Hindawi
Complexity
Volume 2019, Article ID 9723718, 29 pages
https://doi.org/10.1155/2019/9723718

Research Article

Facility Location with Tree Topology and Radial Distance Constraints

Pablo Adasme [1] **and Ali Dehghan Firoozabadi** [2]

[1]*Department of Electrical Engineering, Universidad de Santiago de Chile, Avenida Ecuador 3519, Santiago, Chile*
[2]*Department of Electricity, Universidad Tecnológica Metropolitana, Av. Jose Pedro Alessandri 1242, 7800002 Santiago, Chile*

Correspondence should be addressed to Pablo Adasme; pablo.adasme@usach.cl

Received 22 April 2019; Accepted 30 October 2019; Published 21 November 2019

Academic Editor: Dan Selişteanu

Let $G_d = (V, E_d)$ be an input disk graph with a set of facility nodes V and a set of edges E_d connecting facilities in V. In this paper, we minimize the total connection cost distances between a set of customers and a subset of facility nodes $S \subseteq V$ and among facilities in S, subject to the condition that nodes in S simultaneously form a spanning tree and an independent set according to graphs $\overline{G}_d$ and G_d, respectively, where $\overline{G}_d$ is the complement of G_d. Four compact polynomial formulations are proposed based on classical and set covering p-Median formulations. However, the tree to be formed with S is modelled with Miller–Tucker–Zemlin (MTZ) and path orienteering constraints. Example domains where the proposed models can be applied include complex wireless and wired network communications, warehouse facility location, electrical power systems, water supply networks, and transportation networks, to name a few. The proposed models are further strengthened with clique valid inequalities which can be obtained in polynomial time for disk graphs. Finally, we propose Kruskal-based heuristics and metaheuristics based on guided local search and simulated annealing strategies. Our numerical results indicate that only the MTZ constrained models allow obtaining optimal solutions for instances with up to 200 nodes and 1000 users. In particular, tight lower bounds are obtained with all linear relaxations, e.g., less than 6% for most of the instances compared to the optimal solutions. In general, the MTZ constrained models outperform path orienteering ones. However, the proposed heuristics and metaheuristics allow obtaining near-optimal solutions in significantly short CPU time and tight feasible solutions for large instances of the problem.

1. Introduction

Facility location has been a major research topic within last decades [1–6]. In general, a facility location problem is mainly concerned with the optimal placement of facilities in a predefined geographical area in such a way that customers (users) can connect to facilities at minimum (distance) costs. Example application domains include wireless and wired network communications, warehouse facility location, electrical power systems, water supply networks, and transportation networks, to name a few [1]. Naturally, the optimal placement is also conditioned to several factors which directly depend on the application itself, e.g., avoiding placing hazardous materials near housing, fair distribution of hospitals, public schools, military bases, radar installations, branch banks, shopping centers, waste disposal facilities, police and/or fire departments in a city to meet user demands and avoiding interference between base stations in a wireless communication network. Notice that all of these examples require facilities to be placed separately, which naturally imposes the condition of a radial distance constraint. In principle, any of them can be represented by means of a unit disk graph, and then an independent set structure appears as a natural way to model these constraints. In particular, any disk graph representation need not be a connected graph.

Let $G_d = (V, E_d)$ and $G_c = (V, E_c)$ be, respectively, the input disk and complete graphs composed of a set of facility nodes V drawn in the Euclidean plane with edge sets E_d and E_c, and let S be a subset of V. In this paper, we consider the

problem of assigning a set K of users to facility nodes in S while simultaneously forming a tree backbone $T = (S, E_c(T))$ with nodes in S where $E_c(T)$ denotes the set of edges in E_c spanned by T such that $|E_c(T)| = |S| - 1$ and with the additional condition that no two adjacent nodes in V related with G_d can belong to S. The latter represents the radial distance requirement that we handle with independent set constraints [7]. Notice that finding the optimal subset S leads to a hard combinatorial optimization problem. For this purpose, we assume that all candidate sites for facility placement are represented by means of a unit disk graph with some predefined distance facility radial coverage. If two facility nodes are located near a certain distance, then at most, one of them can be selected for the tree backbone. Finally, users can connect to the resulting subset S. Notice that forming a tree backbone with facilities is an efficient strategy to ensure connectivity in a network [5, 8–12]. A unit disk graph is the resulting graph obtained by intersecting several unit disks in the Euclidean plane where each vertex corresponds to a center of each disk and with edges connecting them whenever the corresponding vertices lie within a constant (unit) distance of each other.

However, an independent or stable set of an arbitrary graph $G = G(V, E)$ with set of nodes V and set of edges E is a subset of vertices $S \subseteq V$, no two of which are adjacent. This means that, for every two vertices in S, there is no edge connecting them. Notice that a maximal independent set is a subset $S \subseteq V$ in which by adding any other vertex forces set S to loose the independent set property. However, a maximum independent set is a maximal independent set with largest cardinality among all maximal independent sets. Consequently, every maximum independent set is maximal although the converse does not necessarily hold. Finding the maximum independent set is an NP-hard optimization problem, and the number of maximal independent sets in a general arbitrary graph with n nodes is of the order of $O(3^{n/3})$ [13]. Furthermore, notice that finding a maximal independent set in an arbitrary graph G is equivalent to finding a maximal clique in its complement graph $\overline{G}$. The reader must not be confused with the fact that if it is possible to list all maximal cliques of G in polynomial time, then listing all maximal cliques is also possible to achieve in its complement graph in polynomial time. This is not true in general. In particular, for unit disk graphs, all maximal cliques can be listed in polynomial time [14, 15]. However, this is not possible to achieve with its complement graph which is no longer a unit disk graph. In fact, finding the maximum independent set in a unit disk graph has proved to be an NP-complete problem even if a disk representation of the graph is given [16–18]. In general, listing all maximal cliques in polynomial time can be achieved for sparse graphs [15, 19]. The latter can be performed, for example, by using the Bron–Kerbosch algorithm [14, 19].

We propose four compact polynomial formulations based on classical and set covering p-Median models in order to minimize the total connection cost distance between facilities and between customers and facilities simultaneously [10, 20, 21]. However, the tree to be formed with S is modelled with Miller–Tucker–Zemlin (MTZ) and path orienteering constraints, respectively [22, 23]. Notice that path orienteering constraints have been recently proposed in [23] to avoid cycles in trees. Subsequently, we further strengthen the MILP models by adding clique valid inequalities which we obtain in polynomial time using the Bron–Kerbosch algorithm [14, 15]. Notice that the fact that we can list all maximal cliques in unit disk graphs allows us to strengthen our proposed models in a straightforward manner, leading to tight gaps when compared to the optimal solutions. Finally, we propose Kruskal-based heuristics and metaheuristics adapted from a greedy approach proposed for the maximum independent set problem [24, 25]. The greedy algorithm mainly consists of selecting the vertex of minimum (or maximum) degree of G and removing it together with its neighbors from the graph, and it iterates on this process on the remaining graph until no vertex remains. Notice that the output of this greedy algorithm always results in a maximal independent set. In particular, the metaheuristics are constructed based on guided local search and simulated annealing strategies [26–29]. As far as we know, placing facilities at a certain radial distance using independent sets on unit disk graphs and with tree backbone representations have never been considered in the literature before. Similarly, the solution methods are new to the proposed models.

The paper is organized as follows: In Section 2, we review some related works. Then, in Section 3, we introduce the mathematical formulations together with their strengthened versions and present two examples of feasible solutions for the problem. In Section 4, we present our proposed heuristics and metaheuristics. Subsequently, in Section 5, we conduct substantial numerical experiments in order to compare all the proposed models and algorithms. Finally, in Section 6, we give the main conclusions of the paper and provide some insights into future research.

2. Related Work

Facility location dates back to the well-known Fermat point problem which consists of finding a point such that the total distance to three other points in an Euclidean plane is minimized. This problem was originally proposed by Pierre de Fermat and later solved by Evangelista Torricelli [4, 6]. Today, its extended version is known as the Geometric Median Problem and it is considered as a continuous facility location problem [1, 2]. Discrete facility location has also been considered as a major research topic since several decades ago [3, 5], and in particular, the distance requirement between facilities or between facilities and demand points has been recognized as a key decision factor. For instance, in [5], the authors illustrate and highlight the distance requirement with several real-life applications. Their examples include distribution systems where transshipment among stocking points are

of significant size, locating emergency facilities such as ambulance stations and fire fighting units with respect to population centers, gasoline stations, fast food restaurants, and spacing between wireless radio base stations, construction of nuclear power plants or missile silos nearby, and dump sites for chemical waste or collected refuse, to name a few.

For a more general and deep review related with facility location topic, the reader is referred to [1–6, 30–32] and references therein. In particular, Farahani et al. [3] present a recent survey related with distance covering location problems together with a comprehensive review of models, solution approaches, and applications where it also highlights the importance of the distance factor required when designing facility networks.

Hereafter, we briefly discuss some works where the authors propose models which are closer to ours. Perhaps, a first model that is worth mentioning is the covering tour problem [33]. This problem consists of determining a minimum length Hamiltonian cycle which has to be formed with a subset of nodes from a larger set of nodes V such that every vertex of an another set W, that must be covered, is within a specified distance from the cycle. This problem is similar in nature to our problem in the sense that it requires to form a backbone network topology with a subset of facilities. In order to solve this problem, the authors proposed an exact branch and cut algorithm and a heuristic. Similarly, covering location problems related with paths and trees are proposed in [34], where the facilities are assumed to be small in size in proportion to their location regions which is the case of subway metro stations and highways for example and where the aim is to find a path through the facility network of minimum length such that the population coverage is maximized. In [34], the authors develop Lagrangian solution approaches to solve their models. The maximal covering location problem (MCLP) is also worth mentioning where the objective is to maximize the amount of demand covered within an acceptable service distance by locating a fixed number of new facilities [35]. The problem is solved with linear programming relaxations and branch and bound techniques as well. Variants of MCLP are also presented and discussed in [3].

The indirect covering tree problem proposed in [36] represents another approach which is similar in nature to our facility location problem. There, it is assumed that a given backbone spanning tree network is given a priori, and then, two optimization problems are derived, namely, the minimum cost covering subtree and the maximal indirect covering subtree problems. In the former, the aim is to find the minimum cost collection of arcs that form a subtree and satisfy covering constraints for nodes of the network, whereas in the latter, the subtree maximizes the demand within a given distance of nodes of the subtree. Reduction techniques that were initially proposed to solve the location set covering problem are extended by the authors to solve these new problems. More variants of these problems are also explained and discussed in [3, 36].

In [37], the authors study two continuous facility location problems. The first one consists of placing a fixed number of unit disks in an area in order to maximize the total weight of the points inside the union of the disks, whilst the second one deals with placing a single disk with center in a given constant region in order to minimize the total weight of the points inside the disk. The authors propose approximation algorithms to obtain solutions for these problems. Similarly, the authors in [12] deal with a continuous facility location problem while including backbone connectivity together with their related costs. In particular, the objective here is to minimize the weighted sum of three costs, namely, the fixed costs required to install facilities, the backbone network costs required to connect facilities, and the transportation costs incurred from providing services from the facilities to the service region. The authors analyze the behaviour of their proposed model and derive two asymptotically optimal configurations of facilities. Finally, they give a fast constant factor approximation algorithm which allows us to find the placement of a set of facilities in a convex polygon that minimizes the total sum of the costs. We note that the authors also highlight the importance of considering connectivity among facilities in this reference. Finally, in the domain of telecommunications, some related works, although not so similar, can be consulted for instance in references [30–32].

3. Description of the Problem and Mathematical Formulations

In this section, we first describe the facility location problem we are dealing with. For this purpose, we present and explain two feasible solutions of the problem for different radial coverage values. Subsequently, we present our MILP formulations.

3.1. Description of the Problem. Consider the undirected graphs $G_d = (V, E_d)$ and $G_c = (V, E_c)$ as defined in Section 1. Notice that E_c contains the edges obtained with all pairs of nodes v_1 and v_2 in V, $(v_1 \neq v_2)$ as G_c is complete. However, E_d is composed of the edges obtained with all pairs of nodes v_1 and v_2 in V, $(v_1 \neq v_2)$ which are located at a radial distance lower than or equal to d. The facility location problem we are dealing with consists of finding a subset of facilities $S \subseteq V$ such that S is an independent set of G_d and a tree backbone of G_c and where each user in the set K is assigned to its nearest facility in S, thus minimizing the total distance costs between facilities and users and among facilities themselves. Notice that S does not necessarily need to be a maximal independent set.

In Figure 1, we plot two examples of input disk graphs with $|V| = 30$ nodes within an area of one square km using radial coverage values of 0.2 and 0.5 km, respectively. More precisely, on the left, we plot each input disk graph G_d and the independent set obtained from G_d. Nodes in S are colored green. However, on the right, we

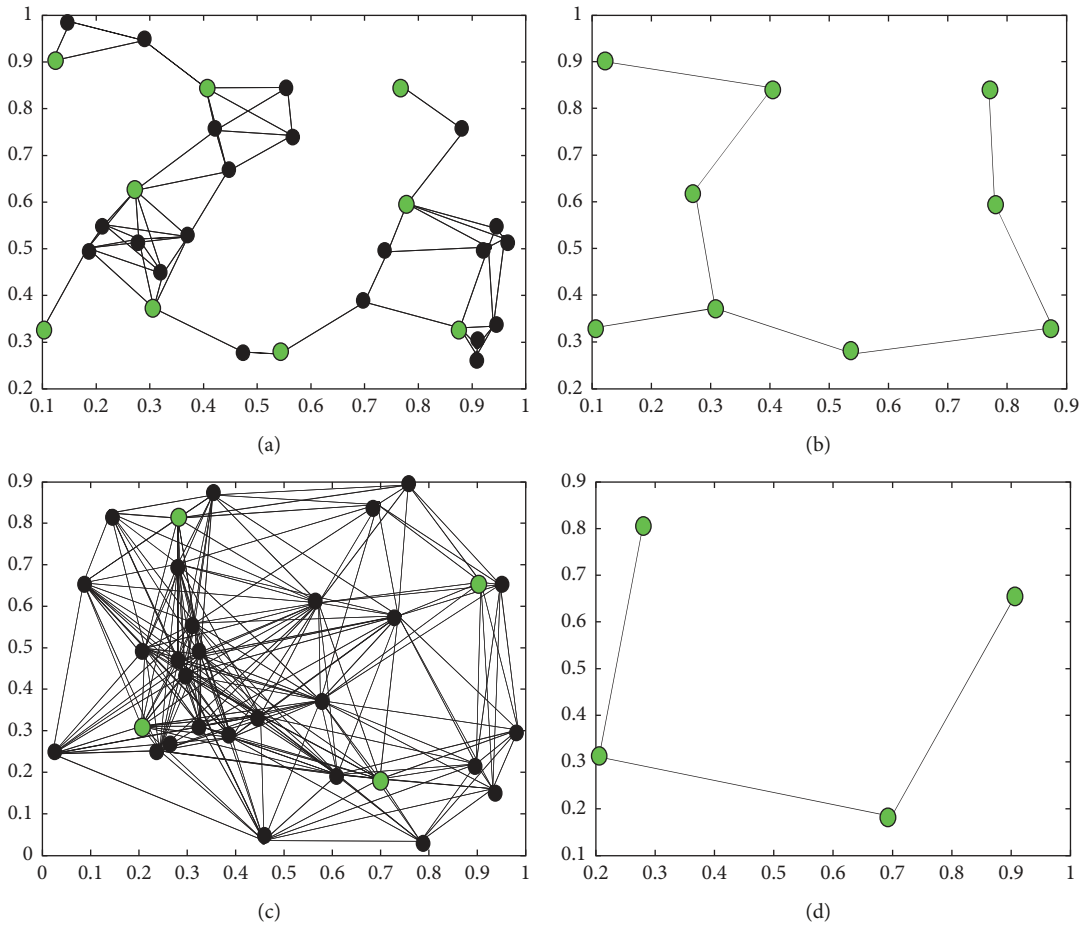


Figure 1: Input graphs and feasible solutions obtained for an instance with $|V| = 30$ nodes and $|K| = 50$ users using radial coverage values of 0.2 and 0.5 km: (a) independent set using 0.2 km; (b) minimum spanning tree backbone using 0.2 km; (c) independent set using 0.5 km; (d) minimum spanning tree backbone using 0.5 km.

plot the spanning tree backbone formed with nodes in S. Users are omitted for the sake of clarity. In particular, notice that, for an input graph G_d with low density, the cardinality of S should be higher. On the opposite, if the density of G_d is high, the cardinality of S should be smaller. By density, we refer to the ratio obtained by dividing the number of edges of E_d over the total number of edges of E_c.

3.2. MILP Formulations. In order to write a first compact polynomial formulation for the facility location problem described in Section 3.1, we first define set $A_c = \{(i, j), (j, i)/\{i, j\} \in E_c\}$ as the set of directed arcs obtained from edges in E_c and $H = (V, A_c)$ as the digraph obtained from G_c. Next, we define the binary variables:

$$x_{kj} = \begin{cases} 1, & \text{if user } k \in K \text{ is connected to facility } j \in S \subseteq V, \\ 0, & \text{otherwise,} \end{cases}$$

$$z_{ij} = \begin{cases} 1, & \text{if the arc } (i, j) \in A_c, i, j \in S \subseteq V \text{ is part of the tree backbone,} \\ 0, & \text{otherwise,} \end{cases} \quad (1)$$

$$y_j = \begin{cases} 1, & \text{if facility } j \in V \text{ belongs to } S \subseteq V, \\ 0, & \text{otherwise.} \end{cases}$$

Thus, a MILP model can be verified by means of the following proposition.

Proposition 1. *The linear model P_1 allows us to obtain a feasible solution with minimum cost for the*

facility location problem while satisfying the following conditions:

(i) *All users in K are assigned to facilities in S.*

(ii) *Subset $S \subseteq V$ forms an independent set (not necessarily maximal) from G_d.*

(iii) *Subset $S \subseteq V$ forms a tree backbone $T = (S, E_c(T))$ from G_c where $E_c(T) \subseteq E_c$ is the set of edges spanned by T where $|E_c(T)| = |S| - 1$:*

$$P_1: \min_{\{x,y,z,u\}} \sum_{k\in K}\sum_{j\in V} C_{kj}x_{kj} + \sum_{i\in V}\sum_{\substack{j\in V\\(i\neq j)}} D_{ij}z_{ij}, \tag{2}$$

$$\text{s.t.} \quad \sum_{j\in V} x_{kj} = 1, \quad \forall k \in K, \tag{3}$$

$$x_{kj} \le y_j, \quad \forall k \in K, j \in V, \tag{4}$$

$$y_i + y_j \le 1, \quad \forall \{i, j\} \in E_d, \tag{5}$$

$$\sum_{\substack{i,j\in V\\(i\neq j)}} z_{ij} = \sum_{j\in V} y_j - 1, \tag{6}$$

$$u_i - u_j \ge 1 - |V|\left(1 - z_{ji}\right), \quad \forall i, j \in V, (i \neq j), \tag{7}$$

$$u_j \le |V| y_j, \quad \forall j \in V, \tag{8}$$

$$u_j \ge y_j, \quad \forall j \in V, \tag{9}$$

$$\sum_{\substack{i,j\in V\\(i\neq j)}} z_{ij} \le y_j, \quad \forall j \in V, \tag{10}$$

$$\sum_{\substack{i\in V\\(i\neq j)}} z_{ij} + \sum_{\substack{i\in V\\(i\neq j)}} z_{ji} \le (|V| - 1) y_j, \quad \forall j \in V, \tag{11}$$

$$x_{kj}, y_j, z_{ij} \in \{0, 1\}, \quad \forall k \in K, i, j \in V. \tag{12}$$

Proof. To prove (i) and (ii), we focus on explaining how the set of constraints in P_1 together with its objective function imply each of the assertions. For this purpose, we define the input matrices $C = (C_{kj})$ and $D = (D_{ij})$, $\forall k \in K$, $i, j \in V$ $(i \neq j)$, to be the distances between user k and facility j and between facilities i and j, respectively. In particular, we require matrix $D = (D_{ij})$ to be symmetric. Thus, the total distance is minimized in (2). It is easy to see from the objective function (2) that the first double sum will be less expensive when the cardinality of S, which can be computed by $|S| = \sum_{j\in V} y_j$, is larger. However, the second term will be cheaper when the cardinality of S is smaller. Consequently, we seek for a tradeoff between these two terms leading to an optimal number of facilities in S which can be obtained by solving P_1. Notice that S is composed of the nodes $j \in V$ for which the binary variable $y_j = 1$. Constraints (3) ensure that each user is connected to a unique facility, whilst constraints (4) ensure that a user is connected to a facility node that belongs to S. Next, constraints (5) are independent set constraints and ensure that no two adjacent facilities in V can belong to S according to E_d. Furthermore, note that S is not conditioned to be a maximal independent set.

To prove condition (iii), we show that any feasible solution (x, y, z, u) of P_1 is an acyclic connected arborescence of $H = (V, A_c)$. First, notice that the constraint (6) imposes the condition that any tree formed with nodes in S contains exactly $|S| - 1$ arcs. However, the constraints (10) ensure that the number of arcs entering node $j \in V$ is at most one if and only if $y_j = 1$. Similarly, the constraints (11) ensure that the number of incoming and outgoing arcs in node $j \in V$ can be at most $|V| - 1$ if and only if $y_j = 1$. Otherwise, if $y_j = 0$, then the binary variables $z_{ij} = z_{ji} = 0$, $\forall i, j \in V$ $(i \neq j)$. Finally, the constraints (7)–(9) together with the constraints (6) and (10) avoid cycles. To see how these constraints avoid cycles, let us assume that the subset $S = \{v_1, v_2, \ldots, v_{|S|}\} \subseteq V$ is solution to P_1. Then, the constraints (7) imply that $u_{v_1} + 1 \le u_{v_2}, u_{v_2} + 1 \le u_{v_3}, \ldots, u_{v_{|S|-1}} + 1 \le u_{v_{|S|}}, u_{v_{|S|}} + 1 \le u_{v_1}$ since $1 \le u_j \le |S|, \forall j \in S$. In turn, this implies that $u_{v_1} < u_{v_2} < u_{v_3} < \ldots < u_{v_{|S|-1}} < u_{v_{|S|}} < u_{v_1}$ which cannot hold. Notice that, in general, the contradiction holds for any subset of nodes of S which implies that the digraph induced by $z = (z_{ij})$ cannot contain directed cycles. Furthermore, notice that constraints (7) do not consider the particular case when an arc is in the opposite direction in the sequence $(v_1, v_2), (v_2, v_3), \ldots, (v_j, v_1)$ for some $j \in S$. This particular case is avoided by means of the constraints (6) and (10), which finally ensure that the digraph is connected with $|S| - 1$ directed arcs while avoiding the fact that a particular node has more than one incoming arc. Consequently, by dropping the directions of each arc, we obtain an undirected tree topology. Recall that the cost distance matrix $D = (D_{ij})$ $\forall i, j \in V$, $(i \neq j)$, is symmetric. Finally, the constraints (12) limit the domain of the decision variables, thus concluding the proof. □

Notice that P_1 is constructed based on a classical formulation of the p-Median problem and a directed Miller–Tucker–Zemlin characterization of the spanning tree polytope [10, 21, 22].

Regarding the complexity of the facility location problem associated with P_1, we outline the following theorem.

Theorem 1. *The facility location problem associated with P_1 is NP-hard.*

Proof. To see this, let us consider the following weighted version of the objective function (2) as

$$(2-\alpha)\sum_{k\in K}\sum_{j\in V}C_{kj}x_{kj}+\alpha\sum_{i\in V}\sum_{\substack{j\in V\\(i\neq j)}}D_{ij}z_{ij}, \tag{13}$$

where $0\leq\alpha\leq 2$. In particular, when $\alpha=0$, any spanning tree will be feasible for P_1 at zero cost for the objective function of the problem. Consequently, in this case, P_1 can be equivalently written as the following optimization problem:

$$\begin{aligned} P_1^{\alpha}:\ \min_{\{x,y\}}\quad & 2\sum_{k\in K}\sum_{j\in V}C_{kj}x_{kj}+\mathscr{M}\sum_{\{i,j\}\in E_d}y_iy_j,\\ \text{s.t.}\quad & \sum_{j\in V}x_{kj}=1,\quad \forall k\in K,\\ & x_{kj}\leq y_j,\quad \forall k\in K, j\in V,\\ & x_{kj},y_j\in\{0,1\},\quad \forall k\in K, j\in V, \end{aligned} \tag{14}$$

where the second quadratic sum in (14) is due to an equivalent form of writing independent set constraints [38]. For this purpose, parameter $\mathscr{M}$ should be a nonnegative real value such that $\mathscr{M}>2\sum_{k\in K}\sum_{j\in V}C_{kj}$. In general, notice that any independent set S implies $\sum_{\{i,j\}\in E_d}y_iy_j=0$. Consequently, it is easy to see that P_1^{α} is a quadratic version of the classical uncapacitated facility location problem which is NP-hard [39].

Subsequently, when $0\leq\alpha<2$, we aim to solve P_1^{α} with the implicit requirement that the number of facilities must be between $1\leq\sum_{j\in V}y_j\leq\beta(G_d)$, where $\beta(G_d)$ denotes the cardinality of the maximum independent set of G_d. Consequently, in this case, the problem is equivalent to finding a minimum spanning tree with variable number of nodes in an arbitrary graph which is also known to be an NP-hard optimization problem by reduction from the Steiner tree problem [40]. Finally, when $\alpha=2$, the objective function (13) equals zero and the solution can be found trivially in linear time. In this case, the cardinality of the optimal solution set S^* will be equal to one and all users will be assigned to this unique facility node. Obviously, the tree obtained in this case does not contain any edge, and hence, its objective value equals zero as well. □

Notice that, in particular, the objective function of P_1 is obtained with $\alpha=1$ which is set by default. The reason behind this is that we minimize the total cost distances with fair degree of importance for both users and facilities. However, below in the numerical result section, we further consider the more general case for values of $\alpha\in[0;2]$ which reflects different cost structures.

An alternative formulation of P_1 can be obtained by removing constraints (7) and (9) together with the decision variables u_j for each $j\in V$ and by introducing the nonnegative precedence variables p_{ij} for all $i,j\in V\ (i\neq j)$. Precedence variables are used to indicate if there exists a direct path from i to j. If such a path exists, then $p_{ij}=1$; otherwise, $p_{ij}=0$. Notice that precedence variables have recently been introduced in order to deal with subtour elimination constraints in combinatorial optimization problems defined under a tree topology structure [23]. Thus, an equivalent MILP model can be written as

$$\begin{aligned} P_2:\ \min_{\{x,y,z,p\}}\quad & \sum_{k\in K}\sum_{j\in V}C_{kj}x_{kj}+\sum_{i\in V}\sum_{\substack{j\in V\\(i\neq j)}}D_{ij}z_{ij},\\ \text{s.t.}\quad & \sum_{j\in V}x_{kj}=1,\quad \forall k\in K,\\ & x_{kj}\leq y_j,\quad \forall k\in K, j\in V,\\ & y_i+y_j\leq 1,\quad \forall\{i,j\}\in E_d,\\ & \sum_{\substack{i,j\in V\\(i\neq j)}}z_{ij}=\sum_{j\in V}y_j-1, \end{aligned} \tag{15}$$

$$z_{ij}\leq p_{ij},\quad \forall i,j\in V,(i\neq j), \tag{16}$$

$$p_{ij}+p_{ji}\leq 1,\quad \forall i,j\in V,(i\neq j), \tag{17}$$

$$p_{ij}+z_{jk}\leq 1+p_{ik},\quad \forall j,k\in V,\ (j\neq k), i\in V-\{j,k\}, \tag{18}$$

$$p_{ik}+z_{jk}\leq 1+p_{ij},\quad \forall j,k\in V,\ (j\neq k), i\in V-\{j,k\}, \tag{19}$$

$$\sum_{\substack{i\in V\\(i\neq j)}}z_{ij}\leq y_j,\quad \forall j\in V, \tag{20}$$

$$\sum_{\substack{i\in V\\(i\neq j)}}z_{ij}+\sum_{\substack{i\in V\\(i\neq j)}}z_{ji}\leq(|V|-1)y_j,\quad \forall j\in V, \tag{21}$$

$$\begin{aligned} & x_{kj},y_j,z_{ij}\in\{0,1\},\quad \forall k\in K, i,j\in V,\\ & p_{ij}\geq 0,\quad \forall i,j\in V,(i\neq j). \end{aligned} \tag{22}$$

Constraints (16) state that if there is a path from i to j, for all $i,j\in V\ (i\neq j)$ that belongs to the solution, then a direct path going from i to j must exist. Similarly, constraints (17) ensure that there should be at most one path either from i to j or from j to i for all $i,j\in V,\ (i\neq j)$. Finally, constraints (18) and (19) ensure that if there is a direct path from i to j and the arc path j to k belongs to the solution, then both path variables P_{ij} and P_{ik} must be equal.

Notice that both P_1 and P_2 are formulated based on a classical formulation of the p-Median problem while using MTZ and path orienteering constraints, respectively. Next, we present two formulations which are based on a set cover formulation of the p-Median problem [20, 41] together with MTZ and path orienteering constraints as well. We denote these two models hereafter by P_3 and P_4, respectively. In order to write a first model under this approach, we first construct for each $k\in K$, the vectors $C_k^o=(C_{k1}^o,C_{k2}^o,\ldots,C_{k|V|}^o)$ by sorting in the ascending order

the different entries of each k-th row of the original cost matrix $C = (C_{kj})$ in (2) in such a way that $C^o_{k1} = 0$ and $C^o_{k|V|} = \max\{C_{k,j}, j \in V\}$. Since the first entry in each row of these vectors equals zero, we need to introduce an artificial facility node labeled as "1" hereafter. Also, we need to expand the input graphs G_d and G_c with this additional node. Thus, we expand sets V, E_d, and E_c and denote them by V^o, E^o_d, and E^o_c, respectively. Since the artificial node has to be excluded from the solution of the problem, the extra edges added to E^o_d and E^o_c are irrelevant for the solution of the problem. Similarly, we construct a new matrix $D^o = (D^o_{ij})$, $\forall i, j \in V^o\ (i \neq j)$, by adding to D an additional row and a column in position one with zero entries. Consequently, a first set covering-based formulation using MTZ constraints can be written as

$$P_3: \min_{\{x,y,u,z\}} \quad \sum_{k\in K} \sum_{\substack{j\in V^o \\ (j>1)}} \left(C^o_{kj} - C^o_{k,j-1}\right)x_{kj} + \sum_{i\in V^o} \sum_{\substack{j\in V^o \\ (i\neq j)}} D^o_{ij} z_{ij}, \tag{23}$$

$$\text{s.t.} \quad x_{kj} + \sum_{\substack{i\in V^o \\ \{C_{ki}<C^o_{kj}\}}} y_i \geq 1, \quad \forall k \in K, j \in V^o, (j>1), \tag{24}$$

$$\begin{aligned}
& y_1 = 0, \\
& y_i + y_j \leq 1, \quad \forall \{i,j\} \in E^o_d, \\
& \sum_{\substack{i,j\in V^o \\ (i\neq j)}} z_{ij} = \sum_{j\in V^o} y_j - 1, \\
& u_i - u_j \geq 1 - |V|\left(1 - z_{ji}\right), \quad \forall i, j \in V^o, (i \neq j), \\
& u_j \leq |V| y_j, \quad \forall j \in V^o, \\
& u_j \geq y_j, \quad \forall j \in V^o, \\
& \sum_{\substack{i\in V^o \\ (i\neq j)}} z_{ij} \leq y_j, \quad \forall j \in V^o, \\
& \sum_{\substack{i\in V^o \\ (i\neq j)}} z_{ij} + \sum_{\substack{i\in V^o \\ (i\neq j)}} z_{ji} \leq (|V|-1) y_j, \quad \forall j \in V^o, \\
& x_{kj}, y_j, z_{ij} \in \{0,1\}, \quad \forall k \in K, i, j \in V^o.
\end{aligned} \tag{25}$$

In P_3, notice that the binary variable x_{kj} for each $k \in K$, $j \in V^o$, acts as a cumulative variable where $x_{kj} = 1$ if and only if the allocation cost distance of user k is at least C^o_{kj} and $x_{kj} = 0$ otherwise. For this purpose, the set covering constraints (24) ensure that variable $x_{kj} = 1$ if there is no open facility at less than distance C^o_{kj}. The positive coefficients of the first double sum in the objective function (23) ensure that variable $x_{kj} = 0$ otherwise. Next, the constraint (25) ensures that node "1" is excluded from the solution of the problem. Finally, all remaining constraints have the same effect as for P_1. Analogously as for P_2, we propose the following MILP formulation using path orienteering constraints:

$$\begin{aligned}
P_4: \min_{\{x,y,z,p\}} \quad & \sum_{k\in K} \sum_{\substack{j\in V^o \\ (j>1)}} \left(C^o_{kj} - C^o_{k,j-1}\right)x_{kj} + \sum_{i\in V^o} \sum_{\substack{j\in V^o \\ (i\neq j)}} D^o_{ij} z_{ij}, \\
\text{s.t.} \quad & x_{kj} + \sum_{\substack{i\in V^o \\ \{C_{ki}<C^o_{kj}\}}} y_i \geq 1, \quad \forall k \in K, j \in V^o, (j>1), \\
& y_1 = 0, \\
& y_i + y_j \leq 1, \quad \forall \{i,j\} \in E^o_d, \\
& \sum_{\substack{i,j\in V^o \\ (i\neq j)}} z_{ij} = \sum_{j\in V^o} y_j - 1, \\
& z_{ij} \leq p_{ij}, \quad \forall i, j \in V^o, (i \neq j), \\
& p_{ij} + p_{ji} \leq 1, \quad \forall i, j \in V^o, (i \neq j), \\
& p_{ij} + z_{jk} \leq 1 + p_{ik}, \quad \forall j,k \in V^o, (j \neq k), i \in V^o - \{j,k\}, \\
& p_{ik} + z_{jk} \leq 1 + p_{ij}, \quad \forall j,k \in V^o, (j \neq k), i \in V^o - \{j,k\}, \\
& \sum_{\substack{i\in V^o \\ (i\neq j)}} z_{ij} \leq y_j, \quad \forall j \in V^o, \\
& \sum_{\substack{i\in V^o \\ (i\neq j)}} z_{ij} + \sum_{\substack{i\in V^o \\ (i\neq j)}} z_{ji} \leq (|V|-1) y_j, \quad \forall j \in V^o, \\
& x_{kj}, y_j, z_{ij} \in \{0,1\}, \quad \forall k \in K, i, j \in V^o, \\
& \qquad p_{ij} \geq 0, \quad \forall i, j \in V^o, (i \neq j).
\end{aligned} \tag{26}$$

Ultimately, we derive strengthened versions for each of the proposed models and denote them hereafter by P^s_1, P^s_2, P^s_3, and P^s_4, respectively. In particular, from the MTZ constrained models, we remove constraints (5), (7), and (11) and replace them by the following sets of constraints:

$$\sum_{j\in V} \mathrm{MC}(q,j) y_j \leq 1, \quad \forall q \in Q, \tag{27}$$

$$u_i - u_j + (|V|-1) z_{ij} + (|V|-3) z_{ji} \leq |V| - 2, \quad \forall i, j \in V,\ (i \neq j), \tag{28}$$

$$z_{ij} \leq y_j, \quad \forall i, j \in V,\ (i \neq j), \tag{29}$$

$$z_{ji} \leq y_j, \quad \forall i, j \in V,\ (i \neq j), \tag{30}$$

where the constraints (27) avoid cliques in the solution set S. For this purpose, we define the set Q containing all maximal cliques of G_d. Recall that, in practice, all cliques can be obtained in polynomial time as G_d is a unit disk graph [14, 15]. Consequently, each row in the binary matrix $\mathrm{MC}(q,\cdot)$ corresponds to a maximal clique $q \in Q$. Constraints (28) are valid cuts obtained from constraints (7) and adapted for the spanning tree polytope [10, 22]. Finally, constraints (29) and (30) are valid cuts for the constraints (11) [11].

From the path orienteering-based models, we only remove constraints (5) and (11), and replace them with (27)

and (29) and (30), respectively. Hereafter, we denote the linear programming (LP) relaxations of P_1, P_2, P_3, P_4, P_1^s, P_2^s, P_3^s, and P_4^s by LP_1, LP_2, LP_3, LP_4, LP_1^s, LP_2^s, LP_3^s, and LP_4^s, respectively. □

4. Algorithms

In this section, we propose Kruskal-based heuristics and metaheuristics which are adapted from a greedy approach initially proposed for the maximum independent set problem [25]. In particular, the metaheuristics are constructed based on guided local search and simulated annealing greedy strategies [26–29].

4.1. Heuristic Approach. Three variants of the same heuristic are proposed. The only difference between them relies on the greedy decision of adding new nodes to S at each iteration. Thus, we only present the first one referred to as GMIN in Algorithm 1 and explain the other ones based on these steps. The heuristic requires the input matrices $C = (C_{kj})$, $D = (D_{ij})$, $k \in K$, $i, j \in V$, $(i \neq j)$, and the input disk graph $G_d = (V, E_d)$. Then, for each vertex $v \in V$, the following steps are performed. The set S is initialized as an empty set, and W is assigned set V. The set W is used as an auxiliary set. Next, v is added to S, and it is removed from W together with its neighbors. Subsequently, we obtain the minimum spanning tree with nodes in S using the Kruskal algorithm and assign each user $k \in K$ to its nearest facility in S. Finally, we compute the objective function value of P_1 and save the current solution if its objective value is less than the best found so far. Initially, the objective value is set to infinity. Then, in Step 2, we enter into a while loop and iterate until $G_d = (W, E_d(W))$ is empty, where $E_d(W)$ denotes the set of edges in E_d induced by remaining nodes in W. Inside the while loop, we proceed similarly by selecting a new vertex $v \in W$ with minimum or maximum degree to be added to S or by interchanging between minimum and maximum degrees. This is the greedy decision which differentiates remaining variants of Algorithm 1. Hereafter, we denote the three heuristics by GMIN, GMAX, and IC, respectively. Finally, Algorithm 1 returns the best solution found and its objective function value.

4.2. Guided Local Search Approach. The first metaheuristic we propose is based on a guided local search strategy and requires the same input parameters of Algorithm 1 [29]. Its pseudocode is presented in Algorithm 2, and it is explained as follows. In Step 1, first we set $W = V$ and $S = \varnothing$, and iterate while $G_d = (W, E_d(W))$ is not empty. The following steps are performed inside the loop. First, we remove the node with minimum degree v from W and add it to S. Then, we remove all neighbors of v from W and find the minimum spanning tree with current nodes in S using the Kruskal algorithm. Finally, we assign each user $k \in K$ to its nearest facility in S, compute the objective function value of P_1, and save $S^* = S$ if the objective value is less than the best found so far. Similarly, in Step 2, we enter into a second, while loop and iterate until the CPU time is greater than or equal to maxCPU which is an input parameter. Inside this loop, we penalize each value in vector $\mathrm{Deg}(S^*)$ corresponding to each node in S^* by adding a random number uniformly distributed in $(0, \rho)$. Initially, $\mathrm{Deg}(v)$ contains the degree of each node $v \in V$. Notice that this step acts as a guided local search strategy, and its main purpose is to avoid repeated nodes in the new solutions generated by the algorithm. Then, we reset $S = \varnothing$ and $W = V$ and construct a new independent set S from $G_d = (W, E_d(W))$ with cardinality less than or equal to R. Observe that, in this case, S is constructed without evaluating the objective function each time a new vertex is added to S. Once set S is constructed, we proceed to find the minimum spanning tree formed with nodes in S using the Kruskal algorithm, assign each user $k \in K$ to its nearest facility in S, and compute the objective function value of P_1. If this value is less than the best found so far, we update $S^* = S$ and reset the current CPU time to zero. Finally, we randomly generate a value $\gamma \in \{-1, 0, 1\}$ in order to decrease, maintain, or increase, respectively, the cardinality of the next independent set to be generated by the algorithm starting from $|S^*|$. Notice that Algorithm 2 updates degree values in $\mathrm{Deg}(\cdot)$ according to new and better solutions obtained following a simple local search rule in contrast to classical guided local search metaheuristics which use augmented objective functions in order to perform the local search [29].

4.3. Simulated Annealing Approach. Next, we further consider another metaheuristic approach based on a simulated annealing (SA) greedy strategy. SA is a classical metaheuristic which was proved to be highly efficient when finding near-optimal solutions for hard combinatorial optimization problems [26–28]. SA randomly searches for ascent moves in order to escape from local minima and to find global optimal solutions. It is basically inspired on the annealing process of a material in metallurgy and consists of heating and cooling steps which must be controlled in such a way that certain equilibrium is reached in terms of temperature. In particular, the cooling step is analog to a slow decrease in the probability of accepting worse solutions in order to allow a major degree of diversity while performing the search process. The underlying idea in the SA algorithm is simple and can be described as follows. First, SA requires an initial feasible solution to start. Next, it randomly generates a neighbor solution. If this neighbor solution is better than the best found so far, it is accepted as a new best solution. Otherwise, SA allows us to move to a worse solution with a certain probability. In fact, this is the key ingredient that allows us to escape from local minima. In general, the probability of moving to a new solution is determined by Boltzmann distribution while varying the temperature which is gradually decreased. Initially, it is high enough to allow a high degree of diversity since probabilities are close to one. However, small temperature values allow probabilities to go down to zero which is equivalent to a pure local search method. The SA procedure we propose is depicted in Algorithm 3, and it is explained as follows. It requires the same parameters used by Algorithms 1 and 2. Initially, the

Data: input matrices $C = (C_{kj})$ and $D = (D_{ij})$ and input graph $G_d = (V, E_d)$.
Result: a feasible solution for P_1.
for each $v \in V$ **do**
  Step 1:;
  Set $S = \varnothing$, $W = V$;
  $S = S \cup \{v\}$, $W = W \setminus \{v\}$;
  Remove all neighbors of v from W;
  Find the minimum spanning tree formed with nodes in S using the Kruskal algorithm;
  Assign each user $k \in K$ to its nearest facility in S and compute the objective function value of P_1;
  Save the current solution found if its objective function value is less than the best found so far;
  Step 2:;
  while ($G_d = (W, E_d(W))$ *is not empty*) **do**
    Let v be the vertex with minimum degree in $G_d = (W, E_d(W))$;
    $S = S \cup \{v\}$, $W = W \setminus \{v\}$;
    Remove all neighbors of v from W;
    Find the minimum spanning tree formed with nodes in S using the Kruskal algorithm;
    Assign each user $k \in K$ to its nearest facility in S and compute the objective function value of P_1;
    Save the current solution found if its objective function value is less than the best found so far;
Return the best solution found and its objective function value;

Algorithm 1: Proposed heuristics.

Data: input matrices $C = (C_{kj})$ and $D = (D_{ij})$ and input graph $G_d = (V, E_d)$.
Result: a feasible solution for P_1.
Step 1:;
Set $S = \varnothing$ and $W = V$;
while ($G_d = (W, E_d(W))$ *is not empty*) **do**
  Let v be the vertex with minimum degree in $G_d = (W, E_d(W))$;
  $S = S \cup \{v\}$ and $W = W \setminus \{v\}$;
  Remove all neighbors of v from W;
  Find the minimum spanning tree formed with nodes in S using the Kruskal algorithm;
  Assign each user $k \in K$ to its nearest facility in S and compute the objective function value of P_1;
  Save the current solution found in S^* if its objective function value is less than the best found so far;
$R = |S^*|$;
Step 2:;
while (cpuTime $\leq$ maxCPU) **do**
  Update degree vector $\mathrm{Deg}(\cdot)$ according to the indices of each element in S^* as $\mathrm{Deg}(S^*) = \mathrm{Deg}(S^*) + (0, \rho)$;
  Set $S = \varnothing$ and $W = V$;
  while ($G_d = (W, E_d(W))$ *is not empty and* ($|S| < R$)) **do**
    Let v be the vertex with minimum value in $\mathrm{Deg}(W)$;
    $S = S \cup \{v\}$ and $W = W \setminus \{v\}$;
    Remove all neighbors of v from W;
  Find the minimum spanning tree formed with nodes in S using the Kruskal algorithm;
  Assign each user $k \in K$ to its nearest facility in S and compute the objective function value of P_1;
  Save the current solution found in S^* if its objective function value is less than the best found so far. If a better solution is found, set cpuTime = 0; $R = |S^*|$;
  If ($|S^*| = 1$) **then**
    $R = |S^*| + |\gamma|$;
  else
    $R = |S^*| + \gamma$;
Return best solution found and its objective function value;

Algorithm 2: Guided local search-based metaheuristics.

temperature T^0 is set to a large positive value T^i. Inside the while loop of Step 2, we set $S = \varnothing$ and $W = V$, and for each $v \in S^*$, which is obtained in step one, we pick randomly a number from $[0, 1]$ and compare it with the input parameter η which is arbitrarily chosen from the interval $[0, 1]$ as well. If the random number is less than η, then we include node $v \in S^*$ in the new independent set S and remove from W node v and its neighbors. Otherwise, we skip node v and continue. This allows us to keep an η percentage of nodes in S^*. Next, we start a while loop in order to add new nodes in

the solution until $G_d = (W, E_d(W))$ is empty or $|S| = R$. Added nodes are chosen randomly from remaining nodes in W. Subsequently, we find the minimum spanning tree with nodes in S, assign each user $k \in K$ to its nearest facility in S, and compute the objective function value of P_1. If a better solution is found, we save this new solution in S^{**} and set $S^* = S^{**}$ and the current CPU time to zero in order to extend the duration of the algorithm. Otherwise, we compute the variable Δ as the difference between the current objective function value minus the previous one, generate a random number r from $[0, 1]$, and compute the probability $p = \exp(-\Delta/T^0)$. If r is less than p, we accept the new solution and save it in S^*; else, we set $S^* = S^{**}$. Notice that the value of Δ will be positive when the objective value of the current solution is greater than the previous one, i.e., when the new solution is worse. In particular, when the initial temperature T^0 is large enough and the difference in Δ is low, the probability p will be close to one. On the opposite, when temperature values decrease and Δ is positive, the probability p will be close to zero and the algorithm will behave as a pure random local search algorithm. The next lines inside the while loop of Step 2 ensures that T^0 decreases at a δ rate. Then, we check if $T^0 < \epsilon$ where ϵ is a small positive value. If so, we reset $T^0 = T^i$ and allow the algorithm to interchange between exploration and exploitation more frequently during the process. Finally, we randomly generate a value $\gamma \in \{-1, 0, 1\}$ to decrease, maintain, or increase, respectively, the size of the next independent set to be generated by the algorithm. Notice that the value of T^0 is of critical importance to handle efficiently the algorithm. If T^0 decreases rapidly, then a premature convergence to a local minimum may occur. On the opposite, if T^0 decreases slowly, then the algorithm will converge slowly as well. In order to guarantee that the algorithm converges to global optimal solutions with probability one, T^0 should be decreased at the logarithmic rate [42]. However, this can lead to poor convergence rates, so in practical settings, one is usually forced to decrease $T^0(t)$ at iteration t by $T^0(t) = \delta * T^0(t-1)$ where $0.85 \leq \delta \leq 0.96$ [43].

5. Numerical Experiments

In this section, we perform substantial numerical experiments in order to compare all the proposed models and algorithms. For this purpose, we randomly generate connected Euclidean disk graph instances. More precisely, nodes are uniformly distributed in a plane within an area of one square kilometer, and each pair of nodes is connected if the distance of the edge connecting them is less than or equal to the predefined radial coverage. However, the complete version is obtained by connecting all pairs of nodes. Consequently, each entry in the input matrices $C = (C_{kj})$ and $D = (D_{ij})$ for all $k \in K$, $i, j \in V$, represents the distance between user k and facility j and distance between facilities i and j, $(i \neq j)$, respectively. Matrix D is symmetric. We generate two sets of instances with dimensions of $|K| = \{500, 600, 700, 800, 1000\}$ users and $|V| = \{50, 100, 200, 300\}$ facility nodes for radial coverage values of 0.2 and 0.5 km. Each set is composed of 20 instances. Then, we further generate two larger sets of instances with dimensions of $|K| = \{1000, 1200, 1300, 1400, 1500\}$ users and $|V| = \{500, 700, 1000\}$ nodes for both radial coverage values of 0.2 and 0.5 km as well. The larger sets are composed of 15 instances each one and are only solved with the proposed algorithms since the linear models cannot be solved with CPLEX due to shortage of memory [44]. In order to solve the strengthened versions of each MILP model, we find all cliques for each disk graph using the Bron–Kerbosch algorithm [14, 19].

We implement a Matlab program using CPLEX 12.8 to solve all the proposed models P_1, P_1^s, LP_1, LP_1^s, P_2, P_2^s, LP_2, LP_2^s, P_3, P_3^s, LP_3, LP_3^s, P_4, P_4^s, LP_4, and LP_4^s [44]. The proposed algorithms GMIN, GMAX, IC, GL, and SA are also implemented in Matlab, where GL refers to Algorithm 2 and SA refers to Algorithm 3, respectively. The parameters in Algorithms 2 and 3 are calibrated to maxCPU = 100 s, $\eta = 0.5$, $\delta = 0.85$, $\epsilon = 10^{-4}$, $T^i = 1000$, and $\gamma \in \{-1, 0, 1\}$. The numerical experiments have been carried out on an Intel(R) 64 bits core (TM) with 3.40 GHz and 8 gigabytes of RAM.

5.1. Numerical Results for the Linear Models. In Tables 1–4, we present numerical results for the linear models. In these tables, we solve the first two sets of instances which use both radial coverage values. Thus, each row in each of these tables corresponds to a particular instance. Notice that, in Tables 2–4, we do not report numerical results for some of the instances because the linear models cannot be solved with CPLEX due to shortage of memory [44]. The column information in these four tables is exactly the same with the exception of Table 1 which includes four additional columns with instance dimensions. The information of each column is described as follows. In column 1, we present the instance number. Then, only in Table 1, columns 2–5 report the number of users, the number of facility nodes, the density of each input graph, and the number of optimal facilities obtained with P_1. The density of each graph is computed by dividing the number of edges of the input disk graph over the total number of edges. Next, in columns 6–10 of Table 1, which correspond to columns 2–6 in Tables 2–4, we present the optimal solution or best solution found with CPLEX in two hours for the MILP models, number of branch and bound nodes, CPU time in seconds, optimal solutions for the LP relaxations, and CPU time in seconds, respectively. Similarly, in columns 11–15 of Table 1 which correspond to columns 7–11 in Tables 2–4, respectively, we present the same column information for the strengthened versions of the MILP models. Finally, the last two columns in Tables 2–4 report gaps that we compute by $((\mathrm{Opt} - \mathrm{Opt}_{\mathrm{LP}})/\mathrm{Opt}) * 100$ where Opt refers to the optimal solution of each MILP model and $\mathrm{Opt}_{\mathrm{LP}}$ represents the optimal solution obtained with its linear relaxation.

From Table 1, we observe that the density of the input disk graphs increases with the radial coverage. On the opposite, we see that the number of optimal facilities obtained with P_1 is larger for the sparse graphs, which is obvious since sparse graphs contain more independent sets than dense graphs. In particular, we observe that, for the instances

Data: input matrices $C = (C_{kj})$ and $D = (D_{ij})$ and input graph $G_d = (V, E_d)$.
Result: a feasible solution for P_1.
Step 1:;
Set $S = \varnothing$ and $W = V$;
while loop as in Step 1 of Algorithm 2;
$R = |S^*|$ and $T^0 = T^i$;
Step 2:;
while (cpuTime $\leq$ maxCPU) **do**
  Set $S = \varnothing$ and $W = V$;
  for each ($v \in S^*$) **do**
    Draw a random number r from $[0, 1]$;
    if ($r < \eta$) **then**
      $S = S \cup \{v\}$ and $W = W \backslash \{v\}$;
      Remove all neighbors of v from W;
  while ($G_d = (W, E_d(W))$ *is not empty and* ($|S| < R$)) **do**
    Chose a vertex v randomly from W;
    $S = S \cup \{v\}$ and $W = W \backslash \{v\}$;
    Remove all neighbors of v from W;
  Find the minimum spanning tree formed with nodes in S using the Kruskal algorithm;
  Assign each user $k \in K$ to its nearest facility in S and compute the objective function value of P_1;
  if (*A better solution is found*) **then**
    Save it in S^{**} and set cpuTime = 0 and $S^* = S^{**}$;
  else
    Compute Δ as the difference between the current objective value minus the previous one;
    Draw a random number r from $[0, 1]$;
    Compute $p = \exp(-\Delta/T^0)$;
    if ($r < p$) **then**
      Accept the new solution and save it in S^*;
    else
      $S^* = S^{**}$;
  $R = |S^*|$ and $T^0 = T^0 * \delta$;
  if ($T^0 < \epsilon$) **then**
    $T^0 = T^i$;
  if ($|S^*| = 1$) **then**
    $R = |S^*| + |\gamma|$;
  else
    $R = |S^*| + \gamma$;
Return best solution found and its objective function value;

Algorithm 3: Simulated annealing-based metaheuristic.

#16–#20, CPLEX cannot solve the MILP models to optimality within two hours of CPU time using a radial coverage of 0.2 km, which is also the case for the instances #17–#20 using a radial coverage of 0.5 km. All remaining instances are solved to optimality. Next, we observe that sparse graphs are harder to solve to optimality which is reflected in terms of branch and bound nodes for example. In particular, these values and the CPU times are lower for the strengthened models. Regarding the optimal solutions obtained with the LP relaxations, we see that these bounds are significantly tighter for the strengthened models as well. This can be confirmed by the gap columns which show that the linear relaxations are less than 6% for most of the instances when compared to the optimal solutions. However, the CPU times for the strengthened linear relaxations are considerably higher.

In Table 2, first we observe that the optimal solutions obtained with P_2 and P_2^s are exactly the same as those obtained with P_1 and P_1^s in Table 1. In general, we observe that path orienteering constraints decrease CPLEX performance significantly. Consequently, we could only solve instances #1–#10 in this case. This observation is also valid for the LP relaxations which require more CPU times than for solving LP_1 and LP_1^s, respectively. Regarding the number of branch and bound nodes, we observe a slightly less number of nodes required to solve the MILP models in Table 2. Finally, we observe that the gaps are exactly the same in both Tables 1 and 2.

In Tables 3 and 4, we report numerical results obtained with P_3, P_3^s, P_4, and P_4^s, respectively. As it can be observed, in these tables, we cannot solve more than 15 and 10 instances with CPLEX in each table, respectively. From the obtained results, we observe that the optimal solutions are the same as in Tables 1 and 2. Similarly, we obtain the same gap values for the LP bounds when compared to the optimal solutions. In general, we observe that the strengthened models allow us to obtain optimal solutions more rapidly and verify that path orienteering constraints deteriorate CPLEX performance.

Table 1: Numerical results for P_1 and P_1^s.

#	$\|K\|$	$\|V\|$	Den.	$\|S^*\|$	P_1	B&Bn	CPU (s)	LP_1	CPU (s)	P_1^s	B&Bn	CPU (s)	LP_1^s	CPU (s)	Gap_1(%)	Gap_1^s(%)
Instances using radial coverage of 0.2 km																
1	500	50	11.26	14	56.20	113	9.61	47.63	0.78	56.20	83	6.60	55.03	1.79	15.25	2.08
2	600	50	10.61	17	63.83	123	10.33	54.63	0.98	63.83	12	5.59	63.26	1.65	14.41	0.89
3	700	50	11.02	14	84.28	166	15.46	73.40	1.14	84.28	13	8.44	83.15	2.64	12.91	1.35
4	800	50	11.02	16	85.13	65	11.81	75.25	1.39	85.13	89	8.99	84.79	3.07	11.61	0.40
5	1000	50	10.86	16	105.24	573	29.09	90.41	1.90	105.24	142	16.47	104.14	3.90	14.09	1.04
6	500	100	10.53	20	49.81	387	24.65	35.85	1.51	49.81	277	22.79	47.97	7.57	28.03	3.69
7	600	100	10.40	21	56.95	307	47.41	40.21	1.97	56.95	130	27.83	55.16	9.19	29.39	3.14
8	700	100	10.24	20	67.01	4402	260.79	47.29	2.28	67.01	5547	236.14	64.46	15.02	29.43	3.80
9	800	100	10.59	20	76.62	633	94.77	56.29	2.61	76.62	176	54.16	74.49	17.74	26.53	2.78
10	1000	100	10.20	20	93.87	225	91.34	68.47	3.79	93.87	159	73.87	91.93	24.74	27.05	2.06
11	500	200	10.19	23	45.05	2331	727.94	26.17	4.41	45.05	219	406.80	41.67	107.66	41.91	7.50
12	600	200	9.97	25	52.38	599	405.91	29.89	4.95	52.38	273	257.96	49.50	135.94	42.93	5.50
13	700	200	10.20	23	62.07	1898	884.70	35.45	5.73	62.07	175	640.04	58.50	193.37	42.90	5.75
14	800	200	10.47	22	70.70	4269	4296.99	38.52	6.58	70.70	346	986.03	66.93	246.15	45.51	5.33
15	1000	200	10.08	23	87.50	2441	1653.07	49.17	9.86	87.50	1226	1324.35	83.87	358.33	43.81	4.16
16	1000	300	10.35	24	85.16	222	7200	41.25	18.95	84.99	101	7200	-	1008.92	51.56	-
17	1200	300	10.81	25	102.77	127	7200	48.09	23.99	102.25	51	7200	*	*	53.20	*
18	1300	300	11.01	21	119.96	90	7200	54.21	27.86	633.94	8	7200	*	*	54.81	*
19	1400	300	10.70	24	119.61	127	7200	55.51	29.75	117.08	30	7200	*	*	53.59	*
20	1500	300	10.59	1	766.47	11	7200	60.52	31.06	765.22	2	7200	*	*	92.10	*
Instances using radial coverage of 0.5 km																
1	500	50	51.18	5	99.83	15	16.74	46.17	0.83	99.83	3	13.12	96.61	3.87	53.75	3.23
2	600	50	44.08	4	122.36	0	14.12	62.68	0.97	122.36	0	11.43	121.36	4.87	48.78	0.82
3	700	50	53.39	4	138.68	0	17.04	70.69	1.59	138.68	0	15.35	138.03	5.96	49.03	0.47
4	800	50	46.12	4	163.48	38	29.89	77.24	1.36	163.48	11	24.80	158.58	8.98	52.75	2.99
5	1000	50	51.92	5	201.94	0	29.91	98.40	2.33	201.94	0	26.66	201.22	13.82	51.27	0.35
6	500	100	48.91	5	97.20	0	65.99	35.74	1.72	97.20	0	60.15	96.14	21.78	63.23	1.09
7	600	100	49.33	5	116.65	40	101.18	44.57	2.04	116.65	0	98.19	113.68	30.03	61.79	2.54
8	700	100	43.54	6	131.50	0	106.99	46.58	2.53	131.50	0	102.31	130.60	45.29	64.58	0.69
9	800	100	47.64	5	157.34	15	154.19	59.78	2.98	157.34	3	148.20	154.90	50.81	62.00	1.55
10	1000	100	52.61	5	192.27	27	231.91	68.04	3.65	192.27	7	222.71	189.23	88.52	64.61	1.58
11	500	200	48.30	6	93.32	0	419.36	26.51	4.90	93.32	0	522.18	91.83	249.68	71.59	1.60
12	600	200	47.85	6	110.63	37	777.41	30.98	5.58	110.63	33	740.69	108.48	401.20	72.00	1.94
13	700	200	49.38	6	133.28	55	1158.32	35.24	7.60	133.28	19	1061.54	129.97	548.92	73.56	2.48
14	800	200	47.90	6	152.25	39	1273.34	41.22	10.16	152.25	4	1174.40	149.29	640.60	72.93	1.94
15	1000	200	47.48	6	188.15	36	2104.82	49.17	12.39	188.15	13	1664.89	182.83	1005.08	73.87	2.83
16	1000	300	46.43	6	184.02	91	6892.81	40.91	23.32	184.02	18	5364.37	*	*	77.77	*
17	1200	300	48.88	1	551.27	0	7200	49.02	24.77	219.90	17	6747.17	*	*	91.11	*
18	1300	300	47.10	1	768.33	0	7200	53.42	38.84	237.04	0	6044.67	*	*	93.05	*
19	1400	300	47.21	1	701.78	0	7200	56.48	43.06	701.78	0	7200	*	*	91.95	*
20	1500	300	48.17	1	728.30	0	7200	61.02	53.02	728.30	0	7200	*	*	91.62	*

*Instance not solved. -, CPLEX could not solve the problem.

Finally, from the numerical results presented in Tables 1–4, we conclude that model P_1^s outperforms the set covering formulations. In general, we observe that MTZ constraints seem to work better than path orienteering ones.

In Tables 5 and 6, for each input disk graph instance in Table 1, we present the total number of maximal cliques, maximum clique number, total number of maximal independent sets, and maximum independent set number, respectively. We compute this information by using the Bron–Kerbosch algorithm [14, 15]. Recall that the total number of cliques can be obtained in polynomial time for disk graphs. Consequently, the maximum clique number can be obtained straightforwardly by selecting the maximal clique with largest cardinality. However, the total number of maximal independent sets can be obtained with the Bron–Kerbosch algorithm listing all maximal cliques from its complement graph $\overline{G}_d = (V, \overline{E}_d)$, where $\overline{E}_d = E_c - E_d$. Notice that $\overline{G}_d$ is no longer a disk graph, and therefore, we cannot expect to find all cliques in polynomial time in this case. In fact, the total number of maximal independent sets in an arbitrary graph with n nodes is of the order of $O(3^{n/3})$ [13]. Moreover, finding the maximum independent set is an NP-hard optimization problem. Although it is a difficult task to list all maximal independent sets, we still run the Bron–Kerbosch algorithm for two hours in order to give some insights with respect to the difficulty of doing this for graph instances with different radial coverage values. As it can be observed from Table 5, while using a radial coverage of 0.2 km, we can find all maximal cliques. On the opposite, we cannot list all maximal independent sets for any of the

Table 2: Numerical results for P_2 and P_2^s.

#	P_2	B&Bn	CPU (s)	LP_2	CPU (s)	P_2^s	B&Bn	CPU (s)	LP_2^s	CPU (s)	Gap_2 (%)	Gap_2^s (%)
Instances using radial coverage of 0.2 km												
1	56.20	40	26.85	47.63	2.65	56.20	31	21.54	55.03	3.23	15.25	2.08
2	63.83	14	23.85	54.63	2.75	63.83	32	25.66	63.26	3.10	14.41	0.89
3	84.28	14	30.59	73.40	2.98	84.28	18	26.55	83.15	3.34	12.91	1.35
4	85.13	15	28.61	75.25	2.84	85.13	18	20.73	84.79	3.62	11.61	0.40
5	105.24	93	41.59	90.41	3.24	105.24	108	34.16	104.14	4.24	14.09	1.04
6	49.81	73	212.35	35.85	18.36	49.81	86	164.74	47.97	34.63	28.03	3.69
7	56.95	65	218.46	40.21	18.53	56.95	73	196.75	55.16	35.30	29.39	3.14
8	67.01	1663	1570.02	47.29	18.84	67.01	1335	1376.37	64.46	41.54	29.43	3.80
9	76.62	59	273.33	56.29	18.91	76.62	60	242.44	74.49	39.23	26.53	2.78
10	93.87	64	334.26	68.47	19.39	93.87	42	251.46	91.93	41.20	27.05	2.06
Instances using radial coverage of 0.5 km												
1	99.83	0	42.78	46.17	2.46	99.83	3	26.86	96.61	4.30	53.75	3.23
2	122.36	0	25.35	62.68	2.73	122.36	0	23.17	121.36	5.12	48.78	0.82
3	138.68	0	31.58	70.69	2.82	138.68	0	28.64	138.03	5.76	49.03	0.47
4	163.48	17	57.44	77.24	2.90	163.48	11	51.17	158.58	7.41	52.75	2.99
5	201.94	0	69.19	98.40	3.23	201.94	0	40.25	201.22	12.45	51.27	0.35
6	97.20	0	278.55	35.74	18.59	97.20	0	249.74	96.14	50.62	63.23	1.09
7	116.65	10	361.80	44.57	18.50	116.65	5	329.04	113.68	39.34	61.79	2.54
8	131.50	0	340.63	46.58	18.38	131.50	0	315.00	130.60	64.91	64.58	0.69
9	157.34	4	442.54	59.78	19.33	157.34	4	439.66	154.90	72.87	62.00	1.55
10	192.27	11	597.43	68.04	19.67	192.27	13	563.11	189.23	83.19	64.61	1.58

Table 3: Numerical results for P_3 and P_3^s.

#	P_3	B&Bn	CPU (s)	LP_3	CPU (s)	P_3^s	B&Bn	CPU (s)	LP_3^s	CPU (s)	Gap_3 (%)	Gap_3^s (%)
Instances using radial coverage of 0.2 km												
1	56.20	141	5.21	47.63	1.59	56.20	85	2.98	55.03	2.45	15.25	2.08
2	63.83	89	4.63	54.63	1.89	63.83	116	4.38	63.26	2.89	14.41	0.89
3	84.28	126	5.65	73.40	2.20	84.28	23	3.67	83.15	3.65	12.91	1.35
4	85.13	125	5.37	75.25	2.60	85.13	30	3.53	84.79	4.71	11.61	0.40
5	105.24	843	12.73	90.41	3.42	105.24	692	11.42	104.14	6.40	14.09	1.04
6	49.81	330	42.95	35.85	5.35	49.81	348	45.86	47.97	16.47	28.03	3.69
7	56.95	390	59.47	40.21	6.52	56.95	169	24.29	55.16	22.76	29.39	3.14
8	67.01	10184	246.96	47.29	7.16	67.01	5419	177.14	64.46	28.75	29.43	3.80
9	76.62	725	78.08	56.29	9.00	76.62	69	31.65	74.49	33.03	26.53	2.78
10	93.87	295	73.12	68.47	11.45	93.87	212	57.27	91.93	45.97	27.05	2.06
11	45.05	2900	1584.04	26.17	22.50	45.05	253	558.45	41.67	158.00	41.91	7.50
12	52.38	924	917.80	29.89	25.71	52.38	341	284.30	49.50	228.23	42.93	5.50
13	62.07	1727	2136.49	35.45	31.61	62.07	230	693.26	58.50	291.16	42.90	5.75
14	70.70	2559	4483.00	38.52	35.13	70.70	421	1071.89	66.90	1017.85	45.51	5.38
15	87.50	2928	2845.94	49.17	45.18	87.50	2269	2128.26	83.87	577.61	43.81	4.16
Instances using radial coverage of 0.5 km												
1	99.83	18	4.96	46.17	1.54	99.83	3	5.16	96.61	9.52	53.75	3.23
2	122.36	0	3.56	62.68	1.87	122.36	0	3.48	121.36	12.11	48.78	0.82
3	138.68	0	4.13	70.69	2.34	138.68	0	4.06	138.03	15.29	49.03	0.47
4	163.48	29	6.76	77.24	2.56	163.48	9	7.07	158.58	20.02	52.75	2.99
5	201.94	0	5.62	98.40	3.35	201.94	0	5.21	201.22	33.18	51.27	0.35
6	97.20	0	61.82	35.74	5.45	97.20	0	52.59	96.14	94.86	63.23	1.09
7	116.65	19	85.66	44.57	6.40	116.65	10	93.71	113.68	120.26	61.79	2.54
8	131.50	0	92.90	46.58	7.29	131.50	0	63.85	130.60	152.85	64.58	0.69
9	157.34	14	104.94	59.78	9.25	157.34	7	136.47	154.90	195.45	62.00	1.55
10	192.27	16	119.76	68.04	11.64	192.27	19	157.03	189.23	292.41	64.61	1.58
11	93.32	0	818.46	26.51	22.57	93.32	0	812.49	86.04	1011.80	71.59	7.81
12	110.63	45	1288.58	30.98	29.36	110.63	22	1332.43	83.82	1013.69	72.00	24.24
13	133.28	35	2712.52	35.24	33.77	133.28	18	2076.74	101.78	1016.23	73.56	23.63
14	152.25	43	2016.52	41.22	36.77	152.25	3	1777.64	121.81	1017.84	72.93	19.99
15	188.15	52	4170.99	49.17	43.91	188.15	13	3638.41	128.55	1059.40	73.87	31.67

Table 4: Numerical results for P_4 and P_4^s.

#	P_4	B&Bn	CPU (s)	LP_4	CPU (s)	P_4^s	B&Bn	CPU (s)	LP_4^s	CPU (s)	Gap_4 (%)	Gap_4^s (%)
Instances using radial coverage of 0.2 km												
1	56.20	23	17.55	47.63	3.43	56.20	19	12.84	55.03	4.96	15.25	2.08
2	63.83	19	20.64	54.63	3.34	63.83	18	12.87	63.26	4.18	14.41	0.89
3	84.28	36	19.55	73.40	4.16	84.28	38	18.47	83.15	5.19	12.91	1.35
4	85.13	53	23.65	75.25	3.87	85.13	22	12.04	84.79	4.45	11.61	0.40
5	105.24	173	32.46	90.41	4.46	105.24	108	24.52	104.14	5.14	14.09	1.04
6	49.81	75	245.39	35.85	21.84	49.81	74	210.43	47.97	44.49	28.03	3.69
7	56.95	87	292.41	40.21	22.49	56.95	54	154.28	55.16	52.54	29.39	3.14
8	67.01	1423	1086.13	47.29	23.96	67.01	1482	1132.80	64.46	64.96	29.43	3.80
9	76.62	52	278.22	56.29	23.20	76.62	76	255.91	74.49	39.22	26.53	2.78
10	93.87	61	303.92	68.47	28.55	93.87	48	233.59	91.93	92.79	27.05	2.06
Instances using radial coverage of 0.5 km												
1	99.83	5	13.99	46.17	3.14	99.83	0	11.76	96.61	4.76	53.75	3.23
2	122.36	0	12.36	62.68	3.59	122.36	0	11.98	121.36	5.19	48.78	0.82
3	138.68	0	10.86	70.69	3.70	138.68	0	10.98	138.03	6.13	49.03	0.47
4	163.48	21	16.66	77.24	4.20	163.48	9	17.10	158.58	20.14	52.75	2.99
5	201.94	0	12.87	98.40	4.70	201.94	0	12.95	201.22	9.73	51.27	0.35
6	97.20	0	308.21	35.74	22.25	97.20	0	211.75	96.14	98.75	63.23	1.09
7	116.65	15	342.09	44.57	22.49	116.65	6	265.36	113.68	101.10	61.79	2.54
8	131.50	0	302.74	46.58	23.84	131.50	0	223.71	130.60	208.23	64.58	0.69
9	157.34	9	331.83	59.78	25.56	157.34	4	339.99	154.90	185.62	62.00	1.55
10	192.27	14	386.21	68.04	27.69	192.27	7	374.94	189.23	293.45	64.61	1.58

Table 5: Maximal cliques and maximal independent sets obtained with the Bron–Kerbosch algorithm for the instances #1–#20 in Table 1 using a radial coverage value of 0.2 km.

Instance number #	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
# Maximal C.	36	41	35	30	38	92	108	97	96	105	313	283	305	334	309	687	674	811	739	654
Maximum C.	9	7	6	11	7	11	12	9	9	9	14	15	14	18	15	21	23	23	21	21
# Maximal I.S.	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Maximum I.S.	16	17	15	16	16	20	21	20	21	21	24	26	24	24	24	25	26	26	26	26

-, not solved in 2 hours.

instances in two hours. Consequently, in this case, we obtained the maximum independent set number by solving the following optimization problem:

$$\begin{aligned} \text{(MIS):}\ \max_{\{y\}} & \sum_{j\in V} y_j, \\ \text{s.t.}\quad & y_i + y_j \le 1, \quad \forall \{i,j\} \in E_d, \\ & y_j \in \{0,1\}, \forall j \in V. \end{aligned} \tag{31}$$

Similarly, from Table 6, while using a radial coverage of 0.5 km, we can find all maximal cliques and all maximal independent sets for half of the instances (e.g., instances #1–#10). This clearly shows that the sparser the input disk graphs are, the harder it is to list its maximal independent sets. We also see from Tables 5 and 6 that the maximum independent set numbers are slightly larger than the optimal number of facilities obtained with P_1 in Table 1. Notice that the maximum independent set number is an upper bound for the optimal number of facilities since no larger maximal set can be obtained from G_d. This observation can be understood by simply looking at the objective function of the proposed P_1 model which seeks for a tradeoff between the two terms. In particular, when the number of facilities is close to the maximum independent set number, it means that the first double sum in the objective function of P_1 dominates the second term, which also means that the cost of assigning the total number of users to the facilities is significantly larger than the cost of the spanning tree required to connect facilities. Notice that the trade off between the two terms can vary with parameter $\alpha \in [0, 2]$ used in Theorem 1. Numerical results while varying parameter α are discussed and reported below.

5.2. Numerical Results for the Algorithms. In Tables 7–9, we report numerical results for the proposed heuristics and metaheuristics for all the instances reported in Table 1 and for the larger sets of instances mentioned above. More precisely, numerical results for the instances in Table 1 are reported in Tables 7 and 8, while Table 9 reports numerical results for the larger sets. In Table 7, for the sake of clarity, we repeat some information of Table 1.

Thus, in columns 1–5, we present the instance number, number of users to be assigned to the facilities, number of nodes of the input graph, and optimal solution or minimum

Table 6: Maximal cliques and maximal independent sets obtained with the Bron–Kerbosch algorithm for the instances #1–#20 in Table 1 using a radial coverage value of 0.5 km.

Instance number #	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
# Maximal C.	77	35	65	56	72	293	220	249	270	264	1801	1978	1418	1057	1491	4940	5829	5570	5552	6529
Maximum C.	17	19	19	15	16	34	31	26	30	35	54	54	57	60	52	73	76	77	74	72
# Maximal I.S.	2066	2684	1559	1903	1678	30617	39721	60324	36088	27438	-	-	-	-	-	-	-	-	-	-
Maximum I.S.	5	5	5	6	5	6	6	6	5	6	6	6	6	6	6	6	6	6	6	7

-, not solved in 2 hours.

Table 7: Numerical results for GMIN, GMAX, and IC for the instances in Table 1.

#	$\|K\|$	$\|V\|$	Opt	CPU (s)	GMIN				GMAX				*IC*			
					$\|S^*\|$	Ub	CPU (s)	Gap (%)	$\|S^*\|$	Ub	CPU (s)	Gap (%)	$\|S^*\|$	Ub	CPU (s)	Gap (%)
Instances using radial coverage of 0.2 km																
1	500	50	56.20	6.60	16	58.71	1.63	4.46	14	63.02	1.29	12.13	15	63.03	1.34	12.15
2	600	50	63.83	5.59	17	63.92	2.11	0.14	14	72.97	1.46	14.32	15	70.82	1.51	10.95
3	700	50	84.28	8.44	15	84.67	2.01	0.46	11	100.05	1.29	18.70	12	94.11	1.47	11.66
4	800	50	85.13	8.99	16	89.62	2.48	5.27	12	105.11	1.40	23.47	12	102.91	1.67	20.89
5	1000	50	105.24	16.47	16	107.64	2.99	2.28	13	117.36	2.17	11.52	13	115.74	2.33	9.98
6	500	100	49.81	22.79	20	50.82	4.78	2.04	15	58.86	2.70	18.17	15	57.69	2.97	15.82
7	600	100	56.95	27.83	21	60.00	5.75	5.34	14	70.28	3.22	23.39	16	65.53	3.76	15.06
8	700	100	67.01	236.14	20	71.41	6.30	6.57	15	77.72	3.81	15.99	16	80.02	4.14	19.43
9	800	100	76.62	54.16	21	78.40	6.96	2.32	16	89.97	4.35	17.42	16	89.50	4.73	16.82
10	1000	100	93.87	73.87	21	98.88	8.25	5.34	15	116.93	4.90	24.57	16	109.91	5.55	17.09
11	500	200	45.05	406.80	24	47.65	13.43	5.76	17	54.04	7.60	19.96	19	51.75	8.53	14.87
12	600	200	52.38	257.96	25	55.12	15.53	5.22	15	66.39	7.75	26.74	19	64.30	8.96	22.75
13	700	200	62.07	640.04	23	66.47	19.94	7.09	16	76.39	10.16	23.06	18	72.16	12.13	16.25
14	800	200	70.70	986.03	24	74.15	21.42	4.88	17	84.40	12.29	19.38	18	83.23	13.90	17.73
15	1000	200	87.50	1324.35	24	91.48	23.30	4.55	16	106.65	13.68	21.87	18	102.06	15.89	16.63
16	1000	300	84.99	7200	25	89.15	42.05	4.90	18	101.61	25.03	19.55	21	96.68	28.72	13.75
17	1200	300	102.25	7200	25	105.67	48.78	3.35	19	122.93	29.05	20.23	20	120.33	32.44	17.69
18	1300	300	119.96	7200	24	114.47	52.28	—	19	130.38	32.54	8.69	20	126.33	36.63	5.32
19	1400	300	117.08	7200	25	120.94	61.71	3.29	18	142.07	29.92	21.34	19	138.97	35.71	18.70
20	1500	300	765.22	7200	26	130.42	57.17	—	18	155.11	33.59	–	20	148.99	39.44	—
Instances using radial coverage of 0.5 km																
1	500	50	99.83	13.12	5	102.04	0.43	2.22	4	102.05	0.34	2.23	4	102.05	0.34	2.23
2	600	50	122.36	11.43	5	126.51	0.47	3.39	4	132.93	0.35	8.64	4	132.93	0.37	8.64
3	700	50	138.68	15.35	5	144.73	0.55	4.36	4	151.77	0.41	9.44	4	156.96	0.41	13.18
4	800	50	163.48	24.80	5	163.75	0.70	0.16	4	169.21	0.50	3.50	4	169.92	0.52	3.94
5	1000	50	201.94	26.66	5	205.42	0.77	1.72	4	214.12	0.60	6.03	4	214.12	0.59	6.03
6	500	100	97.20	60.15	5	102.71	0.96	5.67	4	105.66	0.79	8.71	4	105.73	0.76	8.78
7	600	100	116.65	98.19	5	117.44	1.11	0.68	6	119.35	0.86	2.31	6	119.35	0.93	2.31
8	700	100	131.50	102.31	6	134.68	1.41	2.42	6	137.15	1.11	4.29	6	137.15	1.18	4.29
9	800	100	157.34	148.20	5	159.02	1.39	1.07	5	162.97	1.11	3.58	5	162.97	1.14	3.58
10	1000	100	192.27	222.71	6	195.68	1.76	1.78	5	208.04	1.40	8.20	5	211.09	1.34	9.79
11	500	200	93.32	419.36	6	98.62	2.23	5.68	6	101.43	1.82	8.69	6	100.97	1.90	8.20
12	600	200	110.63	740.69	6	114.30	2.72	3.31	6	123.37	2.11	11.51	6	117.45	2.16	6.16
13	700	200	133.28	1061.54	6	135.63	3.51	1.77	6	137.15	2.65	2.90	6	135.39	2.85	1.58
14	800	200	152.25	1174.40	6	155.98	3.89	2.45	5	161.24	3.05	5.90	6	154.78	3.25	1.67
15	1000	200	188.15	1664.89	6	193.28	4.66	2.73	6	198.10	3.62	5.29	6	195.45	3.76	3.88
16	1000	300	184.02	5364.37	6	190.37	8.55	3.45	6	197.37	7.10	7.25	6	191.07	6.72	3.83
17	1200	300	219.90	6747.17	6	228.88	8.89	4.08	6	239.72	7.13	9.01	6	233.02	7.65	5.97
18	1300	300	237.04	6044.67	6	251.54	9.52	6.11	6	257.77	7.95	8.75	6	250.70	8.19	5.76
19	1400	300	701.78	7200	6	264.27	10.44	—	6	281.32	8.35	—	6	277.52	8.60	—
20	1500	300	728.30	7200	7	280.24	11.25	—	5	297.51	8.79	—	6	290.08	9.39	—

Table 8: Numerical results for GL and SA for the instances in Table 1.

#	$\|K\|$	$\|V\|$	Opt	CPU (s)	GL				SA			
					$\|S^*\|$	Ub	CPU (s)	Gap (%)	$\|S^*\|$	Ub	CPU (s)	Gap (%)
Instances using radial coverage of 0.2 km												
1	500	50	56.20	6.60	14	56.20	20.47	0	14	56.20	1.38	0
2	600	50	63.83	5.59	17	63.83	96.66	0	17	63.83	0.98	0
3	700	50	84.28	8.44	15	84.45	106.28	0.20	14	84.28	7.29	0
4	800	50	85.13	8.99	16	86.67	21.14	1.81	16	85.13	0.81	0
5	1000	50	105.24	16.47	16	105.89	82.76	0.62	16	105.24	0.77	0
6	500	100	49.81	22.79	19	50.71	7.04	1.81	20	49.81	8.39	0
7	600	100	56.95	27.83	19	58.78	20.20	3.21	21	56.95	5.37	0
8	700	100	67.01	236.14	19	68.54	130.87	2.29	20	67.01	13.99	0
9	800	100	76.62	54.16	19	78.71	3.69	2.72	20	76.62	11.19	0
10	1000	100	93.87	73.87	20	94.38	62.83	0.55	20	93.87	50.80	0
11	500	200	45.05	406.80	22	46.95	27.22	4.20	23	45.05	20.44	0
12	600	200	52.38	257.96	24	54.95	48.33	4.90	23	52.45	70.66	0.12
13	700	200	62.07	640.04	22	65.85	21.57	6.08	23	62.07	161.15	0
14	800	200	70.70	986.03	22	72.09	79.85	1.97	22	70.70	253.45	0
15	1000	200	87.50	1324.35	21	90.59	124.55	3.53	23	87.50	103.51	0
16	1000	300	84.99	7200	21	89.53	12.73	5.34	25	86.20	55.58	1.42
17	1200	300	102.25	7200	24	105.52	133.29	3.20	25	102.75	89.55	0.49
18	1300	300	119.96	7200	24	113.76	32.19	−5.16	24	112.00	43.53	−6.63
19	1400	300	117.08	7200	23	121.14	34.62	3.47	24	117.98	116.98	0.77
20	1500	300	765.22	7200	22	133.02	67.43	−82.61	25	129.33	95.53	−83.09
Instances using radial coverage of 0.5 km												
1	500	50	99.83	13.12	5	99.83	0.50	0	5	99.83	0.31	0
2	600	50	122.36	11.43	4	122.36	4.72	0	4	122.36	0.54	0
3	700	50	138.68	15.35	4	142.21	0.66	2.55	4	138.68	15.22	0
4	800	50	163.48	24.80	4	164.51	0.30	0.63	4	163.48	63.01	0
5	1000	50	201.94	26.66	5	201.94	2.31	0	5	201.94	0.71	0
6	500	100	97.20	60.15	5	97.20	3.81	0	5	97.20	6.15	0
7	600	100	116.65	98.19	5	116.69	3.25	0.04	5	116.65	11.89	0
8	700	100	131.50	102.31	6	131.50	8.96	0	6	131.50	0.67	0
9	800	100	157.34	148.20	5	157.49	5.12	0.10	5	157.34	0.59	0
10	1000	100	192.27	222.71	5	192.27	1.04	0	5	192.27	26.65	0
11	500	200	93.32	419.36	6	93.32	36.91	0	6	93.32	2.56	0
12	600	200	110.63	740.69	6	110.63	2.66	0	6	110.63	0.39	0
13	700	200	133.28	1061.54	6	133.28	10.03	0	6	133.28	17.64	0
14	800	200	152.25	1174.40	6	152.78	11.47	0.35	6	152.25	1.84	0
15	1000	200	188.15	1664.89	6	188.15	28.00	0	6	188.15	2.26	0
16	1000	300	184.02	5364.37	6	184.25	17.19	0.13	6	184.72	19.32	0.38
17	1200	300	219.90	6747.17	6	220.81	234.57	0.41	6	219.90	1.04	0
18	1300	300	237.04	6044.67	6	237.78	60.90	0.31	6	237.04	44.37	0
19	1400	300	701.78	7200	6	256.48	74.53	−63.45	6	262.47	5.84	−62.59
20	1500	300	728.30	7200	7	278.49	21.82	−61.76	6	282.01	10.82	−61.27

Table 9: Numerical results for GMIN, GL, and SA algorithms for the large size instances.

#	$\|K\|$	$\|V\|$	Den.	GMIN			GL				SA			
				$\|S^*\|$	Ub	CPU (s)	$\|S^*\|$	Ub	CPU (s)	Gap(%)	$\|S^*\|$	Ub	CPU (s)	Gap(%)
Instances using radial coverage of 0.2 km														
1	1000	500	10.57	26	87.34	82.69	24	86.66	7.33	0.78	25	85.66	31.45	1.97
2	1200	500	10.49	27	104.08	97.78	23	101.50	29.60	2.54	25	100.70	102.46	3.35
3	1300	500	10.56	26	111.15	102.58	25	109.18	128.19	1.80	26	105.04	369.68	5.82
4	1400	500	10.61	27	119.56	112.72	25	119.23	76.84	0.27	25	117.82	201.39	1.47
5	1500	500	10.36	27	128.46	117.14	23	128.03	82.74	0.34	26	124.94	238.19	2.82
6	1000	700	10.56	28	84.74	149.35	26	85.40	226.00	−0.77	27	83.93	273.14	0.97
7	1200	700	10.56	28	104.45	176.45	24	102.27	243.70	2.12	26	101.28	598.77	3.13
8	1300	700	11.12	28	111.94	182.19	23	109.58	112.67	2.15	25	109.52	195.97	2.21
9	1400	700	10.26	29	116.27	207.47	25	117.59	81.74	−1.13	27	113.27	387.75	2.64
10	1500	700	10.45	28	126.42	204.97	26	125.85	31.88	0.45	27	126.12	179.14	0.23
11	1000	1000	10.14	28	84.97	277.61	26	83.26	117.39	2.06	26	83.49	379.99	1.77
12	1200	1000	10.61	29	100.14	331.85	25	101.26	182.85	−1.10	29	98.15	281.68	2.03
13	1300	1000	10.63	28	107.62	354.17	25	106.83	170.39	0.74	28	105.82	518.54	1.71
14	1400	1000	10.56	30	118.58	359.51	24	117.42	57.51	0.99	26	116.20	147.08	2.05
15	1500	1000	10.32	28	124.38	394.33	24	123.45	110.95	0.76	27	123.02	207.92	1.10
Instances using radial coverage of 0.5 km														
1	1000	500	49.22	6	188.96	16.44	6	185.54	29.08	1.84	6	184.39	48.22	2.48
2	1200	500	50.03	6	229.38	19.15	6	220.48	72.47	4.03	6	222.96	9.08	2.88
3	1300	500	46.71	7	247.19	21.06	6	238.58	23.96	3.61	6	242.42	14.16	1.97
4	1400	500	51.39	6	270.10	22.35	6	257.81	8.10	4.77	6	260.91	5.05	3.53
5	1500	500	48.34	6	283.52	23.90	6	273.81	113.59	3.55	6	274.10	3.08	3.44
6	1000	700	48.65	7	189.37	34.88	6	181.26	278.97	4.47	6	183.16	79.83	3.39
7	1200	700	48.04	6	227.75	35.33	6	219.55	95.28	3.74	6	220.21	79.40	3.43
8	1300	700	47.46	7	242.48	36.88	6	238.53	130.71	1.66	6	239.97	16.56	1.04
9	1400	700	46.70	7	258.40	39.01	6	252.52	112.10	2.33	6	258.84	10.01	−0.17
10	1500	700	48.50	7	281.43	40.78	6	271.28	111.47	3.74	6	275.77	38.58	2.05
11	1000	1000	48.96	7	184.28	69.86	6	182.69	282.79	0.87	6	185.89	7.31	−0.87
12	1200	1000	48.46	7	223.99	80.81	6	215.64	125.56	3.87	6	223.71	7.83	0.13
13	1300	1000	47.63	6	245.73	77.27	6	237.25	70.79	3.58	6	240.35	19.43	2.24
14	1400	1000	47.81	7	263.12	81.55	6	254.52	75.90	3.38	7	258.23	5.52	1.89
15	1500	1000	47.01	6	282.72	84.44	6	271.97	139.55	3.95	6	277.50	50.35	1.88

solution found with P_1 or P_1^s together with its minimum CPU time in seconds, respectively. Then, in columns 6–9, 10–13 and 14–17, we report number of facilities, upper bounds, CPU times in seconds, and gaps for each heuristic GMIN, GMAX, and IC, respectively. The gaps are computed by $((\text{UB} - \text{Opt})/\text{Opt}) * 100$.

From Table 7, first we observe that the number of facilities is nearly the same as the optimal one for GMIN heuristics. In general, GMIN allows us to obtain a higher number of facilities than both IC and GMAX, whereas IC obtains more facilities than GMAX. We note that this can be explained as a consequence of the greedy decision that each heuristic performs at each iteration. Notice that GMIN removes less nodes from the graph than IC, which in turn removes less nodes than GMAX. Next, we further observe that GMIN obtains tighter bounds than the other heuristics although at a higher CPU time effort. This is also consequence of removing less nodes at each iteration since more nodes remain in subsequent iterations. Subsequently, by looking at the gap columns, we see that GMIN dominates the other ones with near-optimal gaps. In particular, the gaps are tighter for the instances that use a higher radial coverage value. Finally, we observe that the solutions obtained by GMIN require significantly less CPU time than those obtained with P_1 or P_1^s. Notice the fact that the first term in the objective function of P_1 dominating the second one implies that more facilities should be open, which should not always be the case if the cost to connect facilities under the spanning tree requirement is higher than the cost of assigning users to facilities. In order to deal with other situations where the cost of each term in the objective function of P_1 is different, later we further present numerical while varying the parameter $\alpha \in [0, 2]$ using the weighted objective function presented in Theorem 1.

In Table 8, for the sake of clarity, we repeat columns 1–5 from Table 7, whereas in columns 6–9 and 10–13, we present numerical results for GL and SA, respectively. More precisely, in these columns, we report number of facilities, upper bounds, and CPU times in seconds and gaps that we compute by $((\text{UB} - \text{Opt})/(\text{Opt})) * 100$.

From Table 8, we observe that the number of facilities is slightly larger for SA than for GL for most of the instances while using a radial coverage value of 0.2 km. However, for the coverage value of 0.5 km, most of these numbers remain nearly the same. Next, we see that the upper bounds obtained with SA are tighter than those obtained with GL when compared to the optimal solutions using both radial coverage values. In particular, this improvement is higher for 0.2 km which corresponds to the subset composed of the harder instances. The gap columns confirm these improvements. Finally, we observe that the CPU times obtained with GL and SA are significantly smaller than those required by CPLEX to solve the MILP models. In particular, we obtain negative gaps for the instances #18 and #20 using 0.2 km and for the instances #19 and #20 using 0.5 km which evidences the difficulty of finding optimal solutions for instances with higher dimensions.

In Table 9, we report similar results as those presented in Table 8, but for the larger set of instances. In columns 1–4, we report the instance number, number of users, number of facility nodes, and density of each input graph, respectively. Then, in columns 5–7, we report number of facility nodes, upper bounds, and CPU times in seconds for GMIN heuristics. Similarly, in columns 8–11 and in columns 12–15, we report number of facilities, upper bounds, CPU times in seconds, and gaps for GL and SA, respectively. Notice that, in this case, we cannot solve the MILP models with CPLEX, and then we compute the gaps using as a reference the upper bounds obtained with GMIN. More precisely, we compute the gaps as $((\text{Ub}(\text{GMIN}) - \text{Ub}(\text{GL}))/\text{Ub}(\text{GL})) * 100$ and $((\text{Ub}(\text{GMIN}) - \text{Ub}(\text{SA}))/\text{Ub}(\text{SA})) * 100$ in columns 11 and 15, respectively.

From Table 9, we mainly observe that the number of facilities obtained with the three algorithms remain nearly the same. Next, we observe that the upper bounds obtained with SA are tighter than those obtained with GL, which in turn are tighter than those obtained with GMIN for most of the instances when using a radial coverage of 0.2 km. On the opposite, GL obtains tighter gaps than SA and GMIN when using a radial coverage of 0.5 km. In particular, a few negative gaps are reported when GMIN allows us to obtain better solutions compared to a metaheuristic. This is the case for the instances #6, #9, and #12, and for the instances #9 and #11 using 0.2 and 0.5 km, respectively. Finally, we observe that higher CPU times are required for GL compared to SA for dense instances. However, the opposite occurs for the sparse ones. Notice that the CPU times obtained with GMIN are deterministic which is not the case for the metaheuristics. In order to give more insights with respect to the upper bounds and CPU times obtained, later, we report some average results as well.

In Tables 10 and 11, we further report total number of maximal cliques, maximum clique cardinalities, number of maximal independent sets, and maximum independent set numbers for the large instances presented in Table 9 for the radial coverage values of 0.2 and 0.5 km, respectively.

Similarly as for Tables 5 and 6, the total number of maximal cliques and maximal independent sets are obtained with the Bron–Kerbosch algorithm when it is possible. Otherwise, we omit presenting this information. In particular, when it is not possible to obtain all maximal cliques of G_d, its maximum clique number is obtained by solving the optimization problem (MIS) presented above using the set $\overline{E}_d$ which corresponds to the set of edges of the complement graph of G_d.

From Tables 10 and 11, we observe similar trends as in Tables 5 and 6, respectively. We observe that it is not possible to list all maximal cliques with the Bron–Kerbosch algorithm in two hours for some of the instances although it is known they can be obtained in polynomial time [14, 15]. Finally, we see that the maximum independent sets are slightly larger than the number of facilities found with the proposed algorithms.

In order to give more insights with respect to the behaviours of P_1^s, LP_1^s, GMIN, GMAX, IC, GL, and SA algorithms while varying α using the objective function of Theorem 1, in Figures 2–7, we plot upper bounds, optimal solutions, and number of facilities and CPU times in seconds

Table 10: Maximal cliques and maximal independent sets obtained with the Bron–Kerbosch algorithm for the instances #1–#15 in Table 9 using a radial coverage value of 0.2 km.

Instance number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
# Maximal C.	2184	2295	2130	2273	2073	4966	4798	4923	4743	4863	10974	12250	12843	13131	12086
Maximum C.	30	31	30	32	29	42	37	46	41	39	52	50	56	54	48
# Maximal I.S.	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Maximum I.S.	28	28	28	28	27	29	28	28	29	29	30	30	29	30	30

-, not solved in 2 hours.

Table 11: Maximal cliques and maximal independent sets obtained with the Bron–Kerbosch algorithm for the instances #1–#15 in Table 9 using a radial coverage value of 0.5 km.

Instance number	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
# Maximal C.	35161	34559	32729	45659	33872	-	-	-	-	-	-	-	-	-	-
Maximum C.	128	127	116	132	120	174	165	161	160	165	235	223	225	228	228
# Maximal I.S.	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
Maximum I.S.	6	6	7	7	6	7	7	7	7	7	7	7	7	7	7

-, not solved in 2 hours.

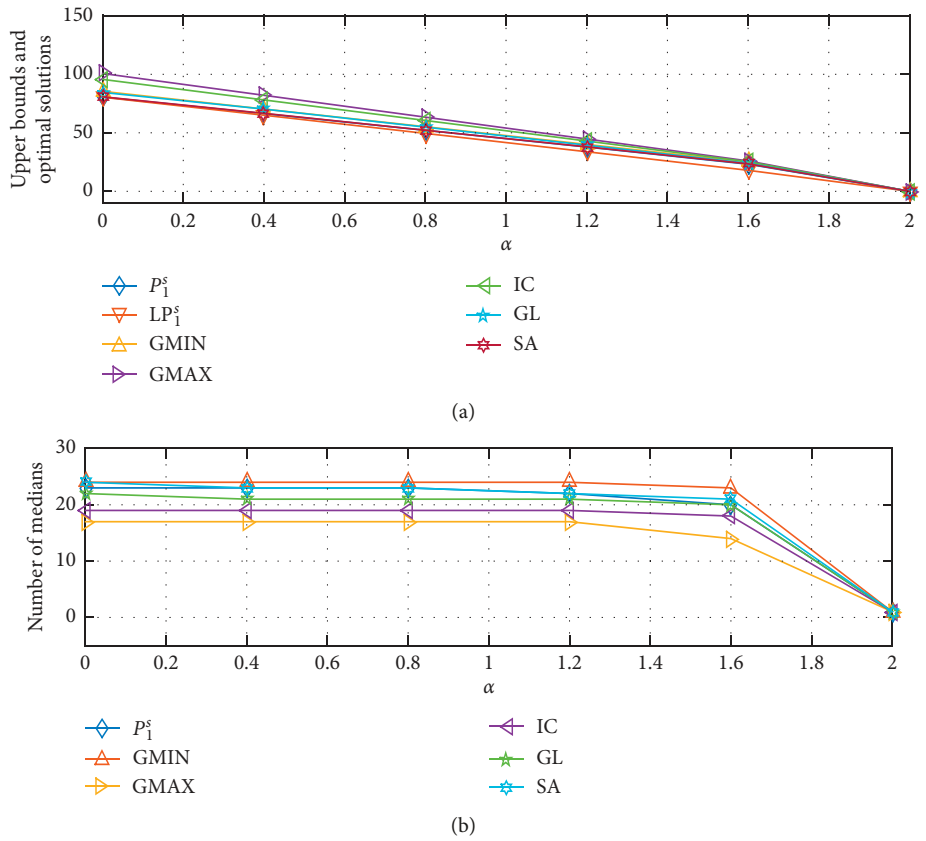
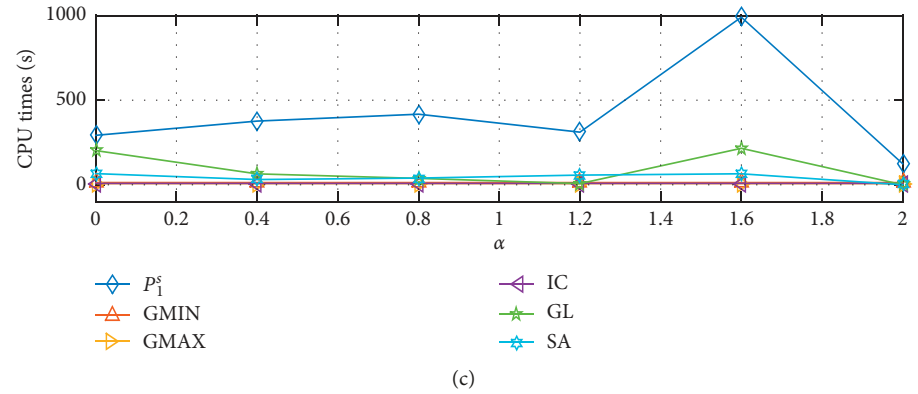
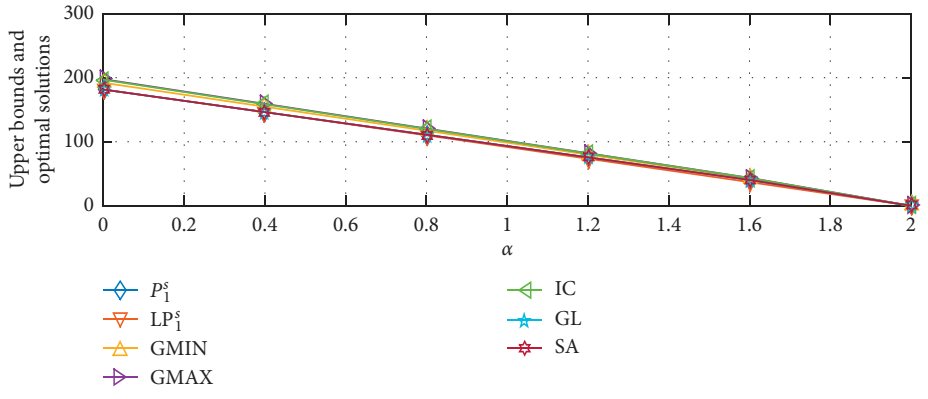


Figure 2: Continued.

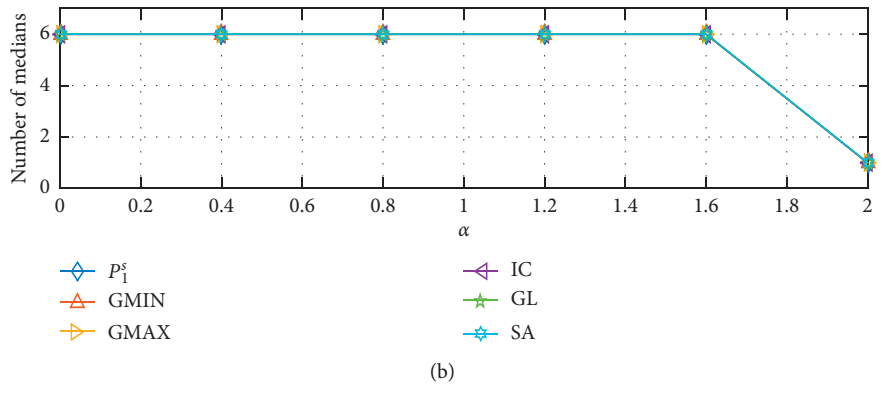


(c)

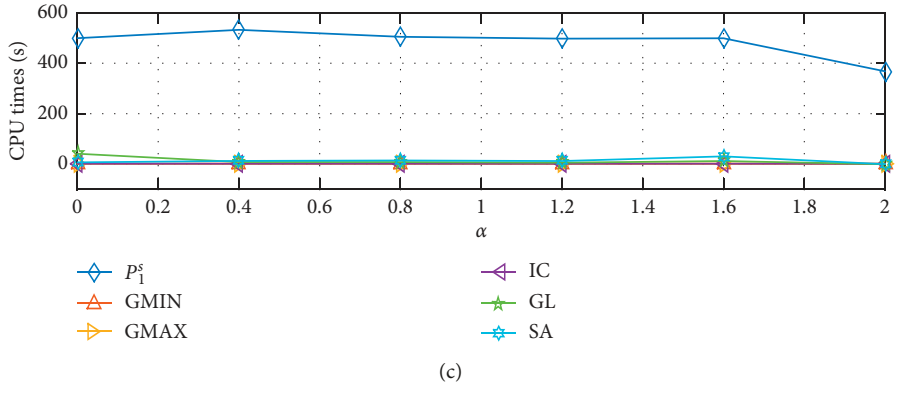
Figure 2: (a) Upper bounds and optimal solutions, (b) number of facilities, and (c) CPU times in seconds for the instance #11 in Table 1 while varying $\alpha \in [0; 2]$ and using a radial coverage value of 0.2 km.



(a)



(b)



(c)

Figure 3: (a) Upper bounds and optimal solutions, (b) number of facilities, and (c) CPU times in seconds for the instance #11 in Table 1 while varying $\alpha \in [0; 2]$ and using a radial coverage value of 0.5 km.

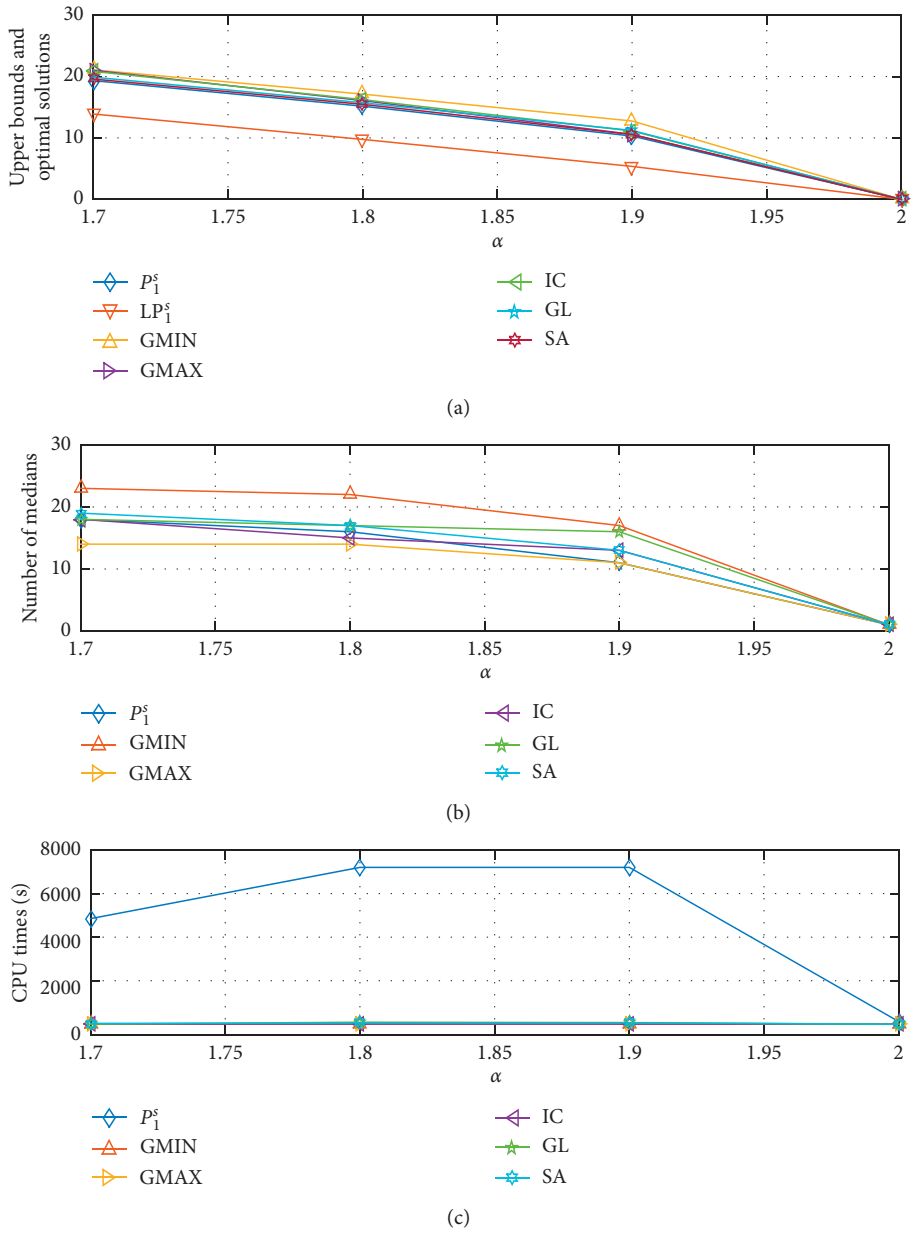


Figure 4: (a) Upper bounds and optimal solutions, (b) number of facilities, and (c) CPU times in seconds for the instance #11 in Table 1 while varying $\alpha \in [1.7; 2]$ and using a radial coverage value of 0.2 km.

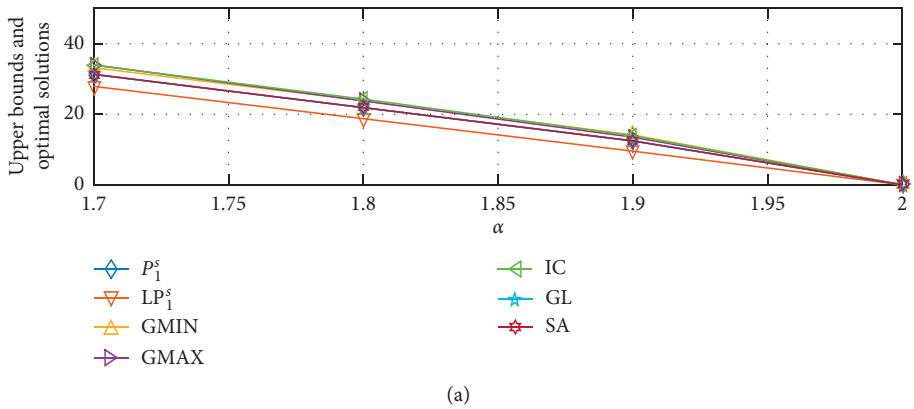


Figure 5: Continued.

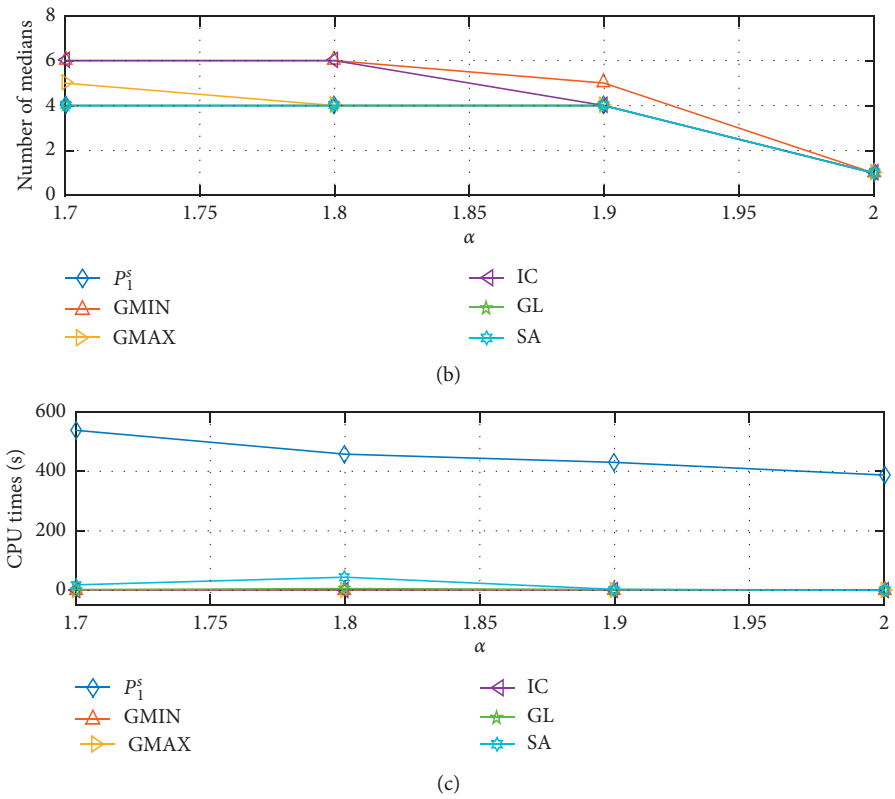


Figure 5: (a) Upper bounds and optimal solutions, (b) number of facilities, and (c) CPU times in seconds for the instance #11 in Table 1 while varying $\alpha \in [1.7; 2]$ and using a radial coverage value of 0.5 km.

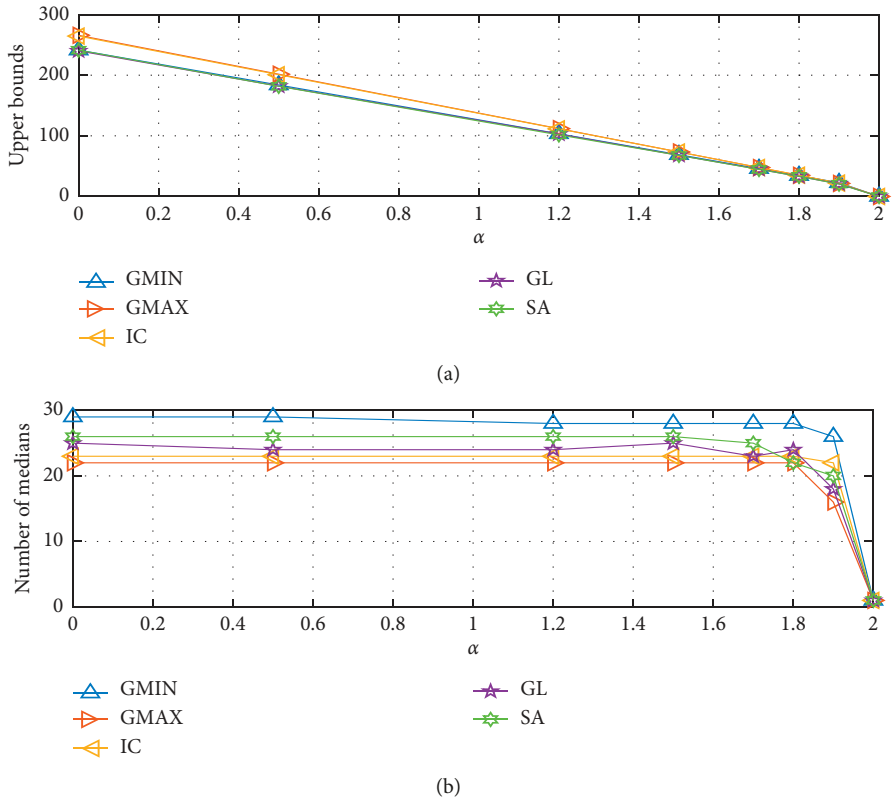
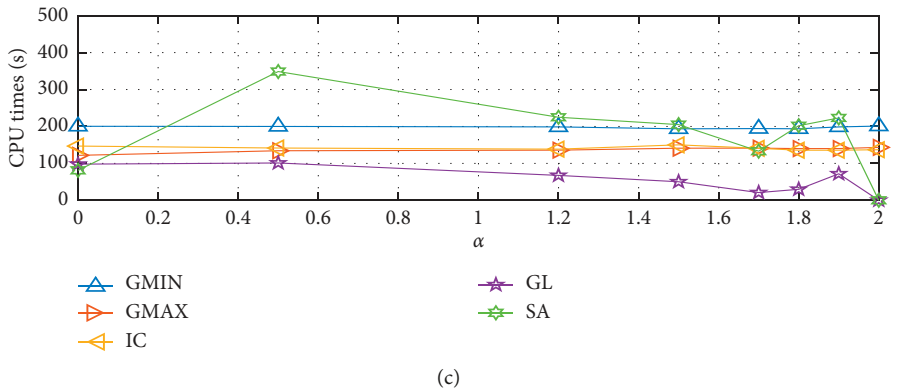
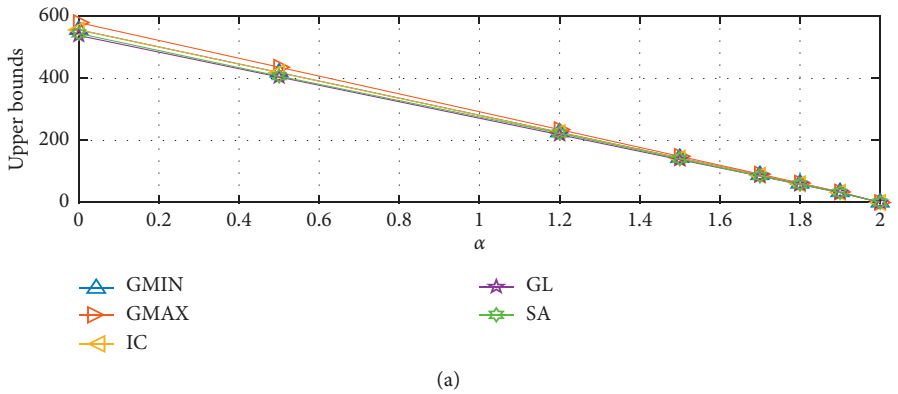


Figure 6: Continued.

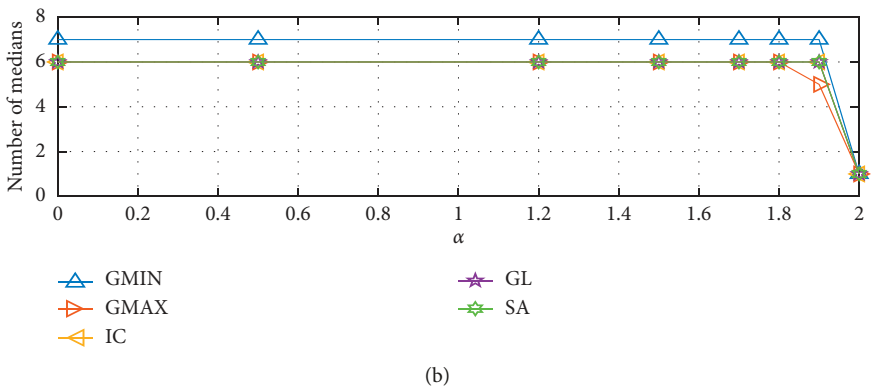


(c)

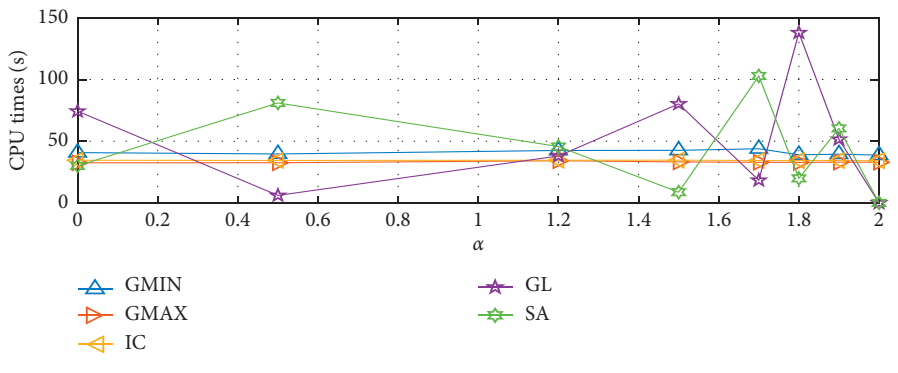
Figure 6: (a) Upper bounds, (b) number of facilities, and (c) CPU times in seconds for the instance #10 in Table 9 while varying $\alpha \in [0; 2]$ and using a radial coverage value of 0.2 km.



(a)



(b)



(c)

Figure 7: (a) Upper bounds, (b) number of facilities, and (c) CPU times in seconds for the instance #10 in Table 9 while varying $\alpha \in [0; 2]$ and using a radial coverage value of 0.5 km.

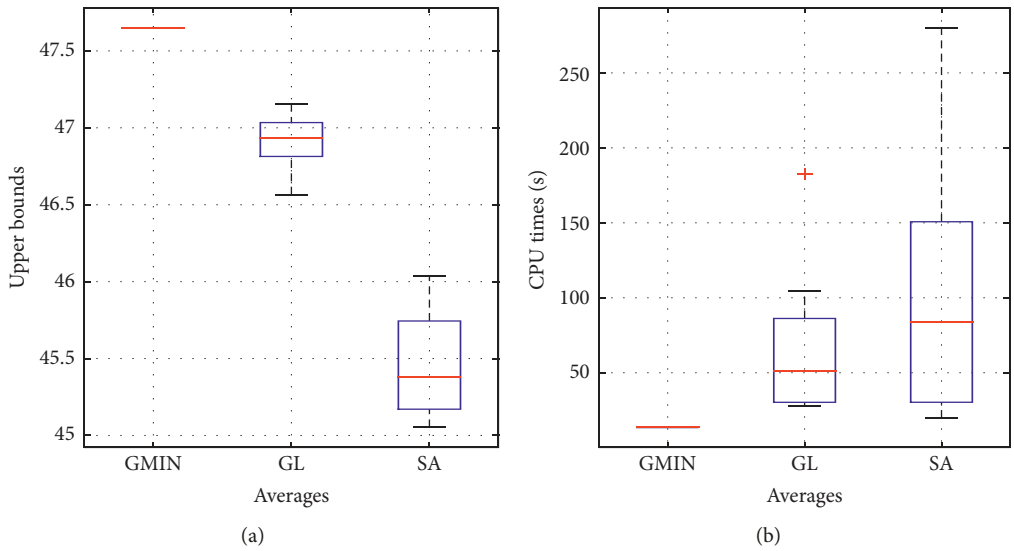


Figure 8: (a) Average upper bounds and (b) CPU times in seconds for the instance #11 in Table 1 while using a radial coverage value of 0.2 km.

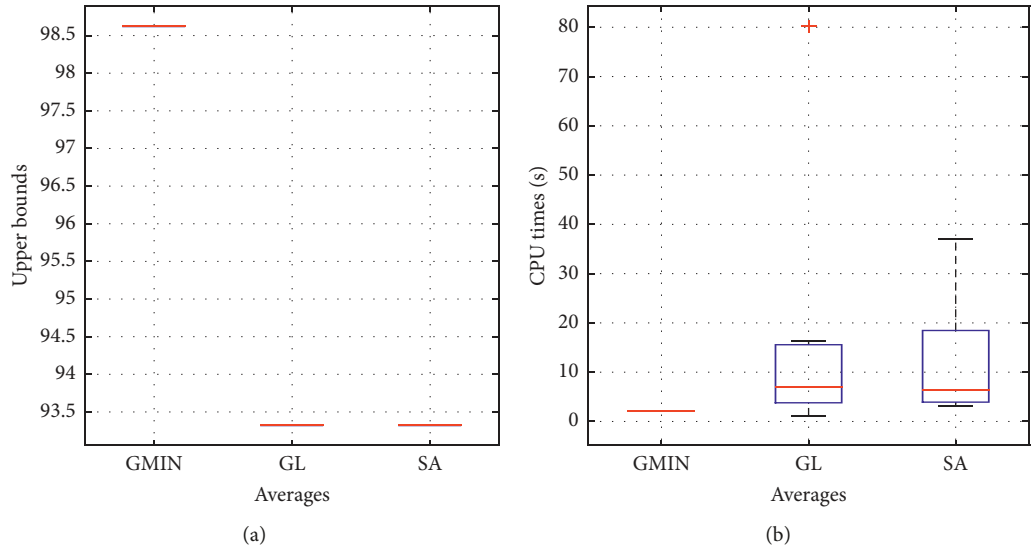


Figure 9: (a) Average upper bounds and (b) CPU times in seconds for the instance #11 in Table 1 while using a radial coverage value of 0.5 km.

for the instances #11 and #10 in Tables 1 and 9, respectively, while varying $\alpha \in [0; 2]$ and using radial coverage values of 0.2 and 0.5 km. In particular, in Figures 4 and 5, we restrict the values of α in the interval $[1.7; 2]$ in order to decrease the number of optimal facilities while searching for a tradeoff between the two terms in the objective function.

From Figures 2 and 3, we observe that the upper bounds and optimal solutions decrease with α. In particular, in Figure 2, the LP bounds remain near optimal for all values of α. Furthermore, we see that the upper bounds decrease in the following order: Ub (GMAX) > Ub (IC) > Ub (GMIN) > Ub (GL) > Ub (SA). Regarding the number of facilities (or medians), we observe that the order of the methods is as follows: Med (GMIN) > Med (SA) > Med (GL) > Med (IC) > Med (GMAX), where the optimal number of facilities is closer to Med (SA). However, the order of the methods with respect to the CPU times is as follows: CPU (P_1^s) > CPU (GL) > CPU (SA). For the heuristics, the CPU times remain nearly the same.

From Figure 3, we observe similar trends although, in this case, the curves are closer to each other with the exception in the CPU times which are significantly higher for

P_1^s. In particular, the number of facilities is exactly the same for all the methods. Finally, from Figures 2 and 3, we observe that the number of facilities decreases for values of $\alpha \in [1.7; 2]$. Consequently, in Figures 4 and 5, we plot for the same instances numerical results as in Figures 2 and 3, respectively, but for values of $\alpha \in [1.7; 2]$.

From Figure 4, we observe the following ordering for the upper bounds: Ub (GMIN) > Ub (IC) > Ub (GMAX) > Ub (GL) > Ub (SA). As it can be observed, in this case, GMAX allows us to obtain better solutions than GMIN. This can be explained by the fact that less number of facilities are required in the optimal solution of the problem. Regarding the number of medians, the ordering is Med (GMIN) > Med (SA), Med (GL) > Med (IC) > Med (GMAX), where the optimal number of facilities is closer to Med (SA) or Med (GMAX). Then, we observe that the CPU times required to solve P_1^s increase significantly for values of $\alpha \in [1.7; 2]$. Notice that, in particular, we cannot solve P_1^s to optimality in two hours for $\alpha = \{1.8, 1.9\}$, which is not the case for the instance #11 in Table 1. In general, the CPU times required by SA are higher than those required by GL. Finally, we observe that the lower bounds obtained with LP_1^s are not tight when compared to the optimal solution of the problem which explains somehow the difficulty in solving P_1^s. Similarly, from Figure 5, we see that GMAX outperforms GMIN and IC in terms of upper bounds. However, SA and GL outperform the proposed heuristics. Regarding the number of medians obtained, we observe that SA and GL obtain less number of facilities than the heuristics. Finally, the CPU times obtained show similar orders of magnitude for all the algorithms.

In Figures 6 and 7, we report analog numerical results as in Figures 2 and 3 for the instance #10 presented in Table 9 using radial coverage values of 0.2 and 0.5 km, respectively. In these figures, we do not report optimal solutions for P_1^s as it is not possible to obtain these results for the instances with higher dimensions due to CPLEX shortage of memory.

From Figure 6, we observe that the upper bounds obtained with GMIN, GL, and SA are nearly the same and better than the other ones. Regarding the number of facilities, we observe that GMIN obtains the largest values, whilst GMAX obtains the lowest ones. The number of facilities of each remaining algorithm is between these values. Finally, we observe that the CPU times obtained with SA and GL algorithms are the largest and smallest ones, respectively. Consequently, the CPU times for the remaining algorithms are between these values as well. From Figure 7, we observe similar trends for the upper bounds and nearly constant values for the number of facilities obtained with each algorithm. In particular, GMIN obtains the largest values. Subsequently, we observe that the CPU times are different for most of the algorithms and in particular higher for the metaheuristics in some cases.

In order to give more insights with respect to the behaviours of GMIN, GL, and SA algorithms, we further present average upper bounds and CPU times in seconds for the instance #11 in Table 1 and for the instance #10 in Table 9 while using radial coverage values of 0.2 and 0.5 km. These numerical results are presented in Figures 8–11 for each of the instances, respectively. In particular, the averages are obtained for 50 runs using each algorithm.

From Figures 8 and 9, we observe that the upper bounds obtained with SA are lower than GMIN and GL. However, the upper bounds obtained with GL are lower than those obtained with GMIN. Regarding the CPU times, the average values can be ordered in the opposite direction for the three algorithms being larger for SA and lower for GMIN. In particular, in Figure 9, the averages for SA and GL are closer to each other.

Next, in Figure 10, we observe that the upper bounds obtained with SA are the lowest ones. However, GMIN obtains the worst solutions. Regarding the CPU times obtained, we see that, in average, GMIN and SA require similar values. However,GL obtains solutions faster than the remaining algorithms. Subsequently, in Figure 11, we observe that GL obtains better solutions compared to GMIN and SA although at a higher CPU time. Finally, we observe that GL allows us to obtain better solutions than SA for higher radial coverage values, i.e., when dealing with dense input disk graphs. However, on the opposite, SA allows us to obtain better solutions for sparse graphs which resulted in more difficult instances to be solved by the proposed MILP models.

6. Conclusions

In this paper, we consider the problem of assigning a set K of users to a subset S of facilities chosen from a larger set V while simultaneously forming a tree backbone with S and with the additional condition that no two adjacent nodes in V of an input disk graph G_d can belong to S. The latter condition is handled by means of independent set constraints. We model this problem on unit disk graphs and propose mixed integer linear programming models in order to minimize the total connection cost distance between facilities and between customers and facilities simultaneously. Four compact polynomial formulations are proposed based on classical and set covering p-Median formulations, whilst the tree backbone formed with S is modelled with Miller–Tucker–Zemlin and path orienteering constraints, respectively. The MILP models are further strengthened with clique valid inequalities which can be obtained in polynomial time for unit disk graphs. Finally, we propose Kruskal-based heuristics and metaheuristics which are adapted from a greedy approach initially proposed for the maximum independent set problem. In particular, the metaheuristics are constructed based on guided local search and simulated annealing strategies. Our main observations and conclusions on this paper can be outlined as follows:

(1) Our numerical results indicated that only the Miller–Tucker–Zemlin constrained models allow us to obtain optimal solutions for instances with up to

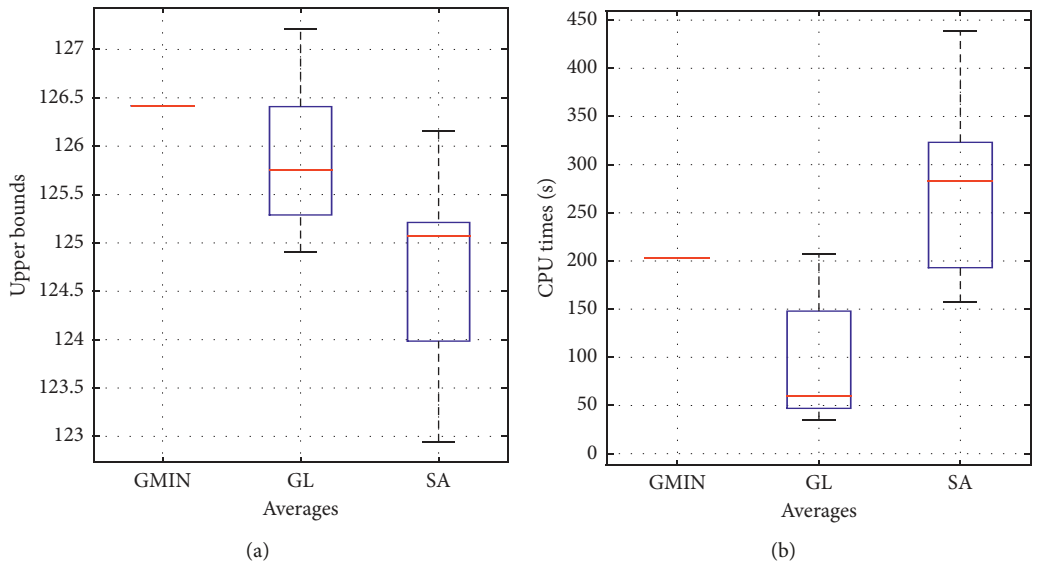


Figure 10: (a) Average upper bounds and (b) CPU times in seconds for the instance #10 in Table 9 while using a radial coverage value of 0.2 km.

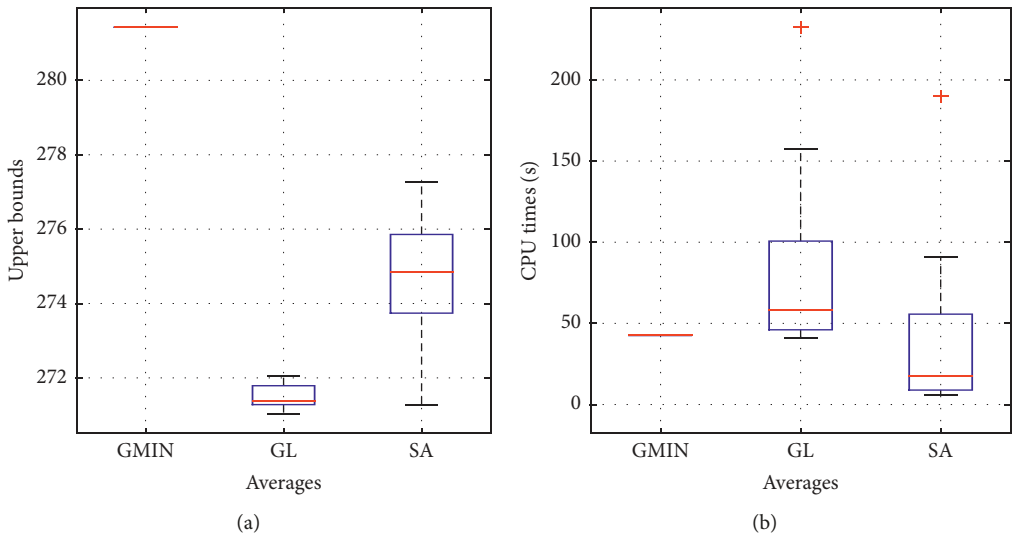


Figure 11: (a) Average upper bounds and (b) CPU times in seconds for the instance #10 in Table 9 while using a radial coverage value of 0.5 km.

200 nodes and 1000 users. Notice that finding feasible solutions with certificate of optimality is an important achievement. Indeed, it allows us to compare the solutions obtained with the proposed algorithms against optimal solutions and also to compare possibly new algorithmic approaches as part of future research.

(2) We obtained the same near-optimal solutions with all linear relaxations and gaps lower than 6% with the strengthened models for most of the instances compared to the optimal solutions.

(3) We compared Miller–Tucker–Zemlin and path orienteering constrained models using novel formulations which are constructed based on classical and set covering p-Median models. We observed, from our numerical results, that Miller–Tucker–Zemlin constrained models outperform path orienteering ones.

(4) We also proposed a heuristic framework with three variants referred to as GMIN, IC, and GMAX and noticed that GMIN outperforms the other ones for most tested instances. Furthermore, we noticed that, in some cases, when varying the weights of the objective function of the proposed MILP models, GMAX outperforms the other ones. In particular, this occurs when the second term of the objective function of P_1^s is greater than the first one, i.e., when the cost incurred to construct the backbone tree is greater than the cost of assigning users to facilities.

Finally, we verified that the heuristic approach allows one to obtain near-optimal solutions in short CPU time.

(5) Finally, we proposed two metaheuristics based on guided local search and simulated annealing greedy strategies (GL and SA, respectively) that outperformed the proposed heuristics for most of the instances. In particular, we noticed that SA obtains better results than GL on sparse disk graphs. However, the opposite occurs for dense graphs. Finally, both GL and SA allowed to obtain near-optimal solutions in significantly short CPU time and tight feasible solutions for large instances of the problem.

As future research, we plan to propose new modelling and algorithmic approaches while taking into account other network structures such as dominating and maximum leaf spanning trees. Ultimately, new modelling approaches should also be considered while including mathematical majorization concepts related with distances in trees and location theory [45], in order to compare with the models proposed in this paper.

Data Availability

All data were generated randomly as it is clearly explained in the manuscript. Consequently, no particular data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

The authors acknowledge financial support from projects FONDECYT (nos. 11180107 and 3190147).

References

[1] Z. Drezner and H. W. Hamacher, *Facility Location: Applications and Theory*, Springer, Berlin, Germany, 2002, https://www.springer.com/gp/book/9783540421726.

[2] D. Zvi, K. Kathrin, S. Anita, and O. Wesolowsky George, *The Weber Problem. Facility Location: Applications and Theory*, Springer, Berlin, Germany, 2002.

[3] R. Z. Farahani, N. Asgari, N. Heidari, M. Hosseininia, and M. Goh, "Covering problems in facility location: a review," *Computers & Industrial Engineering*, vol. 62, no. 1, pp. 368–407, 2012.

[4] J. Krarup and S. Vajda, "On Torricelli's geometrical solution to a problem of Fermat," *IMA Journal of Management Mathematics*, vol. 8, no. 3, pp. 215–224, 1997.

[5] I. D. Moon and S. S. Chaudhry, "An analisys of network location problems with distance constraints," *Management Science*, vol. 30, no. 3, pp. 290–307, 1984.

[6] P. G. Spain, "The Fermat point of a triangle," *Mathematics Magazine*, vol. 69, no. 2, pp. 131–133, 1996.

[7] A. D. Korshunov, "Coefficient of internal stability," *Kibernetika*, vol. 10, no. 1, pp. 17–28, 1974.

[8] P. Adasme, "Optimal sub-tree scheduling for wireless sensor networks with partial coverage," *Computer Standards & Interfaces*, vol. 61, pp. 20–35, 2019.

[9] P. Adasme, "Visible light communication networks under ring and tree topology constraints," *Computer Standards & Interfaces*, vol. 52, pp. 10–24, 2017.

[10] P. Adasme, R. Andrade, J. Leung, and A. Lisser, "Improved solution strategies for dominating trees," *Expert Systems with Applications*, vol. 100, pp. 30–40, 2018.

[11] P. Adasme, R. Andrade, and A. Lisser, "Minimum cost dominating tree sensor networks under probabilistic constraints," *Computer Networks*, vol. 112, pp. 208–222, 2017.

[12] J. G. Carlsson and F. Jia, "Continuous facility location with backbone network costs," *Transportation Science*, vol. 49, no. 3, pp. 433–451, 2015.

[13] D. Eppstein, "Small maximal independent sets and faster exact graph coloring," *Journal of Graph Algorithms and Applications*, vol. 7, no. 2, pp. 131–140, 2003.

[14] F. Cazals and C. Karande, "A note on the problem of reporting maximal cliques," *Theoretical Computer Science*, vol. 407, no. 1–3, pp. 564–568, 2008.

[15] T. Izumi and D. Suzuki, "Faster enumeration of all maximal cliques in unit disk graphs using geometric structure," *IEICE Transactions on Information and Systems*, vol. E98.D, no. 3, pp. 490–496, 2015.

[16] B. N. Clark, C. J. Colbourn, and D. S. Johnson, "Unit disk graphs," *Discrete Mathematics*, vol. 86, no. 1–3, pp. 165–177, 1990.

[17] T. Erlebach and J. Fiala, "Independence and coloring problems on intersection graphs of disks," in *Efficient Approximation and Online Algorithms*, pp. 135–155, Springer, Berlin, Germany, 2006.

[18] D. Wang and Y. S. Kuo, "A study on two geometric location problems," *Information Processing Letters*, vol. 28, no. 6, pp. 281–286, 1988.

[19] D. Eppstein, M. Loffler, and D. Strash, "Listing all maximal cliques in sparse graphs in near-optimal time," in *Proceedings of the International Symposium on Algorithms and Computation ISAAC 2010*, pp. 403–414, Jeju Island, December 2010.

[20] S. García, M. Labbé, and A. Marín, "Solving large p-median problems with a radius formulation," *Informs Journal on Computing*, vol. 23, no. 4, pp. 546–556, 2011.

[21] N. Mladenović, J. Brimberg, P. Hansen, and J. A. Moreno Pérez, "The p-median problem: a survey of metaheuristic approaches," *European Journal of Operational Research*, vol. 179, no. 3, pp. 927–939, 2007.

[22] M. Desrochers and G. Laporte, "Improvements and extensions to the Miller-Tucker-Zemlin subtour elimination constraints," *Operations Research Letters*, vol. 10, no. 1, pp. 27–36, 1991.

[23] C. Luna Mota, *The Optimum Communication Spanning Tree Problem: Properties, Models and Algorithms*, PhD thesis, Universitat Politécnica de Catalunya, Barcelona, Spain, 2015.

[24] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, MIT Press and McGraw-Hill, Cumberland, RI, USA, 2009.

[25] S. Sakai, M. Togasaki, and K. Yamazaki, "A note on greedy algorithms for the maximum weighted independent set problem," *Discrete Applied Mathematics*, vol. 126, no. 2-3, pp. 313–322, 2003.

[26] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science, New Series*, vol. 220, no. 4598, pp. 671–680, 1983.

[27] C. Koulamas, S. Antony, and R. Jaen, "A survey of simulated annealing applications to operations research problems," *Omega*, vol. 22, no. 1, pp. 41–56, 1994.
[28] B. Suman and P. Kumar, "A survey of simulated annealing as a tool for single and multiobjective optimization," *Journal of the Operational Research Society*, vol. 57, no. 10, pp. 1143–1160, 2006.
[29] C. Voudouris and E. Tsang, "Guided local search and its application to the traveling salesman problem," *European Journal of Operational Research*, vol. 113, no. 2, pp. 469–499, 1999.
[30] I. Rodríguez-Martín, J. J. Salazar-González, and H. Yaman, "Hierarchical survivable network design problems," *Electronic Notes in Discrete Mathematics*, vol. 52, pp. 229–236, 2016.
[31] I. Rodríguez, J. J. Salazar, and H. Yaman, "The ring/k-rings network design problem, model and branch-and-cut algorithm," *Networks*, vol. 68, no. 2, pp. 130–140, 2016.
[32] H. Yaman, "Allocation strategies in hub networks," *European Journal of Operational Research*, vol. 211, no. 3, pp. 442–451, 2011.
[33] M. Gendreau, G. Laporte, and F. Semet, "The covering tour problem," *Operations Research*, vol. 45, no. 4, pp. 568–576, 1997.
[34] B. Boffey and S. C. Narula, "Models for multi-path covering-routing problems," *Annals of Operations Research*, vol. 82, pp. 331–342, 1998.
[35] R. Church and C. ReVelle, "The maximal covering location problem," *Papers of the Regional Science Association*, vol. 32, no. 1, pp. 101–118, 1974.
[36] V. A. Hutson and C. ReVelle, "Indirect covering tree problems on spanning tree networks," *European Journal of Operational Research*, vol. 65, no. 1, pp. 20–32, 1993.
[37] M. Berg, S. Cabello, and S. Har-Peled, "Covering many or few points with unit disks," in *WAOA 2006: LNCS*, T. Erlebach and C. Kaklamanis, Eds., vol. 4368, pp. 55–68, Springer-Verlag, Berlin, Germany, 2006.
[38] S. Butenko, *Maximum Independent Set and Related Problems, with Applications*, PhD. thesis, School of the University of Florida, Gainesville, FL, USA, 2003.
[39] J. Krarup and P. M. Pruzan, "The simple plant location problem: survey and synthesis," *European Journal of Operational Research*, vol. 12, no. 1, pp. 36–81, 1983.
[40] R. Ravi, R. Sundaram, M. V. Marathe, D. J. Rosenkrantz, and S. S. Ravi, "Spanning trees-short or small," *SIAM Journal on Discrete Mathematics*, vol. 9, no. 2, pp. 178–200, 1996.
[41] G. Cornuejols, G. L. Nemhauser, and L. A. Wolsey, "A canonical representation of simple plant location problems and its applications," *SIAM Journal on Algebraic Discrete Methods*, vol. 1, no. 3, pp. 261–272, 1980.
[42] B. Hajek, "Cooling schedules for optimal annealing," *Mathematics of Operations Research*, vol. 13, no. 2, pp. 311–329, 1988.
[43] K. Du and M. N. S. Swamy, "Simulated Annealing," in *Search and Optimization by Metaheuristics*, Birkhäuser, Cham, Switzerland, 2016.
[44] IBM ILOG CPLEX Optimization Studio Information Center, http://pic.dhe.ibm.com/infocenter/cosinfoc/v12r4/index.jsp.
[45] G. Dahl, "Majorization and distances in trees," *Networks*, vol. 50, no. 4, pp. 251–257, 2007.