on Foundations of Computer Science (FOCS)*, pp. 378–388.

Chen, Y.W., Lu, Y.Z., Yang, G.K. (2008). Hybrid evolutionary algorithm with marriage of genetic algorithm and extremal optimization for production scheduling. *The International Journal of Advanced Manufacturing Technology*, 36(9–10), 959–968.

Chudak, F., Shmoys, D. (1999). Improved approximation algorithms for capacitated facility location problem. *Lecture Notes in Computer Science*, 1610, 99–113.

Chudak, F., Shmoys, D. (2003). Improved approximation algorithms for the uncapacitated facility location problem. *SIAM Journal of Computing*, 33(1), 1–25.

Dias, J., Captivo, M., Clímaco, J. (2008). Dynamic multi-level capacitated and uncapacitated location problems: an approach using primal-dual heuristics. *Operational Research*, 7(3), 345–379.

Drezner, Z., Hamacher, H. (2002). *Location Theory: Applications and Theory*. Springer, Berlin–Heidelberg.

Eitan, Y., Narula, S.C., Tien, J. (1991). A generalized approach to modelling the hierarchical location–allocation problem. *IEEE Transactions on Systems Man and Cybernetics*, 21, 39–46.

Edwards, N.J. (2001). *Approximation algorithms for the multi-level facility location problem*. PhD thesis, Cornell University.

Espejo, L.G., Galvão, R.D., Boffey, B. (2003). Dual-based heuristics for a hierarchical covering location problem. *Computers and Operations Research*, 30, 165–180.

Fan, S.K.S., Liang, Y.C., Zahara, E. (2006). A genetic algorithm and a particle swarm optimizer hybridized with Nelder–Mead simplex search. *Computers & Industrial Engineering*, 50(4), 401–425.

Filipović, V. (2003). Fine-grained tournament selection operator in genetic algorithms. *Computing and Informatics*, 22, 143–161.

Gabor, A.F., Jan-Kees, C., van Ommerenb, C.W. (2010). A new approximation algorithm for the multilevel facility location problem. *Discrete Applied Mathematics*, 158, 453–460.

Galvão, R.D., Espejo, L.G., Boffey, B. (2002). A hierarchical model for the location of perinatal facilities in the municipality of Rio de Janeiro. *European Journal of Operational Research*, 138, 495–517.

Hart, W. (1994). *Adaptive global optimization with local search*. PhD thesis, University of California, San Diego.

Hinojosa, Y., Puerto, J., Fernández, F.R. (2000). A multiperiod two-echelon multicommodity capacitated plant location problem. *European Journal of Operational Research*, 123, 271–291.

Klose, A. (1999). An LP-based heuristic for two-stage capacitated facility location problems. *Journal of the Operational Research Society*, 50, 157–166.

Klose, A. (2000). A Lagrangean relax-and-cut approach for the two-stage capacitated facility location problem. *European Journal of Operational Research*, 126, 408–421.

Krarup, J., Pruzan, P.M. (1983). The simple plant location problem: survey and synthesis. *European Journal of Operational Research*, 12, 36–81.

Krishnaswamy, R., Sviridenko, M. (2012). Inapproximability of the multi-level uncapacitated facility location problem. In: *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2012)*, pp 718–734.

Lunday, B.J., Sherali, H.D. (2010). A dynamic network interdiction problem. *Informatica*, 21(4), 553–574.

Marić, M. (2010). An efficient genetic algorithm for solving the multi-level uncapacitated facility location problem. *Computing and Informatics*, 29, 183–201.

Melachrinoudis, E., Min, H. (2012). The dynamic relocation and phase-out of a hybrid, two-echelon plant/warehouse facility: a multiple objective approach. *European Journal of Operational Research*, 123, 1–15.

Misevičus, A. (2006). Experiments with hybrid genetic algorithm for the grey pattern problem. *Informatica*, 17(2), 237–258.

Misevičus, A., Rubliauskas, D. (2009). Testing of hybrid genetic algorithms for structured quadratic assignment problems. *Informatica*, 20(2), 255–272.

Moscato, P.A., Cotta, C. (2003). A gentle introduction to memetic algorithms. In: Glover, F., Kochenberger, G. (Eds), *Handbook of Metaheuristics*. Kluwer Academic, Boston, pp. 105–144.

Moscato, P.A., Cotta, C. (2007) Memetic algorithms. In: Gonzáles, T. (Ed.), *Handbook of Approximation Algorithms and Metaheuristics*. Taylor & Francis, London, Part II, 27–1.

Neri, F., Cotta, C., Moscato, P. (2012). Handbook of Memetic Algorithms. *Studies in Computational Intelligence*, Vol. 379. Springer, Berlin–Heidelberg.

Pirkul, H., Jayaraman, V. (2000). A multi-commodity, multi-plant, capacitated facility location problem: formulation and efficient heuristic solution. *Computers and Operations Research*, 25, 869–878.

Prestwich, S., Tarim, S., Rossi, R., Hnich, B. (2009). Evolving parameterised policies for stochastic constraint programming. In: Gent, I. (Ed.), *Principles and Practice of Constraint Programming-CP 2009*, *Lecture Notes in Computer Science*, Vol. 5732. Springer, Berlin–Heidelberg, pp. 684–691.

Raidl, G.R., Gottlieb, J. (2005). Empirical analysis of locality, heritability and heuristic bias in evolutionary algorithms: a case study for the multidimensional knapsack problem. *Evolutionary Computation*, 13(4), 441–475.

Resende, M.G., Werneck, R.F. (2006). A hybrid multistart heuristic for the uncapacitated facility location problem. *European Journal of Operational Research*, 174(1), 54–68.

ReVelle, C.S., Eiselt, H.A. (2005). Location analysis: a synthesis and survey. *European Journal of Operational Research*, 165, 1–19.

Stanimirović, Z., Kratica, J., Dugošija, Dj. (2007). Genetic algorithms for solving the discrete ordered median problem. *European Journal of Operational Research*, 182, 983–1001.

Tragantalerngsak, S., Holt, J., Rönnqvist, M. (2007). Lagrangian heuristics for the two-echelon, single-source, capacitated facility location problem. *European Journal of Operational Research*, 102, 611–625.

Tragantalerngsak, S., Holt, J., Rönnqvist, M. (2000). An exact method for the two-echelon, single-source, capacitated facility location problem. *European Journal of Operational Research*, 123, 473–489.

Villegas, J.G., Palacios, F., Medaglia, A.L. (2006). Solution methods for the bi-objective (cost-coverage) unconstrained facility location problem with an illustrative example. *Annals of Operations Research*, 147(1), 109–141.

Zhang, J. (2006). Approximating the two-level facility location problem via a quasi-greedy approach. *Mathematical Programming*, 108, 159–176.

M. Marić PhD, is Assistant Professor at the Department for Informatics and Computer Science at Faculty of Mathematics, University of Belgrade. He received his BSc degree (2002), MSc (2006) and PhD (2008) in computer science from University of Belgrade, Faculty of Mathematics. His research interests include metaheuristic optimization algorithms, parallel algorithms, operational research, bioinformatics, computer graphics.

Z. Stanimirović, PhD, 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 PhD 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, Evolutionary Algorithms and Mathematical Modeling.

A. Djenić is a PhD 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 trainee at the Faculty of Mathematics, and also as a part-time researcher at the Mathematical Institute SANU in Belgrade. His research is mainly focused on Mathematical Programming Methods and Optimization.

P. Stanojević is a PhD student at the Department of Informatics and Computer Science, Faculty of Mathematics, University of Belgrade. At the same Faculty, he received his B.Sc. degree in 2008 in Computer Science. His research areas are optimization algorithms, parallel algorithms and game theory.

Daugelio lygių vienodo pajėgumo aptarnavimo centrų išdėstymo memetinis algoritmas

Miroslav MARIĆ, Zorica STANIMIROVIĆ, Aleksandar DJENIĆ,
Predrag STANOJEVIĆ

Straipsnyje nagrinėjamas daugelio lygių vienodo pajėgumo aptarnavimo centrų išdėstymo uždavinys (MLUFLP). Šiam uždaviniui pasiūlytas naujas sveikaskaitinio programavimo modelis, kuris įgalina IBM firmos optimizatorių CPLEX rasti optimalius sprendinius, nerastus iki šiol. Taip pat sukurtas memetinis (hibridinis) algoritmas, naudojantis naują lokaliosios paieškos strategiją. Pateikti eksperimentai rodo, kad šis algoritmas greitai suranda optimalius arba iki šiol žinomus geriausius testinių MLUFLP uždavinių sprendinius ir pagerina kelių didelių MLUFLP testinių uždavinių sprendinius.