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

February 2014

CIRREL-2014-10
A Hybrid Method for the Probabilistic Maximal Covering Location-Allocation Problem

Marcos A. Pereira¹, Leandro C. Coelho²*, Luiz A.N. Lorena³, Ligia C. De Souza³

¹ São Paulo State University, Av. Dr. Ariberto Pereira da Cunha, 333, Guaratinguetá, Brazil 12516-410
² Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT) and Department of Operations and Decision Systems, Université Laval, 2325, de la Terrasse, Québec, Canada G1V 0A6
³ 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 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.

Keywords. Facility location, congested systems, hybrid algorithm, adaptive large neighborhood search, exact method, queueing maximal covering location-allocation model, PMCLAP.

Acknowledgements. Marcos A. Pereira 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. Leandro C. Coelho 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.

Results and views expressed in this publication are the sole responsibility of the authors and do not necessarily reflect those of CIRRELT.

Les résultats et opinions contenus dans cette publication ne reflètent pas nécessairement la position du CIRRELT et n’engagent pas sa responsabilité.

* Corresponding author: Leandro.Coelho@cirrelt.ca

© Pereira, Coelho, Lorena, de Souza and CIRRELT, 2014
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 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 Queuing 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 $\mu$. The parameter $\alpha$ defines the minimum probability of, at most:

- a queue with $b$ clients, or;
- a waiting time of $\tau$ 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:
\[
\text{maximize } \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}_i} d_{ij} x_{ij} \quad (1)
\]

subject to

\[
\sum_{j \in \mathcal{N}_i} x_{ij} \leq 1 \quad i \in \mathcal{N} \quad (2)
\]

\[
\sum_{j \in \mathcal{N}} y_j = p \quad (3)
\]

\[
x_{ij} \leq y_j \quad i \in \mathcal{N} \quad j \in \mathcal{N} \quad (4)
\]

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

or

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

\[
y_j, x_{ij} \in \{0, 1\} \quad i \in \mathcal{N} \quad j \in \mathcal{N} \quad (7)
\]

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) and (6) are related to the probabilistic nature of the problem. Specifically, constraints (5) ensure that facility \( j \) has less than \( b \) clients in the queue with at least probability \( \alpha \), and constraints (6) ensure that the waiting time for service at facility \( j \) is at most \( \tau \) minutes with probability of at least \( \alpha \). Constraints (7) define the binary nature of the variables. Obviously, \( x_{ij} \) equals zero for all \( j \notin \mathcal{N}_i \). The PMCLAP is NP-hard [24], therefore this model can only be solved exactly in reasonable computing times for small instances. We evaluate the performance of a state-of-the-art solver in Section 4.

Observe that, by disregarding constraints (5) and (6), the remaining model corresponds to the maximal covering location problem.
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.

\[
y: \begin{array}{cccccccccc}
0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 
\end{array}
\]

**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 $\pi_r$ and a weight $\omega_r$ 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 $\theta$ 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 $\sigma_1$ if the operator identifies a new best solution, by $\sigma_2$ if it identifies a solution better than the incumbent, and by $\sigma_3$ if the solution is not better but is still accepted. After $\theta$ 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

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

(8)

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 $T_{\text{start}}$ and is decreased by a cooling rate factor $\phi$ 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 $\rho$ facilities

This operator randomly selects $\rho$ facilities and closes them. Here, $\rho$ is a random number following a semi-triangular distribution with a negative slope, bounded at $[1, p]$. It is useful for refining the solution since it does not change the solution much when $\rho$ is small, which happens frequently due to the shape of its probability distribution. However, it still yields a major transformation of the solution when $\rho$ is large.

2. Close the facility with fewer potential clients

This operator identifies the opened facility with fewer 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 smaller 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 is repeated as many times as necessary such that in the end the solution contains $p$ 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 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:

$$\text{maximize } \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}} d_i x_{ij}$$  \hspace{1cm} (9)

subject to

$$\sum_{j \in \mathcal{N}} x_{ij} \leq 1 \quad i \in \mathcal{N}$$  \hspace{1cm} (10)

$$x_{ij} \leq \bar{y}_j \quad i \in \mathcal{N}_j \quad j \in \mathcal{N}$$  \hspace{1cm} (11)

$$\sum_{i \in \mathcal{N}_j} f_i x_{ij} \leq \mu^b \sqrt{1 - \alpha} \quad j \in \mathcal{N}$$  \hspace{1cm} (12)

or

$$\sum_{i \in \mathcal{N}_j} f_i x_{ij} \leq \mu_j + \frac{1}{\tau} \ln (1 - \alpha) \quad j \in \mathcal{N}$$  \hspace{1cm} (13)

$$x_{ij} \in \{0, 1\} \quad i \in \mathcal{N} \quad j \in \mathcal{N}.$$  \hspace{1cm} (14)
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 (11). 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 $\theta$ 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).

\[
\begin{align*}
\text{maximize} & \quad \sum_{i \in \mathcal{N}} \sum_{j \in \mathcal{N}} d_i x_{ij} \\
\text{subject to} & \\
\sum_{j \in \mathcal{N}} y_j = p \\
\sum_{j \in \mathcal{N}} x_{ij} & \leq 1 \quad i \in \mathcal{N} \\
x_{ij} & \leq y_j \quad i \in \mathcal{N}_j \quad j \in \mathcal{N} \\
\sum_{i \in \mathcal{N}_j} f_i x_{ij} & \leq \mu \frac{\beta + \sqrt{1 - \alpha}}{2} \quad j \in \mathcal{N}
\end{align*}
\]
or
\[ \sum_{i \in N} f_i x_{ij} \leq \mu_j + \frac{1}{\tau} \ln (1 - \alpha) \quad j \in N \]  
\[ \sum_{j \in N} \bar{y}_j y_j = p - 2 \]  
\[ y_j, x_{ij} \in \{0, 1\} \quad i \in N \quad j \in N. \]  
(20) (21) (22)

In this problem, (15)–(20) are equal to (1)–(6). The added constraint (21) ensures that \( p - 2 \) of the opened facilities need to remain open. The algorithm needs to open two closed facilities to respect constraint (16). 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 \( i_{\text{max}} \) depends on the starting temperature \( T_{\text{start}} \) and on the cooling rate \( \phi \). These parameters were set as follows:

\[ T_{\text{start}} = 30000 \]  
\[ \phi = (0.01/T_{\text{start}})^{1/i_{\text{max}}}. \]  
(23) (24)

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 \( i_{\text{max}} \) 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 \( \theta \) was set to 200 iterations, and the reaction factor \( \eta \) 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.
Algorithm 1: Hybrid ALNS matheuristic - part 1

1: Initialize: set all weights equal to 1 and all scores equal to 0.

2: $s_{best} \leftarrow s \leftarrow \text{initial solution}, T \leftarrow T_{start}$.

3: while $T > 0.01$ do

4:  $s' \leftarrow s$;

5:  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;

6:  solve subproblem defined by (9)–(14), obtaining $z(s')$.

7:  if $z(s') > z(s)$ then

8:     $s \leftarrow s'$;

9:  if $z(s) > z(s_{best})$ then

10:     $s_{best} \leftarrow s$;

11:     update the score for the operators used with $\sigma_1$;

12:     apply the improvement procedure to $s_{best}$.

13:  else

14:     update the score for the operators used with $\sigma_2$.

15:  end if

16: else

17:  if $s'$ is accepted by the simulated annealing criterion then

18:     $s \leftarrow s'$;

19:     update the scores for the operators used with $\sigma_3$.

20:  end if

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

22: \textbf{if} the iteration count is a multiple of $\theta$ \textbf{then}
23: \hspace{1em} update the weights of all operators and reset their scores;
24: \hspace{1em} apply the improvement procedure to $s_{\text{best}}$.
25: \textbf{end if}
26: \hspace{1em} $T \leftarrow \phi T$;
27: \textbf{end while}
28: \textbf{return} $s_{\text{best}}$;

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 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 \( \tau \) in minutes) and the minimum probability \( \alpha \), in percentage value. The rate parameter \( \mu \) 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), (9)–(14) and (15)–(22) is calculated as \( f d_i \), 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 \( \text{Gap} \% \) contains the percentage difference between the solution and a heuristic upper bound.

For the ALNS results, each instance was executed 3 times. The \( \text{Average} \) column in the ALNS section shows the consistency of the proposed method. The figures presented in the \( \text{Time (s)} \) 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 \( \text{Average} \) 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.
Table 1: Detailed computational results for the instance set with 30 nodes

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>1</td>
<td>0.00</td>
<td>3700</td>
<td>0.00</td>
<td>3700</td>
<td>0.31</td>
</tr>
<tr>
<td>30</td>
<td>2</td>
<td>0.00</td>
<td>5100</td>
<td>0.00</td>
<td>5100</td>
<td>0.36</td>
</tr>
<tr>
<td>30</td>
<td>3</td>
<td>0.00</td>
<td>5210</td>
<td>0.00</td>
<td>5210</td>
<td>0.30</td>
</tr>
<tr>
<td>30</td>
<td>4</td>
<td>0.00</td>
<td>4520</td>
<td>0.00</td>
<td>4520</td>
<td>0.30</td>
</tr>
<tr>
<td>30</td>
<td>5</td>
<td>0.00</td>
<td>5390</td>
<td>0.00</td>
<td>5390</td>
<td>0.48</td>
</tr>
<tr>
<td>30</td>
<td>6</td>
<td>0.00</td>
<td>5270</td>
<td>0.00</td>
<td>5270</td>
<td>0.48</td>
</tr>
<tr>
<td>30</td>
<td>7</td>
<td>0.00</td>
<td>5390</td>
<td>0.00</td>
<td>5390</td>
<td>0.47</td>
</tr>
<tr>
<td>30</td>
<td>8</td>
<td>0.00</td>
<td>5390</td>
<td>0.00</td>
<td>5390</td>
<td>0.46</td>
</tr>
<tr>
<td>30</td>
<td>9</td>
<td>0.00</td>
<td>5390</td>
<td>0.00</td>
<td>5390</td>
<td>0.46</td>
</tr>
<tr>
<td>30</td>
<td>10</td>
<td>0.00</td>
<td>5390</td>
<td>0.00</td>
<td>5390</td>
<td>0.46</td>
</tr>
<tr>
<td>30</td>
<td>11</td>
<td>0.00</td>
<td>2160</td>
<td>0.00</td>
<td>2160</td>
<td>0.53</td>
</tr>
<tr>
<td>30</td>
<td>12</td>
<td>0.00</td>
<td>5390</td>
<td>0.00</td>
<td>5390</td>
<td>0.52</td>
</tr>
<tr>
<td>30</td>
<td>13</td>
<td>0.00</td>
<td>5390</td>
<td>0.00</td>
<td>5390</td>
<td>0.52</td>
</tr>
<tr>
<td>30</td>
<td>14</td>
<td>0.00</td>
<td>5390</td>
<td>0.00</td>
<td>5390</td>
<td>0.52</td>
</tr>
<tr>
<td>30</td>
<td>15</td>
<td>0.00</td>
<td>5390</td>
<td>0.00</td>
<td>5390</td>
<td>0.52</td>
</tr>
<tr>
<td>30</td>
<td>16</td>
<td>0.00</td>
<td>5390</td>
<td>0.00</td>
<td>5390</td>
<td>0.52</td>
</tr>
<tr>
<td>30</td>
<td>17</td>
<td>0.00</td>
<td>5390</td>
<td>0.00</td>
<td>5390</td>
<td>0.52</td>
</tr>
<tr>
<td>30</td>
<td>18</td>
<td>0.00</td>
<td>5390</td>
<td>0.00</td>
<td>5390</td>
<td>0.52</td>
</tr>
<tr>
<td>30</td>
<td>19</td>
<td>0.00</td>
<td>5390</td>
<td>0.00</td>
<td>5390</td>
<td>0.52</td>
</tr>
<tr>
<td>30</td>
<td>20</td>
<td>0.00</td>
<td>5390</td>
<td>0.00</td>
<td>5390</td>
<td>0.52</td>
</tr>
<tr>
<td>30</td>
<td>21</td>
<td>0.00</td>
<td>5390</td>
<td>0.00</td>
<td>5390</td>
<td>0.52</td>
</tr>
<tr>
<td>30</td>
<td>22</td>
<td>0.00</td>
<td>5390</td>
<td>0.00</td>
<td>5390</td>
<td>0.52</td>
</tr>
<tr>
<td>30</td>
<td>23</td>
<td>0.00</td>
<td>5390</td>
<td>0.00</td>
<td>5390</td>
<td>0.52</td>
</tr>
<tr>
<td>30</td>
<td>24</td>
<td>0.00</td>
<td>5390</td>
<td>0.00</td>
<td>5390</td>
<td>0.52</td>
</tr>
<tr>
<td>30</td>
<td>25</td>
<td>0.00</td>
<td>5390</td>
<td>0.00</td>
<td>5390</td>
<td>0.52</td>
</tr>
<tr>
<td>30</td>
<td>26</td>
<td>0.00</td>
<td>5390</td>
<td>0.00</td>
<td>5390</td>
<td>0.52</td>
</tr>
<tr>
<td>30</td>
<td>27</td>
<td>0.00</td>
<td>5390</td>
<td>0.00</td>
<td>5390</td>
<td>0.52</td>
</tr>
<tr>
<td>30</td>
<td>28</td>
<td>0.00</td>
<td>5390</td>
<td>0.00</td>
<td>5390</td>
<td>0.52</td>
</tr>
<tr>
<td>30</td>
<td>29</td>
<td>0.00</td>
<td>5390</td>
<td>0.00</td>
<td>5390</td>
<td>0.52</td>
</tr>
<tr>
<td>30</td>
<td>30</td>
<td>0.00</td>
<td>5390</td>
<td>0.00</td>
<td>5390</td>
<td>0.52</td>
</tr>
<tr>
<td>Average</td>
<td>4533.1</td>
<td>0.30</td>
<td>835.577</td>
<td>4389.6</td>
<td>0.00</td>
<td>4528.1</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Solution</td>
<td>Gap (%)</td>
<td>Time (s)</td>
<td>Best</td>
<td>Average</td>
<td>Time (s)</td>
</tr>
<tr>
<td>324.10.0.45</td>
<td>37180</td>
<td>0.000</td>
<td>13</td>
<td>37181</td>
<td>0.220</td>
<td>37173</td>
</tr>
<tr>
<td>324.10.0.95</td>
<td>21460</td>
<td>0.000</td>
<td>9</td>
<td>21460</td>
<td>0.140</td>
<td>21455</td>
</tr>
<tr>
<td>324.10.1.45</td>
<td>51000</td>
<td>0.000</td>
<td>22</td>
<td>50750</td>
<td>0.200</td>
<td>50880</td>
</tr>
<tr>
<td>324.10.1.95</td>
<td>35360</td>
<td>0.000</td>
<td>14</td>
<td>35250</td>
<td>0.160</td>
<td>35421</td>
</tr>
<tr>
<td>324.10.2.45</td>
<td>59740</td>
<td>0.000</td>
<td>22</td>
<td>59598</td>
<td>0.200</td>
<td>59624</td>
</tr>
<tr>
<td>324.10.2.95</td>
<td>45390</td>
<td>0.000</td>
<td>9</td>
<td>45300</td>
<td>0.200</td>
<td>45347</td>
</tr>
<tr>
<td>324.10.4.0.85</td>
<td>27700</td>
<td>0.000</td>
<td>14</td>
<td>27583</td>
<td>0.170</td>
<td>27608</td>
</tr>
<tr>
<td>324.10.4.1.85</td>
<td>29360</td>
<td>0.000</td>
<td>6</td>
<td>29288</td>
<td>0.140</td>
<td>29324</td>
</tr>
<tr>
<td>324.10.4.2.85</td>
<td>30950</td>
<td>0.000</td>
<td>11</td>
<td>30902</td>
<td>0.170</td>
<td>30932</td>
</tr>
<tr>
<td>324.10.4.8.0.0</td>
<td>26920</td>
<td>0.000</td>
<td>15</td>
<td>26855</td>
<td>0.140</td>
<td>26883</td>
</tr>
<tr>
<td>324.10.4.8.4.0</td>
<td>28330</td>
<td>0.000</td>
<td>22</td>
<td>28260</td>
<td>0.250</td>
<td>28280</td>
</tr>
<tr>
<td>324.10.5.0.0.0</td>
<td>29680</td>
<td>0.000</td>
<td>6</td>
<td>29638</td>
<td>0.220</td>
<td>29641</td>
</tr>
<tr>
<td>324.20.0.4.0.85</td>
<td>74360</td>
<td>0.000</td>
<td>198</td>
<td>73891</td>
<td>0.830</td>
<td>73807</td>
</tr>
<tr>
<td>324.20.0.4.0.95</td>
<td>42920</td>
<td>0.000</td>
<td>22</td>
<td>42714</td>
<td>0.730</td>
<td>42577</td>
</tr>
<tr>
<td>324.20.0.4.8.0.0</td>
<td>102000</td>
<td>0.000</td>
<td>612</td>
<td>100628</td>
<td>1.920</td>
<td>99576</td>
</tr>
<tr>
<td>324.20.0.4.8.4.0</td>
<td>70720</td>
<td>0.000</td>
<td>55</td>
<td>70368</td>
<td>0.740</td>
<td>70471</td>
</tr>
<tr>
<td>324.20.0.4.8.8.0.0</td>
<td>119470</td>
<td>0.000</td>
<td>10804</td>
<td>118451</td>
<td>0.770</td>
<td>116639</td>
</tr>
<tr>
<td>324.20.0.4.8.8.4.0</td>
<td>90780</td>
<td>0.000</td>
<td>74</td>
<td>90424</td>
<td>0.810</td>
<td>89970</td>
</tr>
<tr>
<td>324.20.0.4.8.8.8.0.0</td>
<td>55400</td>
<td>0.000</td>
<td>52</td>
<td>55096</td>
<td>0.840</td>
<td>54904</td>
</tr>
<tr>
<td>324.20.0.4.8.8.8.4.0</td>
<td>58720</td>
<td>0.000</td>
<td>85</td>
<td>58577</td>
<td>1.020</td>
<td>58009</td>
</tr>
<tr>
<td>324.30.0.4.2.8.0.0</td>
<td>61900</td>
<td>0.000</td>
<td>35</td>
<td>61367</td>
<td>0.830</td>
<td>61545</td>
</tr>
<tr>
<td>324.30.0.4.8.0.0.0</td>
<td>53840</td>
<td>0.000</td>
<td>49</td>
<td>53177</td>
<td>0.630</td>
<td>53300</td>
</tr>
<tr>
<td>324.30.0.4.8.4.0.0</td>
<td>56660</td>
<td>0.000</td>
<td>167</td>
<td>56180</td>
<td>1.340</td>
<td>56216</td>
</tr>
<tr>
<td>324.30.0.4.8.8.0.0.0</td>
<td>59360</td>
<td>0.000</td>
<td>56</td>
<td>59119</td>
<td>0.940</td>
<td>58941</td>
</tr>
</tbody>
</table>

Average  52883.3  0.000  515500  52958.5  0.532  52415.0  52133.1  16.174  52776.5  52376.0  6.455  52880.7  0.003  2507.465  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

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Solution</td>
<td>Gap (%)</td>
<td>Time (s)</td>
<td>Solution</td>
<td>Gap (%)</td>
<td>Time (s)</td>
</tr>
<tr>
<td>818_10_0_05</td>
<td>21460.00</td>
<td>0.000</td>
<td>553</td>
<td>21420.00</td>
<td>4.390</td>
<td>4340.00</td>
</tr>
<tr>
<td>818_10_1_05</td>
<td>35360.00</td>
<td>0.000</td>
<td>653</td>
<td>35356.00</td>
<td>13.050</td>
<td>5040.00</td>
</tr>
<tr>
<td>818_10_2_05</td>
<td>45390.00</td>
<td>0.000</td>
<td>688</td>
<td>45375.00</td>
<td>12.090</td>
<td>4930.00</td>
</tr>
<tr>
<td>818_10_4_20</td>
<td>26920.00</td>
<td>0.000</td>
<td>586</td>
<td>26907.00</td>
<td>11.380</td>
<td>4835.00</td>
</tr>
<tr>
<td>818_10_4_10</td>
<td>28330.00</td>
<td>0.000</td>
<td>639</td>
<td>28309.00</td>
<td>8.630</td>
<td>4738.00</td>
</tr>
<tr>
<td>818_10_5_00</td>
<td>29860.00</td>
<td>0.000</td>
<td>682</td>
<td>29661.00</td>
<td>18.260</td>
<td>4781.00</td>
</tr>
<tr>
<td>818_20_0_05</td>
<td>74360.00</td>
<td>0.000</td>
<td>785</td>
<td>7413.00</td>
<td>75.050</td>
<td>9060.00</td>
</tr>
<tr>
<td>818_20_1_05</td>
<td>42920.00</td>
<td>0.000</td>
<td>562</td>
<td>42703.00</td>
<td>25.670</td>
<td>7620.00</td>
</tr>
<tr>
<td>818_20_2_05</td>
<td>100200*</td>
<td>0.000</td>
<td>1491</td>
<td>101031.00</td>
<td>76.060</td>
<td>8919.00</td>
</tr>
<tr>
<td>818_20_1_05</td>
<td>70720.00</td>
<td>0.000</td>
<td>619</td>
<td>70644.00</td>
<td>37.030</td>
<td>8042.00</td>
</tr>
<tr>
<td>818_20_2_05</td>
<td>119480*</td>
<td>0.000</td>
<td>1579</td>
<td>119397.00</td>
<td>103.480</td>
<td>8918.00</td>
</tr>
<tr>
<td>818_20_4_05</td>
<td>90780.00</td>
<td>0.000</td>
<td>841</td>
<td>90730.00</td>
<td>75.190</td>
<td>84990.00</td>
</tr>
<tr>
<td>818_20_4_10</td>
<td>55400*</td>
<td>0.000</td>
<td>833</td>
<td>55325.00</td>
<td>66.110</td>
<td>88330.00</td>
</tr>
<tr>
<td>818_20_4_15</td>
<td>58720*</td>
<td>0.000</td>
<td>641</td>
<td>58637.00</td>
<td>69.630</td>
<td>87620.00</td>
</tr>
<tr>
<td>818_30_0_05</td>
<td>61900*</td>
<td>0.000</td>
<td>929</td>
<td>61814.00</td>
<td>71.810</td>
<td>90430.00</td>
</tr>
<tr>
<td>818_30_4_05</td>
<td>53840.00</td>
<td>0.000</td>
<td>640</td>
<td>53728.00</td>
<td>22.410</td>
<td>87870.00</td>
</tr>
<tr>
<td>818_30_4_30</td>
<td>56660*</td>
<td>0.000</td>
<td>948</td>
<td>56602.00</td>
<td>34.340</td>
<td>88710.00</td>
</tr>
<tr>
<td>818_30_5_00</td>
<td>59360.00</td>
<td>0.000</td>
<td>877</td>
<td>59269.00</td>
<td>39.840</td>
<td>87040.00</td>
</tr>
<tr>
<td>818_40_0_05</td>
<td>185900</td>
<td>0.000</td>
<td>2056</td>
<td>185426.00</td>
<td>75620.00</td>
<td>364200.00</td>
</tr>
<tr>
<td>818_50_0_15</td>
<td>255000*</td>
<td>0.000</td>
<td>6332</td>
<td>254909.00</td>
<td>739200.00</td>
<td>387370.00</td>
</tr>
<tr>
<td>818_50_0_25</td>
<td>298700</td>
<td>0.000</td>
<td>4435</td>
<td>298217.00</td>
<td>757300.00</td>
<td>387370.00</td>
</tr>
<tr>
<td>818_30_4_30</td>
<td>134600*</td>
<td>0.000</td>
<td>785</td>
<td>134008.00</td>
<td>596110.00</td>
<td>315000.00</td>
</tr>
<tr>
<td>818_30_4_50</td>
<td>141650*</td>
<td>0.000</td>
<td>1719</td>
<td>141281.00</td>
<td>571170.00</td>
<td>315000.00</td>
</tr>
<tr>
<td>818_30_5_00</td>
<td>148400*</td>
<td>0.000</td>
<td>1946</td>
<td>148393.00</td>
<td>666700.00</td>
<td>315000.00</td>
</tr>
<tr>
<td>Average</td>
<td>91563.8</td>
<td>0.000</td>
<td>1362.417</td>
<td>91402.4.00</td>
<td>201.940</td>
<td>91213.00</td>
</tr>
</tbody>
</table>

* indicates new best known solution obtained by an exact method

1 heuristic gap
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 sub-problems 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.

References


