Author's Accepted Manuscript

www.elsevier.com/locate/caor

A Hybrid Method for the Probabilistic Maximal Covering Location-Allocation Problem

Marcos A. Pereira, Leandro C. Coelho, Luiz A.N. Lorena, Ligia C. de Souza

PII: S0305-0548(14)00322-0
DOI: http://dx.doi.org/10.1016/j.cor.2014.12.001
Reference: CAOR3697

To appear in: *Computers & Operations Research*

A Hybrid Method for the Probabilistic Maximal Covering Location-Allocation Problem

Marcos A. Pereira[a,*], Leandro C. Coelho[b,c], Luiz A. N. Lorena[d], Ligia C. de Souza[d]

[a]*São Paulo State University, Av. Dr. Ariberto Pereira da Cunha, 333, Guaratinguetá, Brazil, 12516-410*

[b]*CIRRELT – Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation*

[c]*Faculté des sciences de l'administration, Université Laval, 2325 de la Terrasse, Quebec City, Canada, G1K 0A6*

[d]*Instituto Nacional de Pesquisas Espaciais, Av. dos Astronautas, 1758, São José dos Campos, Brazil, 12221-010*

Abstract

This paper presents a hybrid algorithm that combines a metaheuristic and an exact method to solve the Probabilistic Maximal Covering Location-Allocation Problem. A linear programming formulation for the problem presents variables that can be partitioned into location and allocation decisions. This model is solved to optimality for small and medium-size instances. To tackle larger instances, a flexible adaptive large neighborhood search heuristic was developed to obtain location solutions, whereas the allocation subproblems are solved to optimality. An improvement procedure

*Corresponding author. Phone: +55 12 3123-2897; Fax: +55 12 3123-2845.

Email addresses: mapereira@feg.unesp.br (Marcos A. Pereira), leandro.coelho@cirrelt.ca (Leandro C. Coelho), lorena@lac.inpe.br (Luiz A. N. Lorena), li.correasouza@gmail.com (Ligia C. de Souza)

Preprint submitted to Computers & Operations Research *December 9, 2014*

based on an integer programming method is also applied. Extensive computational experiments on benchmark instances from the literature confirm the efficiency of the proposed method. The exact approach found new best solutions for 19 instances, proving the optimality for 18 of them. The hybrid method performed consistently, finding the best known solutions for 94.5% of the instances and 17 new best solutions (15 of them optimal) for a larger dataset in one third of the time of a state-of-the-art solver.
Keywords: Facility location, congested systems, hybrid algorithm, adaptive large neighborhood search, exact method, queueing maximal covering location-allocation model, PMCLAP.

1. Introduction

Facility location plays an important role in logistic decisions. Each day, many enterprises resort to quantitative methods to estimate the best or the more economical way to meet clients' demand for goods or services. In some cases, the availability of the service can be associated with a time or distance to an existing facility. In this case, the decision maker has to find the best location to open the facilities in order to satisfy most of the demand.

The Maximal Covering Location Problem (MCLP) is a facility location problem which aims to select some location candidates to install facilities, in order to maximize the total demand of clients that are located within a covering distance from an existing facility [6]. Examples of this problem appear, for instance, in the public sector to determine the location of emergency services such as fire stations, ambulances, etc. Private companies solve this problem to locate ATMs or vending machines, branches of a bank, stores of fast food chains, cellular telephony antennae, etc. A more extensive list of applications can be found in [13], [15] and [27]. Variations of this problem, including negative weights and travel time uncertainty, can be found in [4] and [5]. Solution approaches to the MCLP include greedy heuristics [6, 12], linear programming relaxation [6], lagrangean relaxation [14], genetic algorithm [1], lagrangean/surrogate heuristic [19] and column generation [23].

In some applications, the number of clients and their associated demand allocated to a facility may have an impact on the behavior of the system. Depending on the nature of the service, clients may have to wait to be served. In such congested systems, the service quality is measured not only by the proximity to an open facility, but also by a service level, such as the number of clients in a queue or the waiting times. Under the assumption that the service requests are constant over time, this aspect can be modeled by simply adding capacity constraints to the MCLP model. However, this deterministic approach may yield idle or overloaded servers.

To deal with more realistic scenarios, Marianov and Serra [21] proposed a model considering the arrival of clients to a facility as a stochastic process depending on the clients' demands. The authors define a minimum limit on the quality of service based on the number of clients in the queue or on the maximum waiting time. This yields an extension of the MCLP called Queueing Maximal Covering Location-Allocation Model (QM-CLAM) or Probabilistic Maximal Covering Location-Allocation Problem (PMCLAP). Due to the complexity of the problem and to the size of real world instances, most research is devoted to the development of heuristic methods [10, 11].

Location-allocation models contain, at least, two types of decision variables: where to install a facility (location decision) and which clients to serve by the opened facility (allocation decision). Traditional solution approaches, such as branch-and-bound algorithms, deal with location and allocation decisions simultaneously: clients are allocated to a candidate location that is still being considered for the installation of a facility. However, a hierarchical approach appears as a natural heuristic algorithm: locate facilities first and, in a second step, allocate clients to them.

To explore this characteristic of the problem, a new iterative hybrid method is developed. The location part of the problem is dealt with by an Adaptive Large Neighborhood Search (ALNS) metaheuristic, and the corresponding allocation solution is obtained by solving an integer subproblem to optimality. The ALNS was proposed by Ropke and Pisinger [25] for a class of the vehicle routing problem, and has since then been employed for a myriad of other problems, including the vehicle scheduling problem [3, 22], the fixed charged network flow problem [17], the stochastic arc routing problem [18], several classes of vehicle routing problems [26], the inventory-routing problem [7, 8], and train timetabling [2]. We are aware of only one application of the ALNS to a problem with a facility location component [16], which is significantly different from the problem at hand.

The remainder of this paper is organized as follows. Section 2 provides a formal description and a mathematical formulation for the PMCLAP. The hybrid ALNS metaheuristic is described in Section 3. The results of extensive computational results on benchmark instances are presented in Section 4. Conclusions follow in Section 5.

2. Problem description and mathematical formulation

The MCLP, introduced by Church and ReVelle [6], aims at locating p facilities among n possible candidates, such that the maximal possible population is served. However, this problem does not consider capacities or congestion issues. To overcome this limitation, Marianov and Serra [21] introduced the PMCLAP, an extension of the MCLP in which a minimum quality on the service level is imposed, assuming that clients arrive to the facilities according to a Poisson distribution. The way to measure the demand at a facility is either counting the number of people waiting for the service, or by measuring the waiting time for the service.

Formally, the problem is defined on a graph with a set $\mathcal{N}$ of n nodes. Each node is associated with a demand d_i and a service radius (or covering distance) S_i in case a facility is located at the facility candidate i. Without loss of generality, a service radius equal to S is considered for all facilities. Let $\mathcal{N}_i$ be a subset containing the list of nodes within S units of distance from node i, i.e., the set of location candidates j that can serve client i. In the PMCLAP, f_i represents the contribution of client i to the system congestion. This value is calculated as a fraction of the client's demand. It is assumed that clients will arrive to a facility according to a Poisson distribution with parameter rate μ. The parameter α defines the minimum probability of, at most:

- a queue with b clients, or;
- a waiting time of τ minutes.

To model the PMCLAP, we define two sets of binary variables: one for location and another for allocation decisions. Variables y_j are equal to one if and only if location $j \in \mathcal{N}$ is opened, and variables x_{ij} are equal to one if and only if the demand of node i is associated with facility j, $i, j \in \mathcal{N}$. The problem can be formulated as follows:

$$\text{maximize} \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}_i} d_i x_{ij} \tag{1}$$

subject to

$$\sum_{j \in \mathcal{N}_i} x_{ij} \leq 1 \quad i \in \mathcal{N} \tag{2}$$

$$\sum_{j \in \mathcal{N}} y_j = p \tag{3}$$

$$x_{ij} \leq y_j \quad i \in \mathcal{N} \quad j \in \mathcal{N} \tag{4}$$

$$\sum_{i \in \mathcal{N}} f_i x_{ij} \leq \mu \sqrt[b+2]{1-\alpha} \quad j \in \mathcal{N} \tag{5}$$

or

$$\sum_{i \in \mathcal{N}} f_i x_{ij} \leq \mu + \frac{1}{\tau} \ln\left(1-\alpha\right) \quad j \in \mathcal{N} \tag{6}$$

$$y_j, x_{ij} \in \{0,1\} \quad i \in \mathcal{N} \quad j \in \mathcal{N}. \tag{7}$$

The objective function (1) maximizes the total served demand. Constraints (2) guarantee that each client is served by at most one facility. Constraint (3) defines the number of facilities to be opened. Constraints (4) link location to allocation variables, only allowing clients to be allocated to an opened facility. Constraints (5) ensure that facility j has less than b clients in the queue with at least probability α, and constraints (6) ensure that the waiting time for service at facility j is at most τ minutes with probability of at least α. Constraints (7) define the binary nature of the variables. Obviously, x_{ij} equals zero for all $j \notin \mathcal{N}_i$.

Constraints (5) and (6) are related to the probabilistic nature of the problem. For these constraints, Marianov and Serra [21] assume an $M/M/1/\infty/FIFO$ queueing system with service requests occurring at each demand node i according to a Poisson process with intensity f_i. As customers arrive at a facility j from different demand nodes, the request for service at this facility is the sum of several Poisson processes with intensity λ_j calculated as:

$$\lambda_j = \sum_{i \in \mathcal{N}} f_i x_{ij} \tag{8}$$

According to equation (8), if variable x_{ij} is one, then client i is allocated to facility j and the corresponding intensity will be included in the calculation of λ_j. In order to maintain the equilibrium of the system, an exponentially distributed service time with average rate μ, where $\mu > \lambda_j$, is assumed for all the facilities.

The PMCLAP is NP-hard [24], being harder to solve by exact methods in reasonable computation times as the size of the instance increases. We evaluate the performance of a state-of-the-art solver in Section 4.

3. Hybrid adaptive large neighborhood search algorithm

Considering the location and allocation decisions of the problem separately, an iterative hybrid method was developed. At each iteration, the location solution values are obtained by a powerful and flexible ALNS heuristic, in which an integer subproblem was embedded. This subproblem is then solved exactly by mathematical programming at each iteration, in order to determine the optimal allocation solution values and the solution cost.

Figure 1 shows a representation of a location solution for a network with $n = 10$ location candidates and $p = 3$ open facilities.

y:	0	1	0	0	1	0	0	0	1	0
	1	2	3	4	5	6	7	8	9	10

Figure 1: Representation of the location variable y.

In general terms, the idea of the ALNS is to iteratively destroy and repair parts of a solution with the aim of finding better solutions. There are a number of destroy and repair operators, and they compete to be used at each iteration. Operators that have performed better in the past have a higher probability of being used. In our implementation, destroy and repair mechanisms have been created with the aim of closing and opening facilities, exploring our knowledge of the problem. As a result, allocation decisions are not made by the heuristic, which yields an integer problem in which location decisions are already fixed. This problem is then solved to optimality by mathematical programming at each iteration. The algorithm can therefore be described as a *matheuristic* [20], i.e., as a hybridization of a heuristic and of a mathematical programming algorithm.

The hybrid method aims to maximize the demand satisfaction. Facilities are removed and inserted in the solution at each iteration by means of destroy and repair operators. A roulette wheel mechanism controls the choice of the operators, with a probability that depends on their past performance. More concretely, to each operator r are associated a score π_r and a weight ω_r whose values depend on the past performance of the operator. Then, given h operators, operator $\bar{r}$ will be selected with probability $\omega_{\bar{r}} / \sum_{r=1}^{h} \omega_r$. Initially, all weights are set to one and all scores are set to zero. The search is divided into segments of θ iterations each, and the weights are computed by taking into account the performance of the operators during the last segment. At each iteration, the score of the selected operator is increased by σ_1 if the operator identifies a new best solution, by σ_2 if it identifies a solution better than the incumbent, and by σ_3 if the solution is not better but is still accepted. After θ iterations, the weights are updated by considering the scores obtained in the last segment as follows: let o_{rs} be the number of times operator r has been used in the last segment s. The updated weights are then

$$\omega_r := \begin{cases} \omega_r & \text{if } o_{rs} = 0 \\ (1-\eta)\omega_r + \eta \pi_r / o_{rs} & \text{if } o_{rs} \neq 0, \end{cases} \tag{9}$$

where $\eta \in [0, 1]$ is called the reaction factor and controls how quickly the weight adjustment reacts to changes in the operator performance. The scores are reset to zero at the end of each segment.

As in other ALNS implementations [8, 25, 26], an acceptance criterion based on simulated annealing is used. Let $z(\cdot)$ be the cost of solution $\cdot$. Given a solution s, a neighbor solution s' is always accepted if $z(s') > z(s)$, and is accepted with probability $e^{(z(s)-z(s'))/T}$ otherwise, where $T > 0$ is the current temperature. The temperature starts at T_{start} and is decreased by a cooling rate factor ϕ at each iteration, where $0 < \phi < 1$.

The main features of the algorithm are described in the next subsections. Optional initial solutions are described in Section 3.1. The list of destroy and repair operators is provided in Section 3.2. The subproblem resulting from each ALNS iteration and its solution procedure is described in Section 3.3, and an improvement procedure is described in Section 3.4. Finally, the parameters of the implementation and a pseudocode are provided in Section 3.5.

3.1. Initial solution

The algorithm can be initialized from an empty solution or from an arbitrary solution, e.g., randomly selecting p facilities to be opened. If the solution is empty, then no destroy operator can be applied at the first iteration, and only repair operators are used to open p facilities. This implementation starts with a random solution. Several papers have already demonstrated that the quality of the initial solution does not influence the overall performance of the ALNS algorithm [2, 7, 8].

3.2. List of operators

When designing the operators, the characteristics of the problem have been carefully considered. Given a solution, destroy and repair operators will close and open facilities, taking advantage of the problem's characteristics to conveniently exploit different neighborhoods. The list of the developed destroy and repair operators follows.

3.2.1. Destroy operators

1. Randomly close ρ facilities

This operator randomly selects ρ facilities and closes them. Here, ρ is a random number following a semi-triangular distribution with a negative slope, bounded at $[1, p]$, as shown in Figure 2.

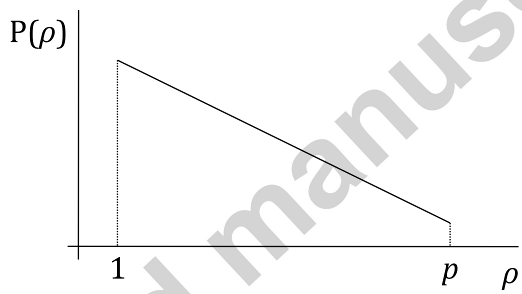

Figure 2: Probability density function of the semi-triangular random variable ρ.

According to this distribution, the value of ρ is calculated as in (10), where u is a random variable with uniform distribution in the interval $[0, 1]$:

$$\rho = \lfloor p - \sqrt{(1-u)(p-1)^2} + 0.5 \rfloor \tag{10}$$

This operator is useful for refining the solution since it does not change the solution much when ρ is small, which happens frequently due to the shape of its probability distribution. However, it still yields a major transformation of the solution when ρ is large.

2. Close the facility with the fewest potential clients

This operator identifies the opened facility with the fewest potential clients and closes it. Here, each opened facility j is evaluated and the number of clients within S units of distance from it is counted. It is useful for closing a facility which has few clients, allowing for a facility with a greater potential to be opened.

3. Close the facility with the smallest potential total demand

This operator closes the facility with the smallest potential demand allocated to it. It is similar to the previous removal heuristic, but here the total demand within S units of distance from each opened facility j is considered.

4. Close one of the two closest facilities

This operator identifies the two closest facilities opened in the current solution and randomly closes one of them. The rationale behind this operator is to avoid too much overlapping and to allow a facility to be opened such that more clients are served.

3.2.2. Repair operators

All repair operators identify the number of open facilities in the solution and are repeated as many times as necessary such that at the end of the repairing procedure the solution contains exactly p open facilities.

1. Randomly open one facility

This operator randomly opens one facility in the current solution. It is useful to diversify the search towards a good solution.

2. Open a facility located at, at least, $2S$ units from all open facilities

This operator opens a facility at the first candidate that is located more than $2S$ units from all open facilities. If no such facility exists, the one located the farthest away from all open facilities is inserted. The motivation for this operator is to open a facility to serve clients not yet served by any of the opened facilities.

3. Open a facility with the highest service potential (clients)

This operator evaluates all closed facilities and identifies the one that could serve the highest number of clients not yet served by any of the opened facilities. The idea is to serve clients not yet covered by any other facility, but this time overlapping is allowed, which helps increasing the service level. This is useful for the probabilistic part of the problem. In the event where all clients are already covered, a random facility is opened.

4. Open a facility with the highest service potential (demand)

This operator evaluates all closed facilities and identifies the one that could serve the highest demand not yet covered by any of the opened facilities. The idea is to serve the demand not yet covered by any other facility, but this time overlapping is allowed, which helps increasing the service level, useful for the probabilistic part of the problem. In the event where all the demand is already covered, a random facility is opened.

3.3. Subproblem solution

Once the ALNS heuristic has destroyed and repaired the solution, one needs to make all allocation decisions taking into account the probabilistic constraints in order to obtain the solution value. This is done efficiently by mathematical programming by solving the following IP, in which variables y_j are updated to their values $\bar{y}_j$ obtained from the ALNS heuristic. In this formulation, the bounds on $\bar{y}_j$ are already defined, implying that the constraint on the number of open facilities is automatically respected:

$$\text{maximize} \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}} d_i x_{ij} \tag{11}$$

subject to

$$\sum_{j \in \mathcal{N}} x_{ij} \leq 1 \quad i \in \mathcal{N} \tag{12}$$

$$x_{ij} \leq \bar{y}_j \quad i \in \mathcal{N}_j \quad j \in \mathcal{N} \tag{13}$$

$$\sum_{i \in \mathcal{N}_j} f_i x_{ij} \leq \mu \sqrt[b+2]{1-\alpha} \quad j \in \mathcal{N} \tag{14}$$

or

$$\sum_{i \in \mathcal{N}_j} f_i x_{ij} \leq \mu_j + \frac{1}{\tau} \ln(1-\alpha) \quad j \in \mathcal{N} \tag{15}$$

$$x_{ij} \in \{0,1\} \quad i \in \mathcal{N} \quad j \in \mathcal{N}. \tag{16}$$

This problem is significantly easier to solve than the problem defined by (1)–(7). Here, at each iteration one just needs to update the bounds on variables x_{ij} in constraints (13). Solving this subproblem by mathematical programming is relatively simple, given that one can take advantage of the fact that only a small portion of the problem changes at each iteration, thus the reoptimization is rather fast. Indeed, one can solve several of these problems per second.

3.4. Improvement procedure

An improvement procedure to polish a given good solution and try to locally improve it was also developed. This is done whenever the ALNS algorithm finds a new best solution, and at every θ iterations. In the improvement procedure, a mathematical programming model with the best known solution is defined, allowing two facilities to be closed and other two facilities to be opened. This can be seen as a neighborhood search in which all combinations of two opened facilities are closed and all combinations of two closed facilities are opened. If this yields an improved solution, the ALNS is updated with it. The improvement procedure consists of solving the following IP, in which $\bar{y}_j$ is a binary vector indicating whether facility j is opened (one) or closed (zero).

$$\text{maximize} \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}} d_i x_{ij} \tag{17}$$

subject to

$$\sum_{j \in \mathcal{N}} y_j = p \tag{18}$$

$$\sum_{j \in \mathcal{N}} x_{ij} \leq 1 \quad i \in \mathcal{N} \tag{19}$$

$$x_{ij} \leq y_j \quad i \in \mathcal{N}_j \quad j \in \mathcal{N} \tag{20}$$

$$\sum_{i \in \mathcal{N}_j} f_i x_{ij} \leq \mu \sqrt[b+2]{1-\alpha} \quad j \in \mathcal{N} \tag{21}$$

or

$$\sum_{i \in \mathcal{N}_j} f_i x_{ij} \leq \mu_j + \frac{1}{\tau} \ln(1-\alpha) \quad j \in \mathcal{N} \tag{22}$$

$$\sum_{j \in \mathcal{N}} \bar{y}_j y_j = p - 2 \tag{23}$$

$$y_j, x_{ij} \in \{0, 1\} \quad i \in \mathcal{N} \quad j \in \mathcal{N}. \tag{24}$$

In this problem, (17)–(22) are equal to (1)–(6). The added constraint (23) ensures that $p-2$ of the opened facilities need to remain open. The algorithm needs to open two closed facilities to respect constraint (18). This local search heuristic is not added as an operator of the ALNS because it is significantly more complex than the remaining operators.

3.5. Parameter settings and pseudocode

The parameter settings used in the ALNS implementation are now described. These were set after an early tuning phase. The maximum number of iterations i_{max} depends on the starting temperature T_{start} and on the cooling rate ϕ. These parameters were set as follows:

$$T_{start} = 30000 \tag{25}$$

$$\phi = \left(0.01/T_{start}\right)^{1/i_{max}}. \tag{26}$$

This makes the cooling rate a function of the desired number of iterations, adjusting accordingly the probability that the ALNS mechanism will accept worsening solutions. The stopping criterion is satisfied when the temperature reaches 0.01. In this implementation, the maximum number of iterations i_{max} was set to 1000 for instances with less than 100 clients, 2000 for instances with less than 500 clients, and 3000 for instances with more than 500 clients. The segment length θ was set to 200 iterations, and the reaction factor η was set to 0.7, thus defining the new weights by 70% of the performance on the last segment and 30% of the last weight value. The scores are updated with $\sigma_1 = 10$, $\sigma_2 = 5$ and $\sigma_3 = 2$. Algorithm 1 shows the pseudocode of the ALNS implementation.

4. Computational experiments

This section provides details and results of the extensive computational experiments carried out to assess the quality of the ALNS matheuristic. The algorithm was coded in C++ and the subproblems were solved by CPLEX using Concert Technology version 12.6. All computations were performed on machines equipped with an Intel Xeon™ processor running at 2.66 GHz. The ALNS algorithm needed less than 1 GB of RAM memory. The model described in Section 2 was implemented in CPLEX and executed on machines with up to 48 GB of RAM. All the machines run under the Scientific Linux 6.0 operating system.

A large dataset of benchmark instances with three classes of instances was solved: 26 instances contain 30 nodes, 24 instances contain 324 nodes, and 24 instances contain 818 nodes. The number p of facilities to open varies from two to 50. The service radius is equal to 1.5 miles for the instances with 30 nodes, 250 meters for instances with 324 nodes, and 750 meters for instances with 818 nodes. The 30-node dataset was proposed by Marianov and Serra [21] and the 324 and 818-node datasets were proposed by Corrêa et al. [10]. These datasets have since then been used to evaluate the heuristic of Marianov and Serra [21], the constructive genetic algorithm of Corrêa and

Algorithm 1: Hybrid ALNS matheuristic - part 1

Initialize: set all weights equal to 1 and all scores equal to 0.
$s_{best} \leftarrow s \leftarrow \textit{initial solution}$, $T \leftarrow T_{start}$.
while $T > 0.01$ **do**
    $s' \leftarrow s$;
    select a destroy and a repair operator using the roulette-wheel mechanism based on the current weights. Apply the operators to s' and update the number of times they are used;
    solve subproblem defined by (11)–(16), obtaining $z(s')$.
    if $z(s') > z(s)$ **then**
        $s \leftarrow s'$;
        if $z(s) > z(s_{best})$ **then**
            $s_{best} \leftarrow s$;
            update the score for the operators used with σ_1;
            apply the improvement procedure to s_{best}.
        else
            update the score for the operators used with σ_2.
        end if
    else
        if s' is accepted by the simulated annealing criterion **then**
            $s \leftarrow s'$;
            update the scores for the operators used with σ_3.
        end if
    end if

Algorithm 1: Hybrid ALNS matheuristic - part 2 (continued)

    if the iteration count is a multiple of θ **then**
        update the weights of all operators and reset their scores;
        apply the improvement procedure to s_{best}.
    end if
    $T \leftarrow \phi T$;
end while
return s_{best};

Lorena [9], the clustering search algorithm of Corrêa et al. [10], and the column generation heuristic of Corrêa et al. [11].

The name of the instances in tables 1–3 contains the values of the parameters used in each run. For example, instance 30_2_0_0_85 refers to a 30-node problem with two facilities, the congestion type based on the number of clients (0 for queue size, 1 for waiting time), the congestion parameter (number of clients b on the queue, or the waiting time τ in minutes) and the minimum probability α, in percentage value. The rate parameter μ is fixed at 72 for the 30-node network, and at 96 for the 324 and 818-node networks. The parameter f_i that appears in formulations (1)–(7), (11)–(16) and (17)–(24) is calculated as fd_i, with $f = 0.01$ for the 324 and 818-node network. For the 30-node network, $f = 0.015$ (queue size type constraints) or $f = 0.006$ (waiting time type constraints).

Tables 1–3 contain, for each instance, the best and the average solutions (when available) and the respective solution times obtained from six different methods. The best solution value for each instance is presented in boldface.

In the CPLEX section of each table, an asterisk denotes a new best known solution obtained by an exact method. For the column generation heuristic, the column *Gap (%)* contains the percentage difference between the solution and a heuristic upper bound. Notice that some of the values are non-zero even when the corresponding instance is solved to optimality, which is an indication that this algorithm may be using the value of the best known solution to terminate the execution.

For the ALNS results, each instance was executed 3 times. The *Average* column in the ALNS section shows the consistency of the proposed method. The figures presented in the *Time (s)* columns are also averages.

As presented in Table 1, the proposed exact method proved optimality for all 26 instances. The ALNS algorithm obtained the optimal solution for all instances, performing better than all the methods of the literature as highlighted in the *Average* row at the bottom of the table, being 10 times faster than CPLEX.

In Table 2, using the proposed exact method, CPLEX proved optimality for 23 out of 24 instances. The ALNS algorithm was able to find the optimal solution for 19 instances and performed better than the literature in all other five instances.

In Table 3, CPLEX proved optimality for all instances. The ALNS algorithm finds 19 of them and performed better than the previous state-of-the-art heuristic for one instance.

Due to the probabilistic constraints, the PMCLAP can be considered a capacitated facility location problem, a class of problems that are hard to be

Table 1: Detailed computational results for the instance set with 30 nodes

Instance	CPLEX			MS heuristic [21]		Constuctive GA [9]			Clustering Search [10]			Column Generation [11]			ALNS		
	Solution	Gap (%)	Time (s)	Solution	Time (s)	Best	Average	Time (s)	Best	Average	Time (s)	Solution	Gap (%)[1]	Time (s)	Best	Average	Time (s)
30_2_0_0_85	**3700**	0.000	0	**3700**	0.000	**3700**	3700	0.310	**3700**	3700.0	0.009	**3700**	0.000	0.530	**3700**	3700.0	12.333
30_2_0_1_85	**5100**	0.000	0	4630	0.000	5090	5090	0.360	5090	5090.0	0.018	5090	0.196	5.550	**5100**	5100.0	9.000
30_2_0_2_85	**5210**	0.000	0	4780	0.000	**5210**	5210	0.380	**5210**	5210.0	0.016	**5210**	1.321	4.770	**5210**	5210.0	4.667
30_2_0_2_95	**4520**	0.000	0	4470	0.000	**4520**	4513	0.300	**4520**	4520.0	0.015	**4520**	0.000	0.700	**4520**	4520.0	11.333
30_3_0_0_85	**5390**	0.000	0	5210	0.000	**5390**	5390	0.510	**5390**	5390.0	0.025	**5390**	0.119	28.030	**5390**	5390.0	5.000
30_3_0_1_85	**5390**	0.000	0	5210	0.000	**5390**	5390	0.480	**5390**	5390.0	0.024	**5390**	0.742	3.770	**5390**	5390.0	3.000
30_3_0_1_95	**5270**	0.000	6	5080	0.000	5240	5240	0.480	5240	5240.0	0.024	5240	0.763	14.750	**5270**	5270.0	44.667
30_3_0_2_85	**5390**	0.000	1	5210	0.000	**5390**	5390	0.470	**5390**	5390.0	0.023	**5390**	0.742	1.880	**5390**	5390.0	3.667
30_3_0_2_95	**5390**	0.000	0	5230	0.000	**5390**	5390	0.500	**5390**	5390.0	0.027	**5390**	0.622	11.450	**5390**	5390.0	3.667
30_3_1_49_90	**2160**	0.000	0	**2160**	0.000	**2160**	2160	0.530	**2160**	2160.0	0.009	**2160**	0.000	0.360	**2160**	2160.0	13.333
30_4_0_1_95	**5390**	0.000	0	5260	0.000	**5390**	5390	0.540	**5390**	5390.0	0.026	**5390**	1.391	30.880	**5390**	5390.0	8.333
30_4_1_42_85	**4600**	0.000	0	4550	0.000	**4600**	4600	0.610	**4600**	4600.0	0.022	**4600**	0.000	0.810	**4600**	4600.0	105.667
30_4_1_48_90	**1920**	0.000	0	1890	0.000	**1920**	1920	0.660	**1920**	1920.0	0.014	**1920**	0.000	0.440	**1920**	1920.0	17.333
30_4_1_49_90	**2880**	0.000	0	2870	0.000	**2880**	2877	0.590	**2880**	2880.0	0.013	**2880**	0.000	0.450	**2880**	2880.0	15.000
30_5_0_0_95	**5330**	0.000	13	5210	0.000	**5330**	5323	0.800	**5330**	5317.6	0.041	**5330**	0.375	6.910	**5330**	5330.0	288.667
30_5_1_40_85	**3050**	0.000	93	2910	0.000	3020	3001	0.740	**3050**	3033.2	0.021	**3050**	0.000	0.890	**3050**	3050.0	234.000
30_5_1_42_85	**5390**	0.000	1	5210	0.000	**5390**	5390	0.770	**5390**	5390.0	0.035	**5390**	0.464	34.170	**5390**	5390.0	15.000
30_5_1_48_90	**2400**	0.000	1	2280	0.000	2390	2390	0.740	**2400**	2398.8	0.019	**2400**	0.000	3.080	**2400**	2400.0	26.667
30_5_1_50_90	**4700**	0.000	1	4670	0.000	**4700**	4700	0.730	**4700**	4700.0	0.028	**4700**	0.000	0.830	**4700**	4700.0	42.667
30_6_0_0_95	**5410**	0.000	1	5390	0.000	**5410**	5392	0.840	**5410**	5391.0	0.037	**5410**	1.054	42.410	**5410**	5410.0	62.333
30_6_1_40_85	**3610**	0.605	10803	3480	0.000	**3610**	3610	0.910	**3610**	3608.8	0.031	**3610**	1.022	30.640	**3610**	3610.0	514.000
30_6_1_41_85	**5330**	0.188	10802	5120	0.000	**5300**	5274	1.000	**5330**	5286.4	0.038	**5300**	0.755	11.740	**5330**	5330.0	322.333
30_6_1_50_90	**5390**	0.000	1	5060	0.000	**5390**	5390	0.970	**5390**	5390.0	0.038	**5390**	0.395	48.440	**5390**	5390.0	31.667
30_7_1_40_85	**4060**	0.000	1	3860	0.000	**4060**	4060	1.030	**4060**	4060.0	0.039	**4060**	0.517	18.920	**4060**	4060.0	301.000
30_7_1_41_85	**5410**	0.000	1	5300	0.000	5390	5390	0.880	5390	5390.0	0.035	5350	2.159	41.840	**5410**	5410.0	61.333
30_8_1_41_85	**5470**	0.000	0	5390	0.000	**5470**	5470	0.950	**5470**	5470.0	0.032	**5470**	0.000	1.200	**5470**	5470.0	8.333
Average	**4533.1**	0.030	835.577	4389.6	0.000	4528.1	4525.0	0.657	4530.8	4527.1	0.025	4528.1	0.486	13.286	**4533.1**	4533.1	83.269

[1] heuristic gap

Table 2: Detailed computational results for the instance set with 324 nodes

Instance	CPLEX			MS heuristic [21]		Constuctive GA [9]			Clustering Search [10]			Column Generation [11]			ALNS		
	Solution	Gap (%)	Time (s)	Solution	Time (s)	Best	Average	Time (s)	Best	Average	Time (s)	Solution	Gap (%)[1]	Time (s)	Best	Average	Time (s)
324_10_0_0_85	**37180**	0.000	13	37081	0.220	37145	37069.0	8.100	37173	37163.2	3.460	**37180**	0.000	275.110	**37180**	37180.0	1255.333
324_10_0_0_95	**21460**	0.000	9	21386	0.140	21431	21373.0	8.630	21455	21447.2	3.160	**21460**	0.000	65.200	**21460**	21460.0	865.667
324_10_0_1_85	**51000**	0.000	22	50750	0.200	50880	50711.0	7.690	50948	50946.3	3.410	**51000**	0.000	507.810	**51000**	51000.0	1435.000
324_10_0_1_95	**35360**	0.000	14	35250	0.160	35342	35304.0	7.790	35359	35354.6	3.170	**35360**	0.000	47.330	**35360**	35360.0	857.000
324_10_0_2_85	**59740**	0.000	22	59598	0.200	59624	59437.0	7.620	59693	59688.5	3.330	**59740**	0.000	604.480	**59740**	59740.0	987.667
324_10_0_2_95	**45390**	0.000	9	45300	0.200	45347	45245.0	7.810	45374	45354.6	3.100	**45390**	0.000	327.840	**45390**	45390.0	1292.000
324_10_1_40_85	**27700**	0.000	14	27583	0.170	27675	27602.0	8.540	27698	27692.1	3.370	**27700**	0.000	238.390	**27700**	27700.0	1406.333
324_10_1_41_85	**29360**	0.000	6	29288	0.140	29324	29260.0	8.270	29351	29341.9	3.520	**29360**	0.000	145.880	**29360**	29360.0	1182.667
324_10_1_42_85	**30950**	0.000	11	30902	0.170	30932	30895.0	8.340	30948	30943.3	3.330	**30950**	0.000	42.910	**30950**	30950.0	1034.667
324_10_1_48_90	**26920**	0.000	15	26855	0.140	26883	26835.0	8.350	26917	26910.6	3.500	**26920**	0.000	144.720	**26920**	26920.0	1492.667
324_10_1_49_90	**28330**	0.000	22	28206	0.250	28280	28221.0	8.280	28318	28310.3	3.430	**28330**	0.000	315.700	**28330**	28330.0	1747.000
324_10_1_50_90	**29680**	0.000	6	29638	0.220	29641	29593.0	8.260	29672	29665.5	3.360	**29680**	0.000	81.810	**29680**	29680.0	1127.333
324_20_0_0_85	**74360***	0.000	198	73981	0.830	73407	73001.0	23.690	74165	74106.7	9.710	74358	0.003	2630.440	**74360**	74357.7	6191.667
324_20_0_0_95	**42920**	0.000	22	42714	0.730	42577	42318.0	24.300	42840	42804.9	9.370	**42920**	0.000	2425.520	**42920**	42919.7	3936.333
324_20_0_1_85	**102000***	0.000	612	100628	1.020	99576	98353.0	24.690	101374	101177.4	9.070	101973	0.024	10801.330	101978	101974.0	7212.667
324_20_0_1_95	**70720**	0.000	55	70368	0.740	70471	70308.0	23.400	70656	70628.2	9.060	**70720**	0.000	1067.410	**70720**	70720.0	4417.333
324_20_0_2_85	**119470***	0.008	10804	118451	0.770	116639	115235.0	23.330	118771	118613.6	8.990	119448	0.019	10801.590	119455	119447.3	6388.000
324_20_0_2_95	**90780***	0.000	74	90424	0.810	89970	89355.0	24.150	90556	90521.2	9.400	**90780**	0.000	2369.980	**90780**	90780.0	6667.333
324_20_1_40_85	**55400***	0.000	52	55006	0.840	54804	54414.0	23.920	55306	55226.7	9.650	55396	0.007	8457.660	55399	55398.3	5488.667
324_20_1_41_85	**58720**	0.000	85	58577	1.020	58009	57571.0	24.700	58604	58583.4	10.330	**58720**	0.000	1377.420	**58720**	58720.0	7417.667
324_20_1_42_85	**61900**	0.000	35	61637	0.880	61545	61266.0	23.810	61847	61822.6	9.740	**61900**	0.000	735.140	**61900**	61900.0	3561.333
324_20_1_48_90	**53840***	0.000	49	53377	0.630	53300	52958.0	24.320	53793	53689.5	9.820	53838	0.004	4645.490	53839	53838.7	5912.667
324_20_1_49_90	**56660***	0.000	167	56180	1.340	56216	55813.0	24.150	56532	56429.9	9.550	56654	0.009	10800.920	56655	56655.0	7130.667
324_20_1_50_90	**59360***	0.000	56	59119	0.940	58941	58577.0	26.030	59285	59242.6	10.080	**59360**	0.000	1273.390	**59360**	59360.0	5551.333
Average	**52883.3**	0.000	515.500	52595.8	0.532	52415.0	52113.1	16.174	52776.5	52736.0	6.455	52880.7	0.003	2507.645	**52881.5**	52880.9	3523.292

* indicates new best known solution obtained by an exact method
[1] heuristic gap

Table 3: Detailed computational results for the instance set with 818 nodes

Instance	CPLEX			MS heuristic [21]		Constuctive GA [9]			Clustering Search [10]			Column Generation [11]			ALNS		
	Solution	Gap (%)	Time (s)	Solution	Time (s)	Best	Average	Time (s)	Best	Average	Time (s)	Solution	Gap (%)[1]	Time (s)	Best	Average	Time (s)
818_10_0_0_95	**21460**	0.000	557	21429	4.940	21455	21449.3	43.650	**21460**	21459.9	11.230	**21460**	0.000	163.08	**21460**	21460.0	6818.000
818_10_0_1_95	**35360**	0.000	657	35339	13.050	35356	35346.0	50.450	**35360**	35360.0	11.540	**35360**	0.000	207.69	**35360**	35360.0	10407.000
818_10_0_2_95	**45390**	0.000	648	45375	12.090	45387	45377.4	49.370	**45390**	45390.0	12.170	**45390**	0.000	251.33	**45390**	45390.0	7400.000
818_10_1_48_90	**26920**	0.000	586	26907	11.380	26915	26901.0	48.350	**26920**	26920.0	11.320	**26920**	0.000	185.41	**26920**	26920.0	8212.000
818_10_1_49_90	**28330**	0.000	659	28309	8.630	28320	28298.0	47.780	**28330**	28330.0	12.230	**28330**	0.000	196.06	**28330**	28330.0	8052.667
818_10_1_50_90	**29680**	0.000	682	29661	18.280	29678	29673.8	47.810	**29680**	29680.0	11.330	**29680**	0.000	199.28	**29680**	29680.0	9078.333
818_20_0_0_85	**74360**	0.000	785	74313	75.050	74341	74279.4	90.040	**74360**	74359.8	66.810	**74360**	0.000	412.74	**74360**	74360.0	17183.667
818_20_0_0_95	**42920**	0.000	562	42793	25.670	42870	42817.7	76.200	**42920**	42918.9	54.840	**42920**	0.000	248.13	**42920**	42920.0	9890.333
818_20_0_1_85	**102000***	0.000	1491	101933	76.060	101955	101898.8	89.190	**102000**	101998.9	67.360	**102000**	0.000	447.06	**102000**	102000.0	16602.000
818_20_0_1_95	**70720**	0.000	610	70644	37.030	70653	70557.1	80.420	**70720**	70719.2	62.910	**70720**	0.000	390.03	**70720**	70720.0	16880.667
818_20_0_2_85	**119480***	0.000	1579	119397	105.480	119445	119306.4	89.180	**119480**	119477.6	74.360	**119480**	0.000	473.52	**119480**	119480.0	19074.667
818_20_0_2_95	**90780**	0.000	841	90730	75.190	90747	90672.9	84.990	**90780**	90779.6	66.800	**90780**	0.000	460.19	**90780**	90780.0	14622.667
818_20_1_40_85	**55400***	0.000	833	55325	66.110	55341	55235.7	88.330	**55400**	55399.0	61.040	**55400**	0.000	293.62	**55400**	55400.0	15044.333
818_20_1_41_85	**58720***	0.000	641	58637	69.630	58706	58677.6	87.620	**58720**	58719.9	57.310	**58720**	0.000	372.28	**58720**	58720.0	14384.000
818_20_1_42_85	**61900***	0.000	929	61814	71.810	61839	61719.7	90.430	**61900**	61898.8	58.710	**61900**	0.000	380.64	**61900**	61900.0	14487.000
818_20_1_48_90	**53840**	0.000	640	53728	22.410	53814	53762.2	87.870	**53840**	53839.6	59.560	**53840**	0.000	343.08	**53840**	53840.0	13808.667
818_20_1_49_90	**56660**	0.000	948	56602	34.340	56605	56465.3	88.710	**56660**	56660.0	67.420	**56660**	0.000	324.64	**56660**	56660.0	13008.333
818_20_1_50_90	**59360**	0.000	877	59269	39.840	59336	59279.8	87.040	**59360**	59359.9	58.480	**59360**	0.000	364.84	**59360**	59360.0	14568.667
818_50_0_0_85	**185900***	0.000	2956	185426	756.720	184428	184153.6	177.340	185880	185775.6	364.200	185898	0.001	3695.94	185637	185587.7	24094.333
818_50_0_1_85	**255000***	0.000	6332	254509	730.200	253438	252884.6	171.860	254985	254905.0	387.370	**255000**	0.000	5072.75	254996	254987.3	23978.000
818_50_0_2_85	**298700***	0.000	4435	298217	757.730	296763	296182.8	171.740	298582	298517.6	392.290	298490	0.070	6004.53	298692	298648.3	23729.667
818_50_1_48_90	**134600***	0.000	785	134088	596.110	134079	133999.4	177.670	134598	134561.6	335.000	**134600**	0.000	2317.55	134597	134593.3	25037.333
818_50_1_49_90	**141650***	0.000	1719	141281	571.170	140439	140143.4	174.960	141586	141532.8	356.800	**141650**	0.000	2713.86	**141650**	141606.7	24425.000
818_50_1_50_90	**148400***	0.000	1946	147931	667.670	147202	147123.8	173.960	148383	148321.9	337.400	**148400**	0.000	2444.64	148397	148378.0	25031.667
Average	**91563.8**	0.000	1362.417	91402.4	201.941	91213.0	91091.9	98.957	91553.9	91536.9	124.937	91554.9	0.003	1165.120	91552.0	91545.1	15659.125

* indicates new best known solution obtained by an exact method

[1] heuristic gap

solved by exact methods. To test this assumption, a new network with 1177 nodes was tested in order to evaluate the performance of CPLEX and ALNS on a larger problem. These instances used the same set of parameters of the 818-node network but with a service radius of 950 meters. Table 4 presents the results for an one-hour and a three-hour run of CPLEX and three one-hour runs for ALNS. As shown on Table 4, the ALNS heuristic is capable of providing consistently good results even for larger instances. In fact, in two of them the heuristic was capable of finding better solution values than CPLEX, even with only one third of the running time. This study also shows the difficulties for CPLEX to obtain good solution values for more restricted instances, i.e., those with fewer facilities.

5. Conclusion

This study presented a hybrid method for solving the probabilistic maximal covering location-allocation problem. The algorithm is based on an adaptive large neighborhood search heuristic to determine the location decisions of the problem. Allocation subproblems are solved exactly by mathematical programming at each iteration. A flexible improvement procedure polishes good solutions obtained by the metaheuristic. A high performance solver has been employed to obtain bounds and prove optimality for some instances. This exact approach has found new best known solutions for 19 instances, proving optimality for 18 of them. The hybrid method performed very consistently, finding the best known solutions for 94,5% of the instances, outperforming the state-of-the-art heuristic method from the literature. A new dataset was tested and confirmed the superiority of the heuristic in instances that are

Table 4: Detailed computational results for the instance set with 1177 nodes

Instance	CPLEX 1h		CPLEX 3h			ALNS			
	Solution	Gap (%)	Solution	Gap (%)	Time	Run 1	Run 2	Run 3	Average
1177_10_0_0_95	**21460**	0.000	**21460***	0.000	3336	**21460**	**21460**	**21460**	21460.0
1177_10_0_1_95	35360	-1515.054	**35360***	0.000	4361	**35360**	**35360**	**35360**	35360.0
1177_10_0_2_95	45377	-1300.672	**45390***	0.000	4601	**45390**	**45390**	**45390**	45390.0
1177_10_1_48_90	26920	-2127.272	**26920***	0.000	5111	**26920**	**26920**	**26920**	26920.0
1177_10_1_49_90	28259	-2075.814	**28330***	0.000	5766	**28330**	**28330**	**28330**	28330.0
1177_10_1_50_90	29473	-2013.025	**29680***	0.000	5040	**29680**	**29680**	**29680**	29680.0
1177_20_0_0_85	38747	-1632.177	**74360***	0.000	4117	**74360**	**74360**	**74360**	74360.0
1177_20_0_0_95	29357	-2403.658	**42920***	0.000	4201	**42920**	**42920**	**42920**	42920.0
1177_20_0_1_85	42256	-1639.396	**102000***	0.000	10306	**102000**	**102000**	**102000**	102000.0
1177_20_0_1_95	37848	-1529.910	**70720***	0.000	5092	**70720**	**70720**	**70720**	70720.0
1177_20_0_2_85	44715	-1543.741	119475	-7.285	10801	**119480***	**119480***	**119480***	119480.0
1177_20_0_2_95	40996	-1692.855	**90780***	0.000	6392	**90780**	**90780**	**90780**	90780.0
1177_20_1_48_90	36211	-1929.767	**53840***	0.000	5813	**53840**	**53840**	**53840**	53840.0
1177_20_1_49_90	56657	-0.005	**56660***	0.000	3449	**56660**	**56660**	**56660**	56660.0
1177_20_1_50_90	38813	-1435.902	**59360***	0.000	4364	**59360**	**59360**	**59360**	59360.0
1177_50_0_0_85	38747	-1796.918	**185900***	0.000	7882	185671	185653	185511	185611.7
1177_50_0_1_85	43192	-1601.702	**254999***	0.000	10801	254959	254966	254979	254968.0
1177_50_0_2_85	283145	-159.584	298311	-146.387	10801	298687	**298688***	298683	298686.0
1177_50_1_48_90	134596	-0.003	**134600***	0.000	6791	**134600**	134578	134492	134556.7
1177_50_1_49_90	141648	-0.001	**141650***	0.000	3637	141105	141477	141306	141296.0
1177_50_1_50_90	147574	-8.163	**148400***	0.000	4369	148239	148370	148387	148332.0
1177_70_1_40_85	48479	-1308.048	**193893***	-0.004	10802	193392	193130	193495	193339.0
1177_70_1_41_85	49522	-1384.187	**205513***	-0.003	10801	204099	203793	203727	203873.0
1177_70_1_42_85	203544	-261.101	**216650***	0.000	10772	215354	215744	216186	215761.3
Average	68454.000	-1223.290	109882.125	-6.403	6641.917	109723.6	109735.8	109751.1	109736.8

* indicates new best known solution obtained by an exact or heuristic method difficult to solve by exact methods.

Acknowledgments

The first author thanks the CIRRELT, the Department of Operations and Decision Systems and the Faculty of Administration Sciences of Université Laval for partial financial support for his research trip. The third author thanks the Conselho Nacional de Desenvolvimento Científico e Tecnológico – CNPq, under grants 476862/2012-4 and 300692/2009-9. The authors thank Calcul Québec for providing parallel computing facilities. The authors thanks the useful comments and suggestions of two anonymous referees.

[1] R. G. I. Arakaki and L. A. N. Lorena. A constructive genetic algorithm for the maximal covering location problem. In *Proceedings of the 4th Metaheuristics International Conference*, Porto, Portugal, 2001.

[2] E. Barrena, D. C. Ortiz, L. C. Coelho, and G. Laporte. A fast and efficient adaptive large neighborhood search heuristic for the passenger train timetabling problem with dynamic demand. Technical report, CIRRELT-2013-64, Montreal, Canada, 2013.

[3] P. Bartodziej, U. Derigs, D. Malcherek, and U. Vogel. Models and algorithms for solving combined vehicle and crew scheduling problems with rest constraints: an application to road feeder service planning in air cargo transportation. *OR Spectrum*, 31(2):405–429, 2009.

[4] O. Berman, Z. Drezner, and G. O. Wesolowsky. The maximal covering problem with negative weights. *Geographical Analysis*, 41(1):30–42, 2009.

[5] O. Berman, I. Hajizadeh, and D. Krass. The maximum covering problem with travel time uncertainty. *IIE Transactions*, 45(1):81–96, 2013.

[6] R. Church and C. ReVelle. The maximal covering location problem. *Papers in Regional Science*, 32(1):101–118, 1974.

[7] L. C. Coelho, J.-F. Cordeau, and G. Laporte. The inventory-routing problem with transshipment. *Computers & Operations Research*, 39(11):2537–2548, 2012.

[8] L. C. Coelho, J.-F. Cordeau, and G. Laporte. Consistency in multi-vehicle inventory-routing. *Transportation Research Part C: Emerging Technologies*, 24(1):270–287, 2012.

[9] F. A. Corrêa and L. A. N. Lorena. Using the constructive genetic algorithm for solving the probabilistic maximal covering location-allocation problem. In *Proceedings of the I Workshop on Computational Intelligence*, Ribeirão Preto, Brazil, 2006.

[10] F. A. Corrêa, A. A. Chaves, and L. A. N. Lorena. Hybrid heuristics for the probabilistic maximal covering location-allocation problem. *Operational Research. An International Journal*, 7(3):323–344, 2008.

[11] F. A. Corrêa, L. A. N. Lorena, and G. M. Ribeiro. A decomposition approach for the probabilistic maximal covering location-allocation problem. *Computers & Operations Research*, 36(10):2729–2739, 2009.

[12] M. S. Daskin. *Network and Discrete Location: Models, Algorithms and Applications*. John Wiley & Sons, New York, 1995.

[13] R. D. Galvão. Uncapacitated facility location problems: contributions. *Pesquisa Operacional*, 24(1):7–38, 2004.

[14] R. D. Galvão and C. ReVelle. A lagrangean heuristic for the maximal covering location problem. *European Journal of Operational Research*, 88(1):114–123, 1996.

[15] T. S. Hale and C. R. Moberg. Location science research: A review. *Annals of Operations Research*, 123(1–4):21–35, 2003.

[16] V. C. Hemmelmayr, J.-F. Cordeau, and T. G. Crainic. An adaptive large neighborhood search heuristic for two-echelon vehicle routing problems arising in city logistics. *Computers & Operations Research*, 39(12):3215–3228, 2012.

[17] M. Hewitt, G. L. Nemhauser, and M. W. P. Savelsbergh. Combining exact and heuristic approaches for the capacitated fixed-charge network flow problem. *INFORMS Journal on Computing*, 22(2):314–325, 2010.

[18] G. Laporte, R. Musmanno, and F. Vocaturo. An adaptive large neighbourhood search heuristic for the capacitated arc-routing problem with stochastic demands. *Transportation Science*, 44(1):125–135, 2010.

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

[20] V. Maniezzo, T. Stützle, and S. Voß. *Matheuristics: Hybridizing Metaheuristics and Mathematical Programming*. Springer, New York, 2009.

[21] V. Marianov and D. Serra. Probabilistic maximal covering location-allocation models for congested systems. *Journal of Regional Science*, 38(3):401–424, 1998.

[22] A.-S. Pepin, G. Desaulniers, A. Hertz, and D. Huisman. A comparison of five heuristics for the multiple depot vehicle scheduling problem. *Journal of Scheduling*, 12(1):17–30, 2009.

[23] M. A. Pereira, L. A. N. Lorena, and E. L. F. Senne. A column generation approach for the maximal covering location problem. *International Transactions in Operations Research*, 14(1):349–364, 2007.

[24] H. Pirkul and D. A. Schilling. The maximal covering location problem with capacities on total workload. *Management Science*, 37(2):233–248, 1991.

[25] S. Ropke and D. Pisinger. An adaptive large neighborghood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40(4):455–472, 2006.

[26] S. Ropke and D. Pisinger. A unified heuristic for a large class of vehicle routing problems with backhauls. *European Journal of Operational Research*, 171(3):750–755, 2006.

[27] D. Serra and V. Marianov. New trends in public facility location modeling. Technical Report Working Paper 755, UPF Economics and Business, Barcelona, 2004.