www.elsevier.com/locate/caor

The Multimode Covering Location Problem

Fabio Colombo, Roberto Cordone, Guglielmo Lulli

PII: S0305-0548(15)00213-0
DOI: http://dx.doi.org/10.1016/j.cor.2015.09.003
Reference: CAOR3854

To appear in: *Computers and Operation Research*

Received date: 6 October 2014
Revised date: 1 July 2015
Accepted date: 5 September 2015

Cite this article as: Fabio Colombo, Roberto Cordone and Guglielmo Lulli, Th Multimode Covering Location Problem, *Computers and Operation Research* http://dx.doi.org/10.1016/j.cor.2015.09.003

The Multimode Covering Location Problem.

Fabio Colombo* Roberto Cordone† Guglielmo Lulli‡

September 15, 2015

Abstract

In this paper we introduce the *Multimode Covering Location Problem*. This is a generalization of the Maximal Covering Location Problem that consists in locating a given number of facilities of different types with a limitation on the number of facilities sharing the same site.

The problem is challenging and intrinsically much harder than its basic version. Nevertheless, it admits a constant factor approximation guarantee, which can be achieved combining two greedy algorithms. To improve the greedy solutions, we have developed a *Variable Neighborhood Search* approach, based on an exponential-size neighborhood. This algorithm computes good quality solutions in short computational time. The viability of the approach here proposed is also corroborated by a comparison with a *Heuristic Concentration* algorithm, which is presently the most effective approach to solve large instances of the Maximal Covering Location Problem.

1 Introduction.

A facility location problem consists in placing a number of facilities to serve a set of demand centers, whose positions are known, while optimizing a given objective function. The problem admits several variants, based on the objective of the decision maker and on the application setting. For a complete taxonomy of facility location problems, the interested reader may refer to ReVelle *et al.* [35].

In this paper we focus on a generalized version of the *Maximal Covering Location Problem* (*MCLP*), first proposed by Church and ReVelle [8]. The *MCLP* belongs to the class of discrete location problems, i.e., problems with a finite set of demand centers and a finite set of candidate locations. The *MCLP* does not require all the demand centers to be served: its purpose is to locate a given number of facilities maximizing the number, or the total weight, of

*University of Milano, Department of Computer Science, via Comelico 39, 20135 Milano, Italy, fabio.colombo2@unimi.it

†University of Milano, Department of Computer Science, via Comelico 39, 20135 Milano, Italy, roberto.cordone@unimi.it

‡University of Milano "Bicocca", Department of Informatics, Systems and Communication, viale Sarca 336, 20122 Milano, Italy, lulli@disco.unimib.it

the served demand centers. Because of its wide applicability in the real world, especially in the planning of service and emergency facilities, the *MCLP* is a well-studied problem. Chung [7] reviewed several other applications of the *MCLP*, such as data abstraction, stock selection and classification problems. Other interesting applications are those described by Dwyer and Evans [18] for the selection of mailing lists, Daskin et al. [14] for flexible manufacturing, Hougland and Stephens [28] for air pollution control and Melo et al. [30] for supply chain management.

Since its proposal, the *MCLP* has been generalized in different ways. Berman et al. [5] reviewed gradual cover models, cooperative cover models and variable radius models. Ghiani et al. [25] introduced a capacitated plant location problem where multiple facilities can be opened in the same site. Rajagopalan, Saydam and Xiao [34] considered a multiperiod set covering location model in the field of application of emergency medical services. Dell'Olmo et al. [16] tackled the optimal location of intersection safety cameras on an urban traffic network to minimize the impact of car accidents, through a multiperiod variant of maximal covering location.

In this paper, we present the *Multimode Covering Location Problem* (*MM-CLP*). This problem consists in placing a given number of facilities of different types (hereafter called *modes*) to serve demand centers that require different types of service. The goal is to maximize the demand coverage over all the considered modes. An additional restriction with respect to the *MCLP* is that only a limited number of different modes can be activated in each candidate facility site. A similar generalization for the uncapacitated facility location problem has been recently proposed by Arora et al. [2]. They present a 4-approximation LP-rounding based algorithm for a class of problems with only two modes.

Possible applications of the *MM-CLP* refer to the distribution of facilities addressed to different users in the same area (e. g., fire stations and police stations). A similar situation occurs in the location of a heterogeneous fleet of ambulances, some of which might have an equipment and crew specialized in the treatment of heart-strokes or other severe health conditions. Another common application of covering location problems is in the design of nature reserves for the protection of biodiversity: each land parcel can be subject to different types of protection, with a different impact on the endangered species which populate the parcel. Finally, telecommunication antennas with different radiuses and pointing in different directions can be installed in the same site, covering different subsets of users, but the number of antennas in each site can be limited by the available space and by the need to avoid reciprocal interferences.

The *MM-CLP* is $\mathcal{NP}$-hard because it includes the *MCLP* as a special case. However, while medium-size instances of the *MCLP* can be quickly solved by commercial solvers, even fairly small instances of the *MM-CLP* prove much harder. Nevertheless, we here prove that the *MM-CLP* admits two greedy algorithms with a constant factor approximation guarantee. This extends in a nontrivial way a similar property of the *MCLP*. To compute better solutions, we also present a *Variable Neighborhood Search* (*VNS*) algorithm, which imple-

ments a *Very Large Scale Neighborhood Search* (*VLNS*) as its basic local search procedure. The hybridization of *VNS* with other metaheuristic approaches is an active field of research, see for example [11, 27, 3]. To evaluate the performance of the proposed *VNS* approach, we have first compared it to a simpler *VNS* implementation based on a polynomial-size neighbourhood. Then, we have implemented an alternative approach, based on the *Heuristic Concentration* (*HC*) framework developed by ReVelle et al. [36]. To the best of our knowledge, *HC* is considered the state-of-the-art heuristic for the *MCLP*.

The remainder of the paper is organized as follows. In Section 2 we formally define the problem, through a mathematical programming formulation. The complexity and approximation properties of the *MM-CLP* are described in detail in Section 2.1. In Section 3, we describe the *VNS* framework. Section 4 reports a computational comparison of the *VNS* and *HC* algorithms on a set of randomly generated benchmark instances, showing that the former clearly outperforms the latter. Finally, Section 5 draws some conclusions.

2 Mathematical model.

Let I be a set of demand centers, J a set of candidate facility sites and M a set of modes. The relation between facility sites and demand centers in each mode can be represented with a binary matrix: $a_{ijm} = 1$ if facility site j is able to serve demand center i in mode m and $a_{ijm} = 0$ otherwise. For each candidate facility site $j \in J$, there is a maximum number b_j of modes that can be activated on the site. The number of facility sites used in each mode, K_m, is given and a weight w_{im} is assigned to each demand center $i \in I$ and mode $m \in M$. The *MM-CLP* requires to find a subset of facility sites for each mode, such that the total weight of the served demand centers is maximum.

Let $x_{jm} = 1$ if a facility of mode $m \in M$ is located on site $j \in J$, $x_{jm} = 0$ otherwise; $y_{im} = 1$ if demand center $i \in I$ is served in mode $m \in M$, $y_{im} = 0$ otherwise. The *MM-CLP* can be formulated as follows.

$$\max z = \sum_{i \in I} \sum_{m \in M} w_{im} y_{im} \tag{1a}$$

$$\sum_{j \in J} a_{ijm} x_{jm} \geq y_{im} \qquad i \in I, m \in M \tag{1b}$$

$$\sum_{j \in J} x_{jm} = K_m \qquad m \in M \tag{1c}$$

$$\sum_{m \in M} x_{jm} \leq b_j \qquad j \in J \tag{1d}$$

$$x_{jm} \in \{0, 1\} \qquad j \in J, m \in M \tag{1e}$$

$$y_{im} \in \{0, 1\} \qquad i \in I, m \in M \tag{1f}$$

The objective function (1a) maximizes the total weight of the served demand centers. The covering constraints (1b) link the x and y variables. For each mode,

Constraints (1c) fix the number of facilities to be placed. Constraints (1d) set the maximum number of facilities (of different modes) that can be located in each site. The integrality of the x and y variables is imposed by Constraints (1e) and (1f).

We here add a few remarks about the formulation. First, note that, once the x variables are fixed, the objective function and the covering constraints implicitly assign integer values to the y variables. Therefore, Constraints (1f) can be relaxed to $0 \leq y_{im} \leq 1$ for all $i \in I, m \in M$. Second, the feasibility of the problem depends only on the cardinality constraints (1c) and (1d): the problem is feasible if and only if $\sum_{m \in M} K_m \leq \sum_{j \in J} b_j$. From the practical point of view this condition is in general satisfied, because the number of candidate facility sites exceeds the number of facilities to be located. If the problem is feasible, Constraints (1c) can be relaxed to $\leq$ inequalities.

2.1 Complexity and approximation properties

The decision version of the *MM-CLP* is $\mathcal{NP}$-complete, because the special case in which the set of modes is a singleton ($|M| = 1$) coincides with the *MCLP*. An alternative reduction to the *MCLP* can be obtained allowing each candidate site to use all available modes ($b_j = |M|$ for all $j \in J$). Under this assumption, in fact, Constraints (1d) are redundant and the problem decomposes into $|M|$ independent instances of the *MCLP*, one for each mode.

The *MM-CLP* also includes as a special case the *k-MCLP*, which requires to compute k disjoint solution of the *MCLP* such that the sum of their values is maximum. This problem corresponds to instances of the *MM-CLP* in which the number of modes is fixed to k, only one facility can be located in each site ($b_j = 1$) and the facilities serve the same demand centers for all the modes.

In what follows, we present approximation properties for the *MM-CLP*. These properties generalize the approximation results provided by Vohra and Hall [39] for the *MCLP*. A maximization problem is α-approximable when it admits a polynomial time algorithm that provides on each instance P a solution $\tilde{z}(P)$ such that $\tilde{z}(P)/z^*(P) \geq \alpha$, where $z^*(P)$ is the value of the optimal solution of P.

With the purpose of establishing the approximation results, we first present two greedy algorithms. As it is customary for the *MCLP*, in what follows we will also denote the facility sites as *columns* and the demand centers as *rows*.

Greedy algorithms

Algorithm *Greedy1* (see Figure 1 for its pseudocode) generates a solution placing for each mode $m \in M$ the required number of facilities K_m one by one.

We denote as I_{jm} the subset of rows that are currently uncovered and would be covered by setting $x_{jm} = 1$. At first, $I_{jm} = \{i \in I : a_{ijm} = 1\}$. At each inner loop iteration, the algorithm selects the column j^* that, in the current mode m, covers the subset of rows I_{j^*m} of maximum weight, sets the corresponding x variable to one and updates all subsets I_{jm} removing the rows covered by j^*. If

column j^* has been selected b_{j^*} times, it is forbidden in the following iterations. As already observed, if $\sum_{j\in J} b_j \geq \sum_{m\in M} K_m$, Algorithm *Greedy1* terminates with a feasible solution.

Algorithm *Greedy1*(I, J, M, a, b, K, w)
$x_{jm} := 0$ for all $j \in J$, $m \in M$;
$I_{jm} = \{i \in I : a_{ijm} = 1\}$ for all $j \in J$, $m \in M$;
for $m := 1$ to $|M|$ **do**
  for $j := 1$ to K_m **do**
    $j^* := \arg\max_{j\in J} \sum_{i\in I_{jm}} w_{im}$;
    $x_{j^*m} := 1$;
    $I_{jm} := I_{jm} \setminus I_{j^*m}$ for all $j \in J$;
    if $\sum_{m\in M} x_{j^*m} = b_{j^*}$ **then** $J := J \setminus \{j^*\}$;
  end for
end for
return x;

Figure 1: Pseudocode of Algorithm *Greedy1*.

Algorithm *Greedy2* (whose pseudocode is given in Figure 2) builds a solution similarly to Algorithm *Greedy1*, with the only difference that it relaxes Constraints (1d). Therefore, the solution obtained after the first loop can be unfeasible, because some columns can be selected in more than b_j modes. The second loop restores feasibility. For each column j selected more than b_j times, the algorithm computes the subset C_{jm} of rows that are covered in mode m in the current solution only by j. It sets to zero the x variable corresponding to the mode for which C_{jm} has the minimum total weight. The `while` loop terminates when column j is selected b_j times.

Approximation results

The approximation properties of Algorithm *Greedy1* require the following technical assumption.

Definition 1 *Coverability assumption: for each mode $m \in M$ there exists a sufficiently large value $\tilde{K}_m$ such that Algorithm* Greedy1 *returns a feasible solution covering all the rows.*

This assumption is trivial when there is a single mode, as it corresponds to requiring each row to be covered by at least one column. In the multimode case, the coverability assumption might not be satisfied due to the limitation on the maximum number of facilities that can be located on a site. In practice, however, this assumption is largely satisfied.

Algorithm *Greedy2*(I, J, M, a, b, K, w)
$x_{jm} := 0$ for all $j \in J$, $m \in M$;
$I_{jm} = \{i \in I : a_{ijm} = 1\}$ for all $j \in J$, $m \in M$;
for $m := 1$ to $|M|$ **do**
  for $j := 1$ to K_m **do**
    $j^* := \arg\max_{j \in J} \sum_{i \in I_{jm}} w_{im}$;
    $x_{j^*m} := 1$;
    $I_{jm} := I_{jm} \setminus I_{j^*m}$ for all $j \in J$;
  end for
end for
for $j := 1$ to $|J|$ **do**
  $C_{jm} = \{i \in I : a_{ijm}x_{jm} = 1$ and $a_{irm}x_{rm} = 0 \forall r \neq j\}$ for all $m \in M$;
  while $\sum_{m \in M} x_{jm} > b_j$ **do**
    $m^* := \arg\min_{m \in M} \sum_{i \in C_{jm}} w_{im}$;
    $x_{jm^*} := 0$;
  end while
end for
return x;

Figure 2: Pseudocode of Algorithm *Greedy2*

Theorem 1 *Under the coverability assumption, Algorithm* Greedy1 *computes a solution of* MM-CLP *with a guaranteed approximation factor of*

$$\alpha_1 = \frac{\sum_{m \in M} K_m W_m}{|J| W_{\text{tot}}}$$

where $W_m = \sum_{i \in I} w_{im}$ *is the total weight of all rows in mode* $m \in M$ *and* $W_{\text{tot}} = \sum_{m \in M} W_m$ *is the total weight of all rows in all modes.*

Proof. Consider the inner loop of Algorithm *Greedy1*, which is performed on each single mode. By construction, the weights of the columns selected form a nonincreasing sequence. In fact, at each step the algorithm selects the column j^* which provides the maximum additional contribution to the objective function and updates the potential contributions of the other columns reducing them to account for the rows covered by j^*. In the algorithm, the process stops after K_m iterations and the total weight of the covered rows is $z_m^{H_1}$. If the process were continued for $\tilde{K}_m$ iterations, thanks to the coverability assumption, all the rows would be covered, thus generating a sequence of nonincreasing values summing up to W_m. Monotonicity implies that the average of the first K_m values exceeds the average of the overall sequence:

$$\frac{z_m^{H_1}}{K_m} \geq \frac{W_m}{\tilde{K}_m}$$

Consequently, the value of the solution returned by Algorithm *Greedy1* is

$$z^{H_1} = \sum_{m \in M} z_m^{H_1} \geq \sum_{m \in M} \frac{K_m W_m}{\tilde{K}_m}$$

and since the total weight of all rows in all modes exceeds the optimum of the problem ($W_{\text{tot}} \geq z^*$) and $\tilde{K}_m \leq |J|$

$$\frac{z^{H_1}}{z^*} \geq \sum_{m \in M} \frac{K_m W_m}{\tilde{K}_m W_{\text{tot}}} \geq \frac{\sum\limits_{m \in M} K_m W_m}{|J| W_{\text{tot}}}$$

which is the thesis. ■

In the specific case in which all modes have the same total weight ($W_m = W$) and require the same number of facilities ($K_m = K$), $\alpha_1 = K/|J|$. If all columns can be selected in one single mode ($b_j = 1$), the approximation can be refined.

Corollary 1 *Under the coverability assumption, if $b_j = 1$ for all $j \in J$, $K_m = K$ and $W_m = W$ for all $m \in M$, Algorithm* Greedy1 *provides a constant approximation factor equal to*

$$\alpha_1' = \frac{K}{|J|} \left(\frac{1}{|M|} + \frac{|J|}{K|M|} \ln \frac{1}{1 - \frac{K}{|J|}(|M| - 1)} \right)$$

Proof. Under the simplifying assumptions of this corollary, the value of $\tilde{K}_m$ is also bounded by the difference between the maximum number of facilities that can be placed and the number of facilities that have already been placed in modes $m' = 1, \ldots, m-1$

$$\tilde{K}_m \leq \sum_{j \in J} b_j - \sum_{m'=1}^{m-1} K_{m'} = |J| - (m-1)\,K$$

Notice that this estimate is always strictly positive, because, by assumption, $\sum_{j \in J} b_j \geq \sum_{m'=1}^{|M|} K_m > \sum_{m'=1}^{m-1} K_{m'}$. Consequently,

$$\begin{aligned} \frac{z^{H_1}}{z^*} &\geq \sum_{m \in M} \frac{K_m W_m}{\tilde{K}_m W_{\text{tot}}} \geq \sum_{m \in M} \frac{K\,W}{(|J| - (m-1)\,K)\,|M| W} \geq \\ &\geq \frac{K}{|M||J|} \sum_{m=1}^{|M|} \frac{1}{1 - \frac{K}{|J|}\,(m-1)} \end{aligned}$$

which can be approximated from below by using an integral approximation [10]

$$\begin{aligned} \alpha_1' = \frac{z^{H_1}}{z^*} &\geq \frac{K}{|M||J|} \left(1 + \int_{x=2}^{|M|+1} \frac{1}{1 - \frac{K}{|J|}(x-2)}\,dx \right) \\ &= \frac{K}{|M||J|} \left(1 + \frac{|J|}{K} \ln \frac{1}{1 - \frac{K}{|J|}(|M| - 1)} \right) \end{aligned}$$

■

Remark 1 *For $|M| = 1$, the approximation factor α'_1 is identical to α_1; for any $|M| > 1$, it is strictly stronger.*

In view of the hypothesis of Corollary 1 ($b_j = 1$ for all $j \in J$), the *k-MCLP* is also approximable.

Remark 2 *Algorithm* Greedy1 *provides a constant approximation factor α'_1 for the k-*MCLP*.*

Contrary to *Greedy1*, Algorithm *Greedy2* provides an approximation guarantee for which the coverability assumption is not required.

Theorem 2 *Algorithm* Greedy2 *computes a solution of* MM-CLP *with a guaranteed approximation factor of*

$$\alpha_2 = \frac{b_{\min}}{|M|}\left[1 - \left(1 - \frac{1}{K_{\min}}\right)^{K_{\min}}\right]$$

where $b_{\min} = \min_{j \in J} b_j$ and $K_{\min} = \min_{m \in M} K_m$.

Proof. The first loop of Algorithm *Greedy2* solves the *MM-CLP* as $|M|$ independent instances of the *MCLP*, one for each mode, by relaxing Constraints (1d). Each of these instances is solved applying the algorithm proposed in [39], which is the classical greedy algorithm for the optimization of submodular set functions, with an approximation factor equal to:

$$\frac{z_m^{H_2}}{z_m^*} \geq \left[1 - \left(1 - \frac{1}{K_m}\right)^{K_m}\right]$$

where $z_m^{H_2}$ and z_m^* are, respectively, the value obtained by this algorithm and the optimal value for the *MCLP* instance associated to mode m. The total value of the objective function after the first loop of Algorithm *Greedy2* is

$$\sum_{m=1}^{|M|} z_m^{H_2} \geq \sum_{m=1}^{|M|}\left[1 - \left(1 - \frac{1}{K_m}\right)^{K_m}\right] z_m^* \geq \left[1 - \left(1 - \frac{1}{K_{\min}}\right)^{K_{\min}}\right] \sum_{m=1}^{|M|} z_m^*$$

where $K_{\min} = \min_{m \in M} K_m$. The second inequality holds because $1-(1-1/K_m)^{K_m}$ is a monotonically increasing function of K_m.

The second loop of Algorithm *Greedy2* removes for each column $j \in J$ at most $|M| - b_j$ modes, which are selected as those which give the smallest contribution to the objective function. Therefore, the remaining modes provide a fraction $\geq b_j/|M|$ of the original objective function. Consequently, Algorithm *Greedy2* returns a value

$$z^{H_2} \geq \frac{b_{\min}}{|M|} \sum_{m=1}^{|M|} z_m^{H_2} \geq \frac{b_{\min}}{|M|}\left[1 - \left(1 - \frac{1}{K_{\min}}\right)^{K_{\min}}\right] \sum_{m=1}^{|M|} z_m^*$$

and since $\sum_{m=1}^{|M|} z_m^*$ is an upper bound on the optimum z^*

$$\alpha_2 = \frac{z^{H_2}}{z^*} \geq \frac{b_{\min}}{|M|}\left[1 - \left(1 - \frac{1}{K_{\min}}\right)^{K_{\min}}\right]$$

■

3 A Heuristic for the Multimode Covering Location Problem

To solve instances of the *MM-CLP*, we here present a *Variable Neighborhood Search* (*VNS*) heuristic, which exploits an exponential-size neighborhood. Section 3.2 describes in detail the local search procedure applied to visit such a neighborhood.

3.1 The Variable Neighborhood Search

The key constituents of the *VNS* approach are a local search procedure, and a hierarchy of size-increasing neighborhoods used to restart the search every time the procedure reaches a local optimum [26].

Figure 3 reports a pseudocode of our algorithm. The algorithm is initialized with a solution produced by the local search procedure applied to the solution of Algorithm *Greedy1*. At each iteration, the current best known solution x^* is used by the *shaking* procedure to generate a new starting solution, which is then improved by the execution of the local search. The shaking procedure randomly perturbates x^* replacing s columns of the current solution with s unused columns for each mode m. The shaking parameter s starts at a conventional minimum value $s_{\min}$ and varies adaptively, depending on the result of the local search: if the best known solution does not improve, s increases by 1, otherwise it goes back to $s_{\min}$. The rationale of this mechanism is to first generate new starting solutions close to the best known result, so as to intensify the search in a promising region of the solution space. If this restart fails, diversification replaces intensification, and the starting solutions are generated farther and farther away from the current best known one. Every time a new best solution is found, the approach switches back to intensification, and once again generates solutions near the best known one. Of course, the best known solution x^* is kept up-to-date. To avoid improductive excessive diversification, an upper limit $s_{\max}$ is imposed on s: whenever such a limit is reached, s goes back to $s_{\min}$. The algorithm terminates after $R_{\max}$ restarts, or a given total time.

The *shaking* parameter and the local search procedure, i.e., the criterium to select the incumbent solution from the neighborhood, and the definition of the neighborhood itself, affect the computational performance of the *VNS*. A basic local search procedure can be obtained by moving a single facility from a candidate site to another one without changing its mode (local search move). Referring to Formulation (1), the described move corresponds to turning two

Algorithm *VNS_MMSCP*$(I, J, M, a, b, K, w, s_{\min}, s_{\max}, R_{\max})$
$x :=$ *Greedy1*(I, J, M, a, b, K, w);
$x^o :=$ *LocalSearch*(I, J, M, a, b, K, w);
$x^* := x^o$;
$s := s_{\min}$;
for $r := 1$ to $R_{\max}$ **do**
  {Restart the local search}
  $x :=$ *Shaking*$(x^*, s, I, J, M, a, b, K, w)$;
  $x^o :=$ *LocalSearch*(I, J, M, a, b, K, w);
  {Update the shaking parameter and possibly the best known solution}
  if $(f(x^o) > f(x^*))$ **then**
    $s := s_{\min}$;
    $x^* := x^o$;
  else
    $s := s + 1$;
    if $s > s_{\max}$ **then** $s := s_{\min}$;
  end if
end for
return x^*;

Figure 3: Pseudocode of the *VNS* algorithm

decision variables, x_{j_1m} and x_{j_2m}, respectively from one to zero and from zero to one (note that the two variables refer to different columns $j_1 \neq j_2$ and the same mode m). The size of such a neighborhood is polynomial with respect to the size of the problem instance. In what follows, by contrast, we present an exponentially large neighborhood that is obtained by more complex moves.

3.2 Very Large Scale Neighborhood Search

The distinguishing feature of *Very Large Scale Neighborhood Search* (*VLNS*) algorithms is to define a neighborhood $\mathcal{N}$ which is exponentially large with respect to the size of the problem instance, and to explore it more efficiently than with exhaustive enumeration. The approach herein developed to solve the *MM-CLP* problem is a customized version of the cyclic exchanges first proposed in Thompson and Orlin [38]. The neighborhood of this approach is given by cyclic sequences of moves. Although each move might violate the problem constraints, the overall sequence is purposely built to guarantee the feasibility of the solution.

In our setting, a move consists of locating a new facility, removing a facility or changing the mode of a facility. Referring to Formulation (1), the first two types of move correspond to turning a decision variable x_{jm}, respectively from zero to one and from one to zero; the third type corresponds to simultaneously turning x_{jm_1} from zero to one and x_{jm_2} from one to zero (note that the two variables refer to the same column j and different modes $m_1 \neq m_2$). As mentioned above, these moves are combined so that the current solution remains

feasible and the cardinality constraints (1c) and (1d) remain strictly enforced. In particular, since the number of facilities to be located for each mode is given by Constraints (1d), when a facility previously used in mode m_1 changes mode or is deactivated, another facility has to replace it assuming mode m_1.

To better visualize the cyclic exchanges that allow to explore the neighborhood of the current solution, we use a directed graph $G = (N, A)$, where $N = \bigcup_{j \in J} N_j$, which is an adaptation to the *MM-CLP* of the so called *improvement graph* [38]. The subsets N_j are pairwise disjoint; each one includes b_j nodes, one for each mode that can be activated in site j. Each node in subset N_j can have a label indicating a mode currently active in column j. If the active modes are less than b_j, the nodes in excess are left unlabelled. Thus, the number of labelled nodes of N_j is equal to $\sum_{m \in M} x_{jm}$, while the number of unlabelled nodes is equal to $b_j - \sum_{m \in M} x_{jm}$. While the nodes are fixed, their labels change from solution to solution.

As for the arcs of graph G, two nodes $i_h \in N_i$ and $j_k \in N_j$, associated respectively to facility sites i and j (with $i \neq j$), are linked by an arc (i_h, j_k) in the following cases:

1. the two nodes have different labels m_h and m_k (this arc represents a facility in site i changing from mode m_h to m_k);
2. the tail node i_h is unlabelled and the head node has label m_k (this arc represents the location of a facility of mode m_k in site i);
3. the head node j_k is unlabelled and the tail node has label m_h (this arc represents the removal of a facility of mode m_h from site i).

A cyclic exchange corresponds to a directed cycle on the improvement graph as depicted in Figure 4.

Each arc (i_h, j_k) is associated to a weight, equal to the variation of the objective function value induced by the corresponding move. The purpose is to represent a feasible group of moves as a cycle in the auxiliary graph, in such a way that the total weight of the arcs of the cycle is equal to the total effect of the group of moves. However, if two or more moves of a cyclic exchange involve the same mode, the overall variation of the objective function is not always equal to the total weight of the cycle, because the effect of the moves could be nonadditive. To overcome this drawback, we only admit cyclic exchanges which affect any mode at most once.

Also note that any feasible cycle with two or more unlabelled node can be splitted in a number of independent cycles equal to the number of unlabelled nodes. If the overall cycle has positive weight, at least one of the component subcycles also has positive weight, and provides an improving move. This allows the following remark.

Remark 3 *Any cycle has at most one unlabelled node.*

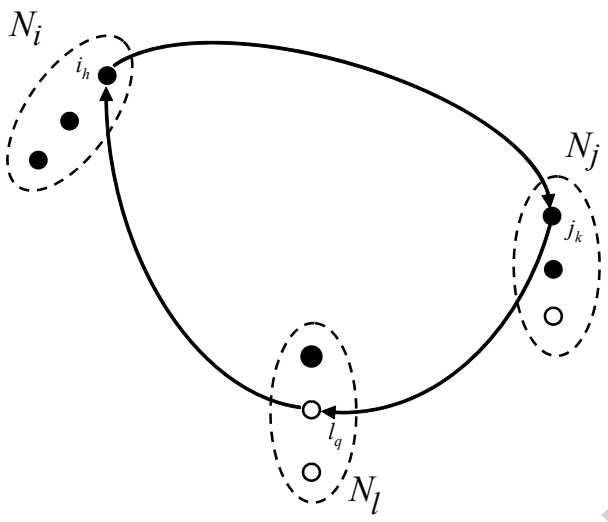

Figure 4: Graphical representation of the basic local search procedure. The picture displays three subsets of nodes, one for each site (i, j and l respectively). Site i hosts three facilities, one of which, denoted as i_h, is of mode m_h.

Move evaluation To compute the best cyclic exchange, i.e., the one of maximum weight, we perform an exhaustive breadth-first generation of all paths of graph G. We first consider all the paths composed of one single arc, then all the paths composed of two arcs and so on. Each path is extended by appending to the end node each outgoing arc, respecting the limitation that all nodes should have different labels and at most one node should be unlabelled. For each path generated, the algorithm evaluates the cost of the cycle obtained by going back from the end of the path to its starting node. This algorithm has computational complexity in $O\left(n^{l_{\max}}\right)$, where n is the number of nodes of the improvement graph and $l_{\max}$ the maximum number of nodes in a cycle. The limitation on the labels of the nodes imposes a bound on the number of nodes in the feasible cycles.

Remark 4 *The maximum number of nodes in any cycle is bounded above by* $|M|+1$, *i.e.,* $l_{\max} \leq |M|+1$.

The efficiency of the neighborhood exploration is strongly improved by the following remark, proved in [1].

Remark 5 *Any cycle of positive total weight has at least one starting node such that all the subpaths along the cycle which originate from that node have positive weights.*

This implies that all nonpositive weight paths generated during the search can be immediately removed. Indeed, if such a path belongs to a positive weight

cycle, there is an alternative path of positive weight which allows to identify the cycle.

4 Computational Experience

In this section we present the computational results of the proposed algorithm, which has been coded in C++, compiled by `gcc 4.4.3` and run on a PC equipped with an Intel Core2 Quad-core 2.66 GHz and 4 GB of RAM.

4.1 Benchmark instances

We have considered two benchmark sets of random instances, denoted as *square* and *rectangular*, respectively. The square instances represent the situation in which a facility can be located in each customer point. The number of columns $|J|$ is therefore equal to the number of rows $|I|$, and in our benchmark both are set to 1 000. The rectangular instances model the situation in which the distribution of the candidate facility sites is denser than that of the demand centers. The number of columns per mode is therefore larger than the number of rows, and more specifically they are set to 500 rows and 5 000 columns.

In addition to the number of rows and columns, each instance is characterized by the following parameters: the number of modes ($|M|$), the maximum number of modes for each column (b_j), the number of columns used in each mode (K_m), the range of the weights (w), the covering pattern of each column. As for the number of modes, we generated instances with $|M| = 2$ modes and instances with $|M| = 3$ modes. The former have $b_j = 1$ for all $j \in J$, whereas for the latter we generated tighter instances with $b_j = 1$, and looser ones with $b_j = 2$. The number of columns for each mode, K_m, is extracted at random, with a uniform distribution in $[0.02\,|J|; 0.03\,|J|]$ for the square instances, in $[0.08\,|J|; 0.10\,|J|]$ for the rectangular ones. These values were chosen so as to avoid both the trivial instances in which the optimal solution covers all rows and the easy ones in which the columns are so few that a greedy choice directly provides the optimal result. The *unweighted* instances have all weights equal to one, while the *weighted* instances have random integer weights uniformly distributed in $\{1, \ldots, 10\}$. Finally, the covering pattern can be *uniform*, i. e. each column covers the same rows in all modes, or *assorted*, i. e. the covered rows are selected independently for each mode. For each combination of the parameters listed above, we have generated a pool of 5 instances: the set of rows covered by each column is generated at random with a uniform distribution, imposing a 5% density[1]. In order to avoid easy instances, we imposed that each column should cover at least one row in each mode and that each row should be covered by at least two columns in each mode.

Overall, we generated 60 instances for each of the two benchmark sets.

[1] The density is the average percentage of rows covered by each column.

4.2 Parameter tuning

A first computational campaign has been devoted to tuning the parameter s that controls the shaking mechanism of the *VNS* procedure, the criterium to select the incumbent solution from the neighborhood, and the size of the neighborhood itself.

First, remark that the number of columns K_m used in each mode $m \in M$ is given by the problem instance. Since the shaking procedure replaces s of these columns with s unused ones in each mode, necessarily $s \leq \min(K_m, |J| - K_m)$. In our experiments we always set $s \leq K_{\min} = \min_{m \in M} K_m$ because in our benchmarks $K_m < |J| - K_m$ for all $m \in M$. We considered three different ranges for parameter s:

- $[s_{\min}; s_{\max}] = [5; K_{\min}]$;
- $[s_{\min}; s_{\max}] = [5; K_{\min}/2]$;
- $[s_{\min}; s_{\max}] = [K_{\min}/2; K_{\min}]$

The first tuning leaves the parameter completely free to evolve depending on the results of the search. The second one avoids the larger values, thus reducing the diversification effect of shaking. The third one, by contrast, stresses the effect of diversification. We did not consider values of s smaller than 5 because preliminary experiments showed that, most of the time, a search starting very close to the best known solution ends up again in it, with a useless waste of computational time.

As for the exploration of the neighborhood, we considered two common variants: the *global-best* strategy explores the whole neighborhood, returning the best solution found; the *first-best* strategy terminates the search as soon as an improving solution is found. In a single iteration, of course, the first strategy yields a stronger improvement, but the latter allows to perform more iterations in the same time. Experimental evidence suggests that in different problems either strategy can perform better [9].

	Global-best			First-best		
s	Gap BK	i_{tot}	i^*	Gap BK	i_{tot}	i^*
$[5; K_{\min}]$	0.25%	607.93	162.98	0.40%	872.08	166.70
$\left[5; \frac{K_{\min}}{2}\right]$	0.19%	609.03	270.87	0.40%	871.83	166.70
$\left[\frac{K_{\min}}{2}; K_{\min}\right]$	0.56%	603.10	64.13	0.59%	871.18	12.27

Table 1: Computational results of the parameter tuning phase on the square instances

For this campaign, we allotted a computational time of 600 seconds for each instance. For each of the three ranges of the shaking parameter s and for each

s	Global-best			First-best		
	Gap BK	i_{tot}	i^*	Gap BK	i_{tot}	i^*
$[5; K_{\min}]$	0.09%	370.28	192.05	0.15%	425.87	169.88
$\left[5; \frac{K_{\min}}{2}\right]$	0.09%	370.37	194.70	0.15%	426.07	169.88
$\left[\frac{K_{\min}}{2}; K_{\min}\right]$	0.22%	356.38	3.02	0.22%	427.70	6.40

Table 2: Computational results of the parameter tuning phase on the rectangular instances

of the two neighborhood exploration strategies, Table 1 reports the average values computed on all the square instances of the following statisics: column "Gap BK" provides the average percentage gap $(z^* - z)/z$ between the result z obtained by the current parameter setting and the best result z^* found in the overall experimental campaign. This is assumed as a reference because the optimal value for these problems is unknown and, as we shall discuss in the following, even a state-of-the-art general purpose solver such as CPLEX 12 is unable to provide meaningful upper bounds on the optimum. Statistics i_{tot} and i^* are the number of local search iterations performed in 600 seconds and the iteration at which the result z has been found, respectively. Table 2 reports the corresponding information for the rectangular instances. The two tables suggest a clear dominance of the global-best strategy on the first-best strategy: the latter, in fact, has worse results, on average; it finds them earlier, but is unable to improve them, even though it performs more iterations in the same time. The difference between the average performances is confirmed by the application of *Wilcoxon's matched-pairs signed-ranks test* [40] to the instance-by-instance results. This test estimates that the probability to obtain such results with a random fluctuation is $p = 1.585 \cdot 10^{-14}$. As for the shaking parameter, the setting which favors diversification over intensification by adopting large values of s is outperformed by the other two settings ($p < 10^{-26}$ in both cases). The intensifying setting, which restricts s in $[5; K_{\max}/2]$, proves the better of the remaining two, with a probability of random fluctuations equal to $p = 1.221 \cdot 10^{-4}$. The intensifying setting has a particularly strong dominance on the square instances, whereas for several rectangular instances the two settings are equivalent. This is partly due to the fact that the two settings behave exactly in the same way as long as the shaking parameter s has not yet reached the threshold value $s_{\max}$. Since in the allotted time the algorithm performs less iterations on the rectangular instances, the difference between these two settings is less marked on that benchmark.

Polynomial *vs* exponential sized neighborhood

The parameter tuning computational campaign described above has been carried out on the *VNS* implementing the exponentially large neighborhood (*VLNS*). Indeed, experiments on the polynomial-size neighborhood confirm the performance with respect to the shaking parameter and the exploration strategy: the intensifying setting and the global-best exploration strategy outperform the alternative settings. The results are, however, worse than those obtained, in the same time, by the *VLNS* procedure. In particular, the average gap with respect to the best known result is 0.24% for the square instances and 0.12% for the rectangular ones, versus 0.19% and 0.09% (see Tables 1 and 2). Considering instance-by-instance solutions, Wilcoxon's test estimates the probability to obtain such a difference with a random fluctuation to be $p = 1.011 \cdot 10^{-12}$ when considering all six parameter settings, and $p = 4.058 \cdot 10^{-12}$ when considering only the best one. Using a good parameter setting probably allows to find optimal or nearly optimal solutions even exploring a smaller neighborhood.

An interesting remark is that, when applying the first-best exploration strategy, the difference between the polynomial and exponential size neighborhoods is less marked (0.44% and 0.16% versus 0.40% and 0.15% for square and rectangular instances, respectively). This is easily explained by the fact that the first-best strategy terminates the exploration of the neighborhood as soon as an improving solution has been found and that the moves adopted in the polynomial-size neighborhood correspond, in the improvement graph, to cycles of two arcs visiting an unlabelled node. These cycles are among the first explored in the exponential-size neighborhood. Therefore, the *VLNS* procedure with the first-best exploration strategy has a good probability to return a solution included also in the polynomial-size neighborhood. Anyway, this remark does not involve the best performing parameter setting. Our conclusion is that, even if the polynomial-size neighborhood can be explored more quickly, the exponential-size one provides solutions of better quality and a stronger robustness with respect to local optima. On the basis of such results, in the following experiments we have adopted the intensifying setting for the shaking parameter, the global-best exploration strategy and the exponential-size neighborhood.

4.3 Performance of the different phases of the algorithm

An important feature of the *MM-LCP* is the fact that nearly optimal solutions can be found in a very short time even by the greedy initialization procedure. This was remarked also in the early works on the *MLCP* and is related to the approximation properties discussed in Section 2.1. We have therefore compared the result obtained by the greedy initialization, the one improved by the basic local search procedure with a global-best strategy, the result further improved by the *VNS* algorithm with the best parameter setting and the one obtained independently by CPLEX 12.0. The greedy and the local search procedures run to completion: the former takes on average 0.1 seconds, the latter about 10 seconds. The *VNS* algorithm is terminated after 600 seconds, which also

include the greedy and local search phases. Finally, CPLEX runs for 1 hour. Tables 3 and 4 refer, respectively, to the square and rectangular instances. The first column (Instance) provides a description of each subclass of instances: $|M|$ is the number of modes, b the capacity of each location – that, we recall, has the same value for all $j \in J$ –; w distinguishes the weighted instances (W) from the unweighted one (U); A/U distinguishes the assorted mode instances, denoted with A, from the uniform mode instances, denoted with U. The following three columns report the percentage gap between the result obtained by each phase of the algorithm and the best known one. Finally, the last two columns provide the percentage gap between the heuristic solution returned by CPLEX and the best known one, and the percentage gap between the upper bound computed by CPLEX and the best known result ($(UB - z^*)/z^*$). The value in each row is the average over the 5 instances of each group.

Instance				Greedy	Local Search	VNS	Cplex	
$\|M\|$	b	w	A/U	(%)	(%)	(%)	Best Sol. (%)	UB (%)
2	1	U	A	0.85	0.75	0.12	15.21	15.68
2	1	U	U	0.78	0.65	0.10	14.82	11.08
2	1	W	A	1.03	0.81	0.15	16.03	13.38
2	1	W	U	0.85	0.66	0.20	17.48	8.85
3	1	U	A	0.85	0.68	0.24	19.38	14.58
3	1	U	U	0.63	0.44	0.07	15.23	12.48
3	1	W	A	0.99	0.73	0.12	14.82	12.55
3	1	W	U	0.66	0.59	0.10	12.93	10.40
3	2	U	A	0.51	0.37	0.16	19.51	14.68
3	2	U	U	0.69	0.45	0.45	17.57	12.14
3	2	W	A	0.68	0.50	0.28	21.03	12.74
3	2	W	U	0.71	0.40	0.30	17.68	9.93

Table 3: Average GAP (with respect to the best known solution) computed on each group of square instances.

The greedy algorithm finds in a fraction of a second a solution very close to the best known one (on average 0.55% worse). The local search procedure improves it in few seconds, reducing the average gap to 0.41%. Finally, the *VNS* algorithm further improves the solution reducing the gap to 0.14% on average. By contrast, CPLEX after one hour returns a heuristic solution which is on average 11% worse than the best known one. This suggests that an *ad hoc* heuristic for the *MM-LCP* is indeed useful, as it cannot be easily replaced by a general-purpose solver. This is in sharp contrast with the common experience on the *MCLP*, for which general-purpose solvers are very effective.

Instance				Greedy	Local Search	VNS	Cplex	
$\|M\|$	b	w	A/U	(%)	(%)	(%)	Best Sol. (%)	UB (%)
2	1	U	A	0.34	0.21	0.03	5.17	4.14
2	1	U	U	0.38	0.27	0.03	5.22	4.45
2	1	W	A	0.43	0.34	0.07	5.67	3.60
2	1	W	U	0.37	0.27	0.10	5.74	3.90
3	1	U	A	0.42	0.32	0.11	5.79	4.99
3	1	U	U	0.33	0.21	0.05	5.36	5.05
3	1	W	A	0.29	0.17	0.02	6.17	4.43
3	1	W	U	0.30	0.21	0.06	5.99	4.49
3	2	U	A	0.30	0.17	0.12	5.62	5.01
3	2	U	U	0.31	0.21	0.20	5.59	4.87
3	2	W	A	0.34	0.21	0.16	6.07	4.36
3	2	W	U	0.24	0.14	0.12	6.08	4.36

Table 4: Average GAP (with respect to the best known solution) computed on each group of rectangular instances.

The hardness of the *MM-LCP* for *MIP* solvers is confirmed by the scarce quality of the upper bound provided. In fact, the gap with respect to the upper bound is on average 8%. Indeed, this bound is often equal to the trivial combinatorial bound given by the sum of the weights of all rows in all modes, which can be computed in negligible time. This happens for all rectangular instances and for more than half of the square instances.

4.4 Comparison with heuristic concentration

To evaluate the performances of the proposed *VNS* approach, we have also implemented an alternative heuristic, based on the *Heuristic Concentration* approach, which has been applied to the *MCLP* in [36] and, to the best of our knowledge, is the most effective heuristic to solve large instances of this problem.

This is a two-stage metaheuristic. In the first stage, q random starting solutions are improved by a basic local search heuristic, and all the x_{jm} variables which are set to 1 in the best $t \leq q$ local optimal solutions are collected in a list, called the *concentration set* (*CS*). In the second stage, Formulation (1) is solved with CPLEX 12.0, under the additional constraint that all variables not belonging to the *CS* are set to zero. The *CS*, in fact, is considered likely to contain the optimal solution, provided that the basic heuristic is good, and the number of restarts q and of best solutions t are large enough. On the other hand, excessively large values of q increase the computational time of the first stage and excessively large values of t increase the computational time of the second stage.

In our experiments, we set $q = 30$ and $t = 10$, as in [36]. Tables 5 and 6

compare the *VNS* approach with the best parameter setting, the first stage of the *HC* algorithm (column *HC-1*), and the final result returned by the second stage of the *HC* algorithm (column *HC-2*). The former table refers to the square instances, the latter to the rectangular ones. The first column of each table identifies each subclass of instances, and each row reports the average results over the 5 instances of the subclass. The second column provides the gap with respect to the best known result achieved by the *VNS* approach with the best parameter setting as identified above. The following two columns provide the corresponding gap and the computational time in seconds for the $q = 30$ random restarts of the first stage of *HC* . The last two columns provide the number of binary variables x_{jm} which compose the *CS* and the percentage gap for the result returned by the second stage of the *HC* algorithm with a time limit of 1 800 seconds.

Instance				*VNS*	*HC-1*		*HC-2*	
$\|M\|$	b	w	A/U	Gap BK	Gap BK	CPU	$\|CS\|$	Gap BK
2	1	U	A	0.12%	1.14%	653.44	305.40	3.92%
2	1	U	U	0.10%	1.24%	729.84	379.40	5.89%
2	1	W	A	0.15%	1.47%	698.81	311.20	4.29%
2	1	W	U	0.20%	1.43%	786.55	371.40	5.26%
3	1	U	A	0.24%	1.17%	1719.74	485.00	6.59%
3	1	U	U	0.07%	1.39%	1925.27	560.00	7.06%
3	1	W	A	0.12%	1.23%	1836.91	488.60	6.13%
3	1	W	U	0.10%	1.16%	2033.36	552.80	7.25%
3	2	U	A	0.16%	1.06%	3780.03	483.60	6.01%
3	2	U	U	0.45%	1.64%	4091.90	568.80	7.59%
3	2	W	A	0.28%	1.09%	4075.43	486.60	6.48%
3	2	W	U	0.29%	1.49%	4401.99	549.20	7.35%

Table 5: Computational results of the *HC* algorithm on the square instances.

The results show that, contrary to what happens for the basic *MLCP*, the *HC* approach is not very effective on the *MM-LCP*. The reason derives from both stages of the *HC* algorithm. The random restart strategy used in the first stage is much less effective than the shaking mechanism of *VNS*, because it produces solutions of bad quality, thus requiring to the local search procedure a long computational time before retrieving a local optimum. Moreover, the second stage of the *HC* algorithm is neither efficient nor effective. In fact, within the imposed time limit, CPLEX proves unable to recombine the $t = 10$ best solutions composing the *CS* into a better one, and actually also unable to extract the best solution included in the *CS*. So, while it is true that the

Instance				*VNS*	*HC-1*		*HC-2*	
$\|M\|$	b	w	A/U	Gap BK	Gap BK	CPU	$\|CS\|$	Gap BK
2	1	U	A	0.03%	0.58%	1559.04	544.80	4.07%
2	1	U	U	0.03%	0.45%	1552.26	543.80	4.15%
2	1	W	A	0.07%	0.58%	1651.40	549.40	4.44%
2	1	W	U	0.10%	0.48%	1657.52	545.00	4.41%
3	1	U	A	0.11%	0.64%	3351.59	790.00	4.50%
3	1	U	U	0.05%	0.53%	3434.01	827.40	4.42%
3	1	W	A	0.02%	0.50%	3573.03	777.80	4.71%
3	1	W	U	0.06%	0.44%	3603.25	821.60	4.86%
3	2	U	A	0.11%	0.68%	7483.37	795.60	4.55%
3	2	U	U	0.20%	0.67%	7713.76	841.60	4.71%
3	2	W	A	0.16%	0.53%	8155.12	775.20	4.89%
3	2	W	U	0.12%	0.60%	8094.79	817.20	4.72%

Table 6: Computational results of the *HC* algorithm on the rectangular instances.

percentage gap obtained by *HC* is much better than that obtained by CPLEX on the whole problem (see Tables 3 and 4), the experiments do not support the effectiveness of *HC* on the *MM-LCP*.

It could be objected that the bad performance of *HC* could be due to an excessive value of the two crucial parameters q and t. This idea, however, is contradicted by the number of variables which compose the *MIP* problem used in the second stage of the *HC* algorithm. As it can be seen from column $|CS|$ in Tables 5 and 6, the number of binary variables is strongly reduced by the heuristic selection performed in the first phase. For the square instances, in fact, the binary variables x decrease from $2\,000 - 3\,000$ (depending on the number of modes) to around $300 - 500$, while the number of variables which must be set to 1 is $\sum_{m \in M} K_m \in [50; 75]$. For the rectangular instances, the binary variables x decrease from $10\,000 - 15\,000$ to around $500 - 800$ and $\sum_{m \in M} K_m \in [100; 150]$.

5 Conclusion

In this paper, we study the *Multimode Covering Location Problem* (*MM-CLP*), which is a generalization of the Maximal Covering Location Problem (*MCLP*). This problem consists of locating a given number of facilities of different types to serve demand centers with the restriction that only a limited number of different types can be activated in each candidate facility site.

The problem is intrinsically difficult to solve and much more challenging than the *MCLP*. In fact, commercial solvers fail to compute good quality solutions in a reasonable amount of time. The hardness of the *MM-LCP* for *MIP* solvers

is confirmed by the poor quality of the upper bound that they are able to compute. Indeed, for most of our benchmark instances, this bound is often equal to the trivial combinatorial bound given by the sum of the weights of all rows in all modes. Nevertheless, the problem admits a constant factor approximation guarantee that we proved by means of two greedy algorithms extending in a nontrivial way a similar property of the *MCLP*.

To improve the greedy solutions, we have developed a *VNS* approach, which implements a *VLSNS* algorithm as its basic local search procedure. The proposed procedure is able to compute good quality solutions in short computational times. The viability of the proposed approach is also corroborated by a comparison with an alternative *VNS* based on a polynomial-size neighborhood and a *Heuristic Concentration* algorithm, which is, to the best of our knowledge, the most effective approach to solve large instances of the *MCLP*.

References

[1] R.K. Ahuja, J.B. Orlin, D. Sharma (2003) A composite very large-scale neighborhoodstructure for the capacitated minimum spanning tree problem *Operations Research Letters*, 31, pp. 185–194

[2] S. Arora, N. Gupta, S. Khuller, Y. Sabharwal, S. Singhal (2014) Facility location with redblue demands *Operations Research Letters*, 42, pp. 462–465

[3] D. Beltrán-Cano, B. Melián-Batista, J. M. Moreno-Vega (2009) Solving the Rectangle Packing Problem by an iterative hybrid heuristic *Lecture Notes in Computer Science*, Volume 5717, pp. 673–680

[4] P. Berman, B. DasGupta, E. Sontag (2007) Randomized approximation algorithms for set multicover problems with applications to reverse engineering of protein and gene network. *Discrete Applied Mathematics*, 155 (6-7), pp. 733–749.

[5] O. Berman, Z. Drezner, D. Krass (2010) Generalized coverage: New developments in covering location models. *Computers & Operations Research* 37, pp. 1675–1687

[6] K. Bicakci, I. E. Bagci, B. Tavli, Z. Pala (2013) Neighbor Sensor Networks: Increasing Lifetime and Eliminating Partitioning Through Cooperation. *Computer Standards & Interfaces*, 35 (4), pp 396–402.

[7] C.H. Chung, Recent applications of the Maximal Covering Location Problem (MCLP) model, *Journal of the Operational Research Society* 37 (1986) 735-746.

[8] R.L. Church, C.S. ReVelle (1974) The maximal covering location problem. *Papers of the Regional Science Association* 32, pp. 101–118.

[9] F. Colombo, R. Cordone, G. Lulli (2013) A Variable Neighborhood Search Algorithm for the Multimode Set Covering Problem. *Journal of Global Optimization*, [in press: DOI 10.1007/s10898-013-0094-6]

[10] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein (2001) *Introduction to algorithms*. MIT Press.

[11] R. Aringhieri, R. Cordone, (2011) Comparing local search metaheuristics for the Maximum Diversity Problem, *Journal of the Operational Research Society*, 62 (2), pp. 266–280

[12] R. Cordone, G. Lulli (2013) A GRASP metaheuristic for microarray data analysis *Computers & Operations Research*, 40 (12), pp. 3108–3120

[13] J. Current, M. O'Kelly (1992) Locating emergency warning sirens, *Decision Sciences* 23, pp. 221–234.

[14] M.S. Daskin, P.C. Jones, T.J. Lowe (1990) Rationalizing tool selection in a flexible manufacturing system for sheet metal products, *Operations Research* 38, pp.1104–1115.

[15] M.S. Daskin (1995) Network and discrete location: Models, algorithms, and applications. J. Wiley and Sons, Inc., New York.

[16] P. Dell'Olmo, N. Ricciardi, A. Sgalambro (2014) Multiperiod Maximal Covering Location Model for the optimal location of intersection safety cameras on an urban traffic network. *Working Paper*.

[17] B.T. Downs, J.D. Camm (1996) An exact algorithm for the maximal covering location problem, *Naval Research Logistics Quarterly* 43, pp. 435–461.

[18] F.R. Dwyer, J.R. Evans (1981) A branch and bound algorithm for the list selection problem in direct mail advertising, *Management Science* 27, pp. 658–667.

[19] D. Eaton, M. Hector, V. Sanchez, R. Latingua, J. Morgan (1986) Determining ambulance deployment in Santo Domingo, Dominican Republic *Journal of the Operational Research Society*, 37, pp. 113–126.

[20] H.A. Eiselt, C.L. Sandblom (2004) Decision Analysis, Location Models, and Scheduling Problems. Springer-Verlag, BerlinHeidelbergNew York.

[21] T. A. Feo, M. G. C. Resende (1989) A probabilistic heuristic for a computationally difficult set covering problem *Operations Research Letters*, 8, pp. 67–71.

[22] R. D. Galvao, L. G. Acosta Espejo, B. Boffey (2000) A comparison of Lagrangean and surrogate relaxations for the maximal covering location problem. *European Journal of Operational Research* 124, pp. 377–389.

[23] R. D. Galvao, C. ReVelle (1996) A Lagrangean heuristic for the maximal covering location problem. *European Journal of Operational Research* 88, pp. 114–123.

[24] M. R. Garey, D. S. Johnson (1979) *Computers and Intractability: A Guide to the Theory of NP-completeness.* W. H. Freeman and Company, New York.

[25] G. Ghiani, F. Guerriero, R. Musmanno (2002) The capacitated plant location problem with multiple facilities in the same site. *Computers & Operations Research* 29, pp. 1903–1912

[26] P. Hansen, N. Mladenoviç, J. A. Moreno Pérez (2010) Variable Neighbourhood Search: Methods and Applications. *Annals of Operations Research*, 175, pp. 367–407.

[27] M. I. Hosny, C. L. Mumford (2010) Solving the One-commodity Pickup and Delivery Problem using an adaptive hybrid VNS/SA approach. *Lecture Notes in Computer Science*, Volume 6239, pp. 189–198

[28] E.S. Hougland, N.T. Stephens (1976) Air pollutant monitor siting by analytical techniques, *Journal of the Air Pollution Control Association* 26, pp. 52–53.

[29] L.A.N. Lorena, M.A. Pereira (2002) A lagrangean/surrogate heuristic for the maximal covering location problem using Hillsman's edition. *International Journal of Industrial Engineering*, 9 (1), pp. 57–67.

[30] M.T. Melo, S. Nickel, F. Saldanha-da-Gama (2009) Facility location and supply chain management – A review, *European Journal of Operational Research* 196, pp. 401–412

[31] P. B. Mirchandani, R.L. Francis (Eds.) (1990) *Discrete location theory.* Wiley-Interscience.

[32] J.T. Pastor (1994) Bicriterion programs and managerial locations: Application to the banking sector, *Journal of the Operational Research Society* 45, pp. 1351–1362.

[33] M.A. Pereira, L.A.N. Lorena, E.L.F. Senne (2007) A column generation approach for the maximal covering location problem. *Intl. Transaction in Operations Research* 14, pp. 349–364

[34] H.K. Rajagopalan, C. Saydam, L. Xiao (2008) A multiperiod set covering location model for dynamic redeployment of ambulances. *Computers & Operations Research*, 35, pp. 814–826.

[35] C.S. ReVelle, H.A. Eiselt, M.S. Daskin (2008) A bibliography for some fundamental problem categories in discrete location science. *European Journal of Operational Research* 184, pp. 817–848

[36] C. ReVelle, M. Scholssberg, J. Williams (2008) Solving the maximal covering location problem with heuristic concentration. *Computers & Operations Research* 35, pp. 427–435

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

[38] P.M. Thompson, J.B. Orlin (1989) The theory of cyclic transfers, *Working Paper OR200-89*, Operations Research Center, MIT, Cambridge, MA.

[39] R.V. Vohra, N. Hall (1993) A probabilistic analysis of the maximal covering location problem *Discrete Applied Mathematics*, 43, pp. 175–183.

[40] F. Wilcoxon (1945) Individual Comparisons by Ranking Methods. *Biometrics*, 1, pp. 80–83.