Les Cahiers du GERAD ISSN: 0711–2440

Median and covering location problems with interconnected facilities

M. Cherkesly, M. Landete,
G. Laporte

G–2018–19

March 2018

La publication de ces rapports de recherche est rendue possible grâce au soutien de HEC Montréal, Polytechnique Montréal, Université McGill, Université du Québec à Montréal, ainsi que du Fonds de recherche du Québec – Nature et technologies.

Dépôt légal – Bibliothèque et Archives nationales du Québec, 2018
– Bibliothèque et Archives Canada, 2018

The publication of these research reports is made possible thanks to the support of HEC Montréal, Polytechnique Montréal, McGill University, Université du Québec à Montréal, as well as the Fonds de recherche du Québec – Nature et technologies.

Legal deposit – Bibliothèque et Archives nationales du Québec, 2018
– Library and Archives Canada, 2018

GERAD HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7

Tél. : 514 340-6053
Téléc. : 514 340-5665
info@gerad.ca
www.gerad.ca

Median and covering location problems with interconnected facilities

Marilène Cherkesly [a]

Mercedes Landete [b]

Gilbert Laporte [c]

[a] *GERAD & Département de management et technologie, École des Sciences de la gestion, Université du Québec Montréal, Montréal (Québec), Canada, H3C 3P8*

[b] *Departamento de Estadstica, Matemticas e Informtica, Instituto Unversitario Centro de Investigacin Operativa, Universidad Miguel Hernndez de Elche, Avenida Universidad s/n. 03202 Elche (Alicante), Spain*

[c] *GERAD & Department of Decision Sciences, HEC Montréal, Montréal (Québec), Canada, H3T 2A7*

cherkesly.marilene@uqam.ca
landete@umh.es
gilbert.laporte@cirrelt.ca

March 2018
Les Cahiers du GERAD
G–2018–19

Abstract: This paper introduces two classes of location problems with interconnected facilities. These problems differ from classical location problems in the sense that the facilities to be located must be interconnected, i.e. located within a prescribed distance of a central office or of each other. Such problems arise, for example, in contexts where forest fire-fighters must be able to reach each other by short-range radios. Median and covering problems are modeled through several formulations. A greedy randomized adaptive search procedure (GRASP) is then developed. Extensive tests are performed on instances of the median problem in order to compare two of the models and to assess the performance of the GRASP.

Keywords: Facility location, GRASP, interconnectivity, median problem, covering, metaheuristic

Acknowledgments: This work was partially funded by the Canadian Natural Sciences and Engineering Research Council under grants 2017-06106 and 2015-06189 and by Spanish Ministerio de Economía y Competividad (MINECO/FEDER) project MTM-2015-68097(P). This support is gratefully acknowledged.

1 Introduction

The purpose of this paper is to introduce two classes of location problems with interconnected facilities. These problems differ from classical location problems in the sense that the facilities to be located must be interconnected. More precisely, two facilities are interconnected whenever the distance between them does not exceed a given limit r (see Figure 1). These problems are rooted in the work of Demaine et al. (2009) who were concerned with the design of a connected communication network linking forest fire-fighters who must extinguish fires in several forest parcels. The fire-fighters communicate between themselves and with a central office (a fixed root) by means of short-range radios (walkie-talkies). These problems arise, for example, in remote regions where mobile phone connections are not available. The range of the radio connection defines the value of r. Because the designed network is connected, messages can be relayed between its nodes.

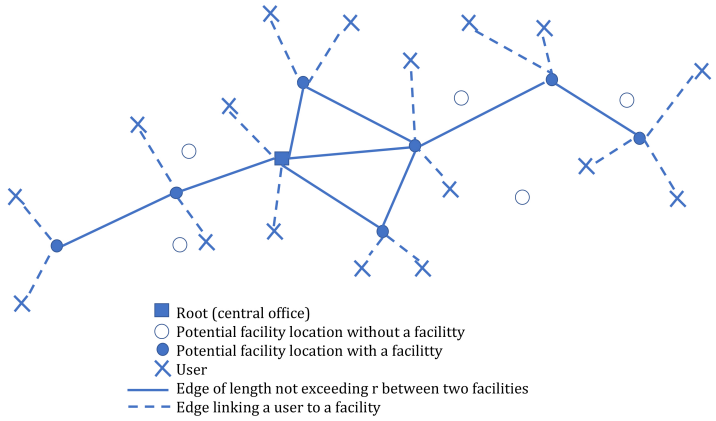

Figure 1: Network showing a set of interconnected facilities and an allocation of users to these facilities

Several variants of these problems exist depending on the number p of facilities to locate, on their cost and on the objective function. The most general case arises when p is a variable, facility fixed costs are imposed, and the objective is to minimize the total access cost of the users to the facilities. This problem is similar to the simple plant location problem (Kuehn and Hamburger (1963)). When the number of facilities is a given value p, then the problem is similar to the classical p-median problem (Hakimi (1964)). Another variant is to locate exactly p facilities in order to maximize the total demand within a distance R of a facility, as in Church and ReVelle (1974), for example, or at most p facilities if these have fixed costs (Kolen and Tamir (1990)). When $r = \infty$ the location problems with interconnected facilities reduce to their classical counterparts.

Related problems in which the facilities are interconnected are the so-called tree of hubs location problem (Contreras et al. (2010)) and some more general network design problems where the underlying connected network is not necessarily a tree (Contreras and Fernndez (2012)). In these problems the interconnected network often acts as a hub through which origin-destination commodity flows transit and the cost of the network is accounted for in the optimization model. In contrast, in this paper the cost of the edges linking the facilities is not considered; all that matters is that these facilities form a connected network. Laporte and Rodrguez Martn (2007) surveyed the related cycle location problems in which the interconnections are restricted to cycles.

We consider two basic problems arising in the context: the Median Problem with Interconnected Facilities (MPIF) and the Covering Location Problems with Interconnected Facilities (CPIF). To this end end, we will first introduce in Section 2 several mathematical models for the basic version of the MPIF and of the CPIF, and of some of their variants. Since these models can only be solved exactly for moderate sizes by commercial solvers, we will develop in Section 3 a GRASP metaheuristic to solve larger instances. Extensive computational experiments will be conducted in Section 4 on median instances. This will be followed by conclusions in Section 5 .

2 Mathematical models

The problems under study are defined in an undirected graph $G = (N, E)$, where N is the node set and E is the edge set. The node set N is the union of a set of potential facility sites I including a root node 0, and a set of users J. These two sets may be disjoint or may intersect. The set E is made up of all edges connecting two nodes of I within a distance r of each other, as well as all edges connecting nodes of I and J. Let d_k denote the demand of user $k \in J$. Locating a facility at node i incurs a fixed cost g_i. The cost of allocating users $k \in J$ to a facility located at $i \in I$ is equal to c_{ki}. The costs associated with the edges linking two nodes of J are not considered in the MPIF and in the CPIF. As is usually the case in location problems, all costs relate to the same time horizon.

2.1 Models for the MPIF

The subgraph of G induced by $V \subseteq N$ is the graph $G_V = (V, E_V)$, where E_V is the set of all the edges $[u, v] \in E$ with $u, v \in V$, $u < v$. For each $i \in I$, let y_i be a binary location variable equal to 1 if and only if a facility is located at site i. For each $i, j \in I, i \neq j$, let x_{ij} be a binary variable equal to 1 if and only if facilities are located at both sites i and j and are interconnected (within a distance r). For each $k \in J$ an $i \in I$, the binary variable w_{ki} indicates whether user k is allocated to site i or not.

In this context, let $x = (x_{ij})$ be a solution vector and let $C(x)$ be the set of solutions consisting of a connected subgraph of G_I. The problem is then

$$
\begin{aligned}
\text{(MPIF)} \qquad & \text{minimize} && \sum_{i\in I} g_i y_i + \sum_{k\in J}\sum_{i\in I} d_k c_{ki} w_{ki} && && \\
& \text{subject to} && x \in C(x) && && (1)\\
& && y_0 = 1 && && (2)\\
& && x_{ij} = 0 && i, j \in I : i \neq j, c_{ij} > r && (3)\\
& && x_{ij} \leq y_i && i, j \in I : i \neq j && (4)\\
& && x_{ij} \leq y_j && i, j \in I : i \neq j && (5)\\
& && w_{ki} \leq y_i && k \in J, i \in I && (6)\\
& && y_k + \sum_{i\in I} w_{ki} = 1 && k \in J \cap I && (7)\\
& && \sum_{i\in I} w_{ki} = 1 && k \in J \setminus (J \cap I) && (8)\\
& && x_{ij} \in \{0, 1\} && i, j \in I : i \neq j && (9)\\
& && w_{ki} \in \{0, 1\} && k \in J, i \in I && (10)\\
& && y_i \in \{0, 1\} && i \in I. && (11)
\end{aligned}
$$

Constraint (1) means that x consists of a connected subgraph of G_I. Constraint (2) ensures that the central office or fixed root is open. Constraints (3) state that both ends of any edge of the connected subgraph of G_I defined by x are interconnected within a distance r, while constraints (4) and (5) state that both ends of any edge of the same connected subgraph of G_I are open facilities. From constraints (6) users are only assigned to open facilities, and from constraints (7) users at sites where facilities are open are not allocated to other facilities. Constraints (8) guarantee that each user is served. Finally, constraints (9),(10) and (11) define the domain of the variables.

The set $C(x)$ can be restricted to trees because the cost associated with the edges of G_I do not appear in the objective function. Imposing a tree structure can be achieved by solving

$$
\begin{aligned}
\text{(MPIF-1)} \qquad & \text{minimize} && \sum_{i\in I} g_i y_i + \sum_{k\in J}\sum_{i\in I} d_k c_{ki} w_{ki} \\
& \text{subject to} && (2)\text{–}(11)
\end{aligned}
$$

$$\sum_{i,j\in I:i\neq j} x_{ij} = \sum_{i\in I} y_i - 1 \tag{12}$$

$$\sum_{i\in S, j\in \bar{S}} x_{ij} \geq \sum_{i\in I} y_i/|S| \qquad S \subseteq I \setminus \{0\}. \tag{13}$$

In this formulation, constraints (12) state that the number of edges in a tree is one less than the number of nodes. Constraints (13) impose connectivity (see, e.g., Laporte and Rodrguez Martn (2007)) and are exponential in $|I|$. However, following Landete and Marn (2014) a more compact formulation containing a quadratic number of variables and constraints can be obtained by introducing non-negative flow variables f_{ij} for all $i, j \in I$. In this formulation, two reverse arcs (i, j) and (j, i) are associated with edge $[i, j] \in E_I$:

(MPIF-2) minimize $\sum_{i\in I} g_i y_i + \sum_{k\in J}\sum_{i\in I} d_k c_{ki} w_{ki}$

subject to (2)–(11)

$$x_{i0} = 0 \qquad i \in I : i \neq 0 \tag{14}$$

$$\sum_{i\in I\setminus\{0\}} f_{0i} = \sum_{i\in I} y_i - 1 \tag{15}$$

$$\sum_{i\in I\setminus\{j\}} f_{ij} - \sum_{i\in I\setminus\{j\}} f_{ji} = y_j \qquad j \in I \tag{16}$$

$$f_{ij} \leq M x_{ij} \qquad i, j \in I : i \neq j \tag{17}$$

$$\sum_{i\in I\setminus\{j\}} x_{ij} = y_j \qquad j \in I \tag{18}$$

$$f_{ij} \geq 0 \qquad i, j \in I : i \neq j, \tag{19}$$

where M is a large number equal to $|I| - 1$. Constraints (14) ensure that no arc enters the root node. Constraint (15) states that the root node sends $\sum_{i\in I} y_i - 1$ units of flow, which corresponds to the number of nodes of the subgraph of G_I. By constraints (16) the amount of flow entering a node of the tree is one unit larger than the flow going out of the same node. From constraints (17) if an arc has a flow it belongs to the arborescence, and from (18) exactly one arc enters each node of the tree.

Note, however, that since the solution does not need to be a tree, it is possible to remove the x variables from (MPIF-2) to arrive at the simpler formulation

(MPIF-3) minimize $\sum_{i\in I} g_i y_i + \sum_{k\in J}\sum_{i\in I} d_k c_{ki} w_{ki}$

subject to (2), (6)–(8), (10), (11), (15), (16), (19)

$$f_{ij} \leq M y_i \qquad i, j \in I \tag{20}$$

$$f_{ij} \leq M y_j \qquad i, j \in I \tag{21}$$

$$f_{ij} = 0 \qquad i, j \in I : i \neq j, c_{ij} > r. \tag{22}$$

In this model, constraints (20) and (21) are similar to (4) and (5), and (22) is similar to (3). Common variants of (MPIF-1), (MPIF-2) and (MPIF-3) are obtained by restricting the number of facilities to be equal to p. In this case, the constraint

$$\sum_{i\in I} y_i = p \tag{23}$$

is imposed and $\sum_{i\in I} y_i$ is replaced with p wherever it appears. In addition, one can set $M = p - 1$.

2.2 Models for the CPIF

Let α be a user-defined non-negative parameter and let a_{ki} be a binary parameter equal to 1 if $c_{ki} \leq R$ and to 0 otherwise. Finally, let z_k be a binary variable equal to 1 if and only if the user k is covered by an open facility. Then, as in Section 2.1, the CPIF can be modelled in three different ways:

$$\text{(CPIF-1)} \quad \text{minimize} \quad \alpha \sum_{i \in I} g_i y_i + \sum_{k \in J} d_k (1 - z_k)$$

$$\text{subject to} \quad (2)\text{–}(5), (9), (11)\text{–}(13)$$

$$z_k \leq \sum_{i \in I} a_{ki} y_i \qquad k \in J \tag{24}$$

$$z_k \in \{0, 1\} \qquad k \in J, \tag{25}$$

$$\text{(CPIF-2)} \quad \text{minimize} \quad \alpha \sum_{i \in I} g_i y_i + \sum_{k \in J} d_k (1 - z_k)$$

$$\text{subject to} \quad (2)\text{–}(5), (9), (11), (15)\text{–}(19), (24), (25),$$

$$\text{(CPIF-3)} \quad \text{minimize} \quad \alpha \sum_{i \in I} g_i y_i + \sum_{k \in J} d_k (1 - z_k)$$

$$\text{subject to} \quad (2), (11), (15)\text{–}(17), (19)\text{–}(21), (24), (25).$$

Formulation (CPIF-1) has three families of variables: x-variables, y-variables and z-variables. Formulation (CPIF-2) has four families of variables: x-variables, y-variables, z-variables and f-variables. Formulation (CPIF-3) has three families of variables: y-variables, f-variables and z-variables. In these three models, the parameter α enforces the decision-maker's ponderation between demand coverage and facility cost. Alternatively, one can remove the first term of the objective function and impose a budget B on the facility cost:

$$\sum_{i \in I} g_i y_i \leq B. \tag{26}$$

In addition, as for the MPIF we can force the number of open facilities to be equal to p.

3 Metaheuristic

The proposed metaheuristic we have developed is a local search algorithm. It applies to the median and covering problems. Algorithm 1 presents its pseudo-code. An initial solution is generated by means of a greedy randomized adaptive search procedure (GRASP) (see, e.g., Feo and Resende (1995), Resende and Werneck (2004), Resende and Ribeiro (2010)). The cost evaluation for each potential facility location is based on a savings criterion. Each solution undergoes a local search phase to first minimize the total assignment cost ($\sum_{k \in J} \sum_{i \in I} d_k c_{ki} w_{ki}$) and then the total location cost ($\sum_{i \in I} g_i y_i$). When this local search phase is completed, it is desirable to perturb the solution since the algorithm often yields a local optimum. The proposed perturbation operator is a simplified version of an adaptive large neighborhood search procedure. After the solution has been perturbed, it undergoes a local search phase. The solution is improved with the perturbation operator and the local search operators until no additional improvement is possible. Here follows a detailed description of the algorithm and of the local search operators.

3.1 GRASP

Algorithm 2 details our GRASP construction algorithm. In it, the solution is initialized with the root node 0. A set of potential facilities which are within a distance r of an existing node of the solution is then created. This part ensures that each selected facility will be interconnected. The insertion cost of a potential facility i is computed as the impact of adding it to the current solution, i.e., $\Delta = c - c(i^+)$, where c is the cost of

the current solution and $c(i^+)$ is the cost of the current solution that includes the potential facility. The restricted candidate list is then made up of the λ facilities with the largest values of Δ, one of which is randomly selected and added to the solution. This process is repeated until a stopping criterion has been reached. For a fixed number p of facilities, this stopping criterion is reached when the solution contains exactly p facilities. Otherwise, the stopping criterion is reached when there are no more facilities whose inclusion would decrease the current total cost, i.e., $\Delta_i < 0, \forall i \in I$.

3.2 Local search operators

We use four local search operators in the MPIF and the CPIF: an exchange operator, a removal operator, an insertion operator, and a perturbation operator. Algorithm 3 presents a pseudo-code of the local search phase.

3.2.1 Exchange operator

The exchange operator selects a located facility $i_1 \in I$ in the current solution ($y_{i_1} = 1$) and exchanges it with another facility that is currently not in the current solution ($i_2 \in I, y_{i_2} = 0$). For each facility of the current solution ($i_1 \in I, y_{i_1} = 1$) and each facility not in the current solution ($i_2 \in I, y_{i_2} = 0$), the exchange cost is computed as the impact of adding facility i_2 to the current solution and removing facility i_1, i.e., $\Delta = c - c(i_1^-, i_2^+)$, where c is the cost of the current solution and $c(i_1^-, i_2^+)$ is the cost of the solution when adding facility i_2 and removing facility i_1. In addition, the exchange must yield a feasible solution (i.e., all facilities must be interconnected). The exchange operator then selects the facility yielding the largest cost decrease, i.e., $\max_{i_1,i_2} \Delta = c - c(i_1^-, i_2^+)$. Algorithm 4 provides the pseudo-code of the exchange operator.

3.2.2 Removal operator

The removal operator selects a located facility $i_1 \in I$ in the current solution ($y_{i_1} = 1$) and removes it. The removal cost of a facility ($i_1 \in I, y_{i_1} = 1$) is computed as the impact of removing the facility, i.e., $\Delta = c - c(i_1^-)$, where c is the cost of the current solution and $c(i_1^-)$ is the cost of the solution without facility i_1. In addition, the removal must yield a feasible solution, that is, when removing facility i_1 all other facilities must remain interconnected. The removal operator then selects the facility yielding the largest cost decrease, i.e., $\max_{i_1} \Delta = c - c(i_1^-)$. Algorithm 5 provides the pseudo-code of the removal operator.

3.2.3 Insertion operator

The insertion operator aims to increase the number of facilities by inserting a facility $i_2 \in I$ not currently located ($y_{i_2} = 0$). For each unlocated facility ($i_2 \in I, y_{i_2} = 0$), the insertion cost is computed as the impact of adding the facility to the current solution, i.e., $\Delta = c - c(i_2^+)$, where c is the cost of the current solution and $c(i_2^+)$ is the cost of the solution when inserting facility i_2. In addition, the insertion must result in a feasible solution, that is, facility i_2 must be interconnected with at least one facility already located. The insertion operator then selects the facility for which the insertion results in the highest cost decrease, i.e., $\max_{i_2} \Delta = c - c(i_2^+)$. Algorithm 6 provides the pseudo-code of the insertion operator.

Algorithm 1 Metaheuristic

```
Input G = (N, E), where N is the node set and E is the edge set
Solution ← ConstructGraspSolution( G )
do
    BestSolution ← LocalSearch( Solution )
    PerturbSolution ← Perturbation( BestSolution )
    NewBestSolution ← LocalSearch( PerturbSolution )
    if cost of NewBestSolution ≤ cost of BestSolution then
        Solution ← NewBestSolution
    end if
while cost of NewBestSolution ≤ cost of BestSolution
Return( BestSolution )
```

Algorithm 2 ConstructGraspSolution(G)

Define λ as a user-defined parameter
Solution $\leftarrow$ { 0 }
for Solution construction not completed **do**
    $PotentialFacility \leftarrow \{i \in I \backslash Solution : \exists j \in Solution, c_{ij} \leq r\}$
    for Each potentital facility $i \in PotentialFacility$ **do**
        $\Delta_i \leftarrow$ ComputeInsertionCost(i)
    end for
    RestrictedCandidateList $\leftarrow$ Keep λ potential facilities according to the largest impact Δ_i
    $s \leftarrow$ SelectElementAtRandom(RestrictedCandidateList)
    Solution $\leftarrow$ Solution $\cup \{s\}$
end for
Return(Solution)

Algorithm 3 LocalSearch(Solution)

do
    BestSolution $\leftarrow$ Exchange(Solution)
    if cost of BestSolution $\leq$ cost of Solution **then**
        Solution $\leftarrow$ BestSolution
    else
        BestSolution $\leftarrow$ Remove(Solution)
        if cost of BestSolution $\leq$ cost of Solution **then**
            Solution $\leftarrow$ BestSolution
        else
            BestSolution $\leftarrow$ Insert(Solution)
            if cost of BestSolution $\leq$ cost of Solution **then**
                Solution $\leftarrow$ BestSolution
            end if
        end if
    end if
while cost of BestSolution $\leq$ cost of Solution
Return(Solution)

Algorithm 4 Exchange(Solution)

$c \leftarrow$ Cost of Solution
$\overline{\Delta} \leftarrow 0$
bestRemoval $\leftarrow \emptyset$
bestInsertion $\leftarrow \emptyset$
for Each located facility $i_1 \in$ Solution **do**
    $\overline{\Delta_{i_1}} \leftarrow 0$
    bestExchange $\leftarrow \emptyset$
    for Each unlocated facility $i_2 \notin$ Solution **do**
        $c(i_1^-, i_2^+) \leftarrow$ Cost of Solution when removing facility i_1 and inserting facility i_2
        $\Delta \leftarrow c - c(i_1^-, i_2^+)$
        if $\Delta \geq \overline{\Delta_{i_1}}$ and exchange is feasible **then**
            $\overline{\Delta_{i_1}} \leftarrow \Delta$
            bestExchange $\leftarrow i_2$
        end if
    end for
    if $\overline{\Delta_{i_1}} > \overline{\Delta}$ **then**
        $\overline{\Delta} \leftarrow \overline{\Delta_{i_1}}$
        bestRemoval $\leftarrow i_1$
        bestInsertion $\leftarrow i_2$
    end if
end for
if $\overline{\Delta} > 0$ **then**
    Solution $\leftarrow$ Solution $\cup$ bestInsertion $\setminus$ bestRemoval
end if
Return(Solution)

Algorithm 5 Remove(Solution)

```
1: c ← Cost of Solution
2: Δ̄ ← 0
3: bestRemoval ← ∅
4: for Each located facility i_1 ∈ Solution do
5:     c(i_1^-) ← Cost of Solution when removing facility i_1
6:     Δ ← c − c(i_1^-)
7:     if Δ ≥ Δ̄ and removal is feasible then
8:         Δ̄ ← Δ
9:         bestRemoval ← i_1
10:    end if
11: end for
12: if Δ̄ > 0 then
13:    Solution ← Solution \ bestRemoval
14: end if
15: Return( Solution )
```

Algorithm 6 Insert(Solution)

```
1: c ← Cost of Solution
2: Δ̄ ← 0
3: bestInsertion ← ∅
4: for Each unlocated facility i_2 ∉ Solution do
5:     c(i_2^+) ← Cost of Solution when inserting facility i_2
6:     Δ ← c − c(i_2^+)
7:     if Δ ≥ Δ̄ and insertion is feasible then
8:         Δ̄ ← Δ
9:         bestInsertion ← i_2
10:    end if
11: end for
12: if Δ̄ > 0 then
13:    Solution ← Solution ∪ bestInsertion
14: end if
15: Return( Solution )
```

3.2.4 Perturbation operator

Implementing the above operators often yields a local optimum. We therefore propose a perturbation operator in order to find a better solution. This operator randomly selects η located facilities while ensuring that the remaining solution is feasible, i.e., that all remaining located facilities are interconnected. Then, η new facilities are added to the remaining solution. To determine which facilities will be inserted, the insertion cost of each uninserted facility i is computed as the impact of adding it to the current solution, i.e., $\Delta = c - c(i^+)$, where c is the cost of the current solution and $c(i^+)$ is the cost of the solution obtained by inserting the potential facility. The facility for which the insertion yields the largest cost decrease is then inserted. Algorithm 7 provides the pseudo-code for the perturbation operator.

4 Computational results

The metaheuristic algorithm was implemented in C++ and was tested on two sets of median instances. All tests were performed on a iMac computer equipped with an Intel Core i5 processor (3.1GHz). In this section, we report the computational results. The mathematical models were solved with CPLEX 12.7.1. We impose a maximal computational time of one hour (3,600 seconds).

4.1 Instances

The set of instances contains 40 instances with 100 to 900 nodes taken from the OR-Library (http://people.brunel.ac.uk/~mastjjb/jeb/info.html). These instances have no fixed facility location cost ($g_i = 0$, $\forall i \in I$), and contain between five and 200 facilities (given in the instance). For these instances, we have tested different values of r from 25 to 150.

Algorithm 7 Perturb(Solution)

```
Define η as a user-defined parameter
RemovedFacilities ← ∅
NumberInsertions ← 0
do
    Choose at random a visited facility i1 ∈ Solution
    if Deleting i1 results in all remaining facilities being interconnected then
        Solution ← Solution \{i1}
        RemovedFacilities ← RemovedFacilities ∪ {i1}
    end if
while size of RemovedFacilities < η
do
    Δ̄ ← −∞
    bestInsertion ← ∅
    for Each unlocated facility i2 such that i2 ∉ Solution and i2 ∉ RemovedFacilities do
        c(i2) ← Cost of Solution when inserting facility i2
        Δ ← c − c(i2)
        if Δ ≥ Δ̄ and insertion is feasible then
            Δ̄ ← Δ
            bestInsertion ← i2
        end if
    end for
    if bestInsertion ≠ ∅ then
        Solution ← Solution ∪ bestInsertion
        NumberInsertions ← NumberInsertions + 1
    end if
while bestInsertion ≠ ∅ and NumberInsertions < η
Return( Solution )
```

The second class of instances contains benchmark simple plant location problem instances (http://www.math.nsc.ru/AP/benchmarks/UFLP/Engl/uflp_eucl_eng.html). There are 29 instances, each containing exactly 100 nodes. They have a fixed facility location cost ($g_i = 3000, \forall i \in I$), and there is no limit on the maximum number of facilities. For these instances, we have tested different values of r from 500 to 2,500.

4.2 Computational results for the p-median instances

Table 1 reports the results obtained with the different mathematical models, i.e., MPIF-2 and MPIF-3, for the p-median instances. We have not tested model MPIF-1 as it would require the implementation of a branch-and-cut algorithm. We report the name of the instance (*Inst.*), the maximal distance r, the optimal solution found z^*, and the total time in seconds (*Sec.*) needed by models MPIF-2 and MPIF-3 to solve the instance to optimality. In all tables, "–" means that the time limit has been reached. We do not report the results for instances 26 to 40, as no solution was found within the prescribed time limit. We also report the average for all instances solved by both models (*Average**), as well as the average overall instances (*Average all*), and the number of solved instances (*Nb solved*).

We can observe that model MPIF-2 solves all instances, compared with model MPIF-3 for which instance 12 with $r = 25$ was not solved within the prescribed time limit. Thus, even though model MPIF-3 seems faster on average than model MPIF-2, 44 seconds compared with 46 seconds, the average time for model MPIF-2 without instance 12 with $r = 25$ is 36 seconds, which is faster than model MPIF-3. Hence, for the p-median instances, we will compare our metaheuristic with model MPIF-2 which outperforms model MPIF-3.

4.3 Computational results for the Euclidean instances

Table 2 reports the results obtained with the different mathematical models. The results are reported as in Table 1. On average, model MPIF-3 is faster than model MPIF-2 for the Euclidean instances, 142 seconds on average compared to 248 seconds on average. In addition, 85 out of the 89 instances are solved with model MPIF-3, compared to 84 instances for model MPIF-2. Hence, for the Euclidean instances, we will compare our metaheuristic with model MPIF-3 which outperforms model MPIF-2.

Table 1: Computational results for the p-median instances obtained with CPLEX

Inst.	r	z^*	MPIF-2 Sec.	MPIF-3 Sec.
1	150	5,915	1	1
1	100	5,976	1	1
1	80	6,168	4	4
2	150	4,298	0	0
2	100	4,346	2	1
2	80	4,419	2	2
3	150	4,282	1	1
3	100	4,307	1	0
3	80	4,582	7	29
4	150	3,034	1	0
4	100	3,034	0	1
4	80	3,192	4	5
5	150	1,355	1	1
5	100	1,359	1	0
5	80	1,432	1	1
6	150	8,288	19	25
6	100	8,288	16	48
6	80	8,288	19	30
7	100	5,826	7	7
7	80	5,826	5	8
7	50	5,843	4	7
8	100	4,491	5	3
8	80	4,510	4	3
8	50	4,618	3	3
9	100	2,734	5	2
9	80	2,758	4	3
9	50	2,898	3	1
10	80	1,272	6	2
10	50	1,356	2	1
10	25	1,755	3	3
11	80	7,921	17	24
11	50	7,921	13	15
11	25	7,950	8	14
12	80	6,789	25	21
12	50	6,789	12	21
12	25	6,944	961	–
13	80	4,416	17	10
13	50	4,422	9	10
13	25	4,713	39	57
14	80	2,968	17	11
14	50	3,033	9	8
14	25	3,209	10	10
15	80	1,729	15	6
15	50	1,731	10	5
15	25	1,985	4	3
16	80	8,514	142	187
16	50	8,514	91	283
16	25	9,251	61	61
17	80	7,096	53	44
17	50	7,096	46	84
17	25	7,096	14	19
18	80	4,824	50	27
18	50	4,824	43	29
18	25	4,905	39	32
19	80	2,862	36	16
19	50	2,862	29	18
19	25	2,896	12	12
20	80	1,796	43	18
20	50	1,796	34	17
20	25	2,045	12	12
21	80	9,514	194	145
21	50	9,514	74	153
21	25	9,514	39	45
22	80	8,665	454	707
22	50	8,665	202	341
22	25	8,665	56	203
23	80	4,648	107	48
23	50	4,648	72	49
23	25	4,665	36	41
24	80	2,976	76	41
24	50	2,976	76	40
24	25	3,020	43	31
25	80	1,832	79	46
25	50	1,832	90	47
25	25	1,929	43	45
		Average*	36	44
		Average all	49	44
		Nb solved	75	74

4.4 Computational results for the metaheuristic with the p-median instances

Tables 3 and 4 present detailed results for the p-median instances. We report the name of the instance (*Inst.*), the number of nodes ($|N|$), the maximum number of locations (p), the best known solution value (*Best*), and the time in seconds needed by CPLEX to solve the instance to optimality (*Sec.*). When an instance is not solved optimally within the prescribed time limit, we do not report the time, and the best known solution value is in italic. It corresponds to the best solution found with our metaheuristic. We report for different size of the restricted candidate list, $\lambda = \{0, 5, 10, 1000\}$, the deviation ($\Delta(\%)$) from the best known solution value computed as $(z' - z^*)/(z^*)$, where z' is the value obtained with our heuristic and z^* is the value of the best known solution value, and the time in seconds needed by our heuristic (*Sec.*). Note that when $\lambda = 0$, the construction heuristic is equivalent to a pure greedy insertion algorithm and when $\lambda = 1000$, it is equivalent to a pure random insertion algorithm.

Table 2: Computational results for the Euclidean instances obtained with CPLEX

Inst.	r	z^*	MPIF-2 Sec.	MPIF-3 Sec.
111	2500	93,415	1	0
111	1500	98,400	–	1,639
111	500	341,409	0	0
211	2500	95,607	1	1
211	1500	99,473	2,203	377
211	500	316,046	0	2
311	2500	93,293	0	0
311	1500	99,043	631	83
311	500	368,330	0	1
411	2500	99,960	1	1
411	1500	103,611	555	899
411	500	392,389	0	2
511	2500	96,553	1	0
511	1500	100,105	285	561
511	500	396,534	0	2
611	2500	95,394	0	1
611	1500	100,227	210	182
611	500	309,889	0	1
711	2500	96,415	1	0
711	1500	101,922	–	952
711	500	347,487	0	0
811	2500	98,637	0	0
811	1500	102,212	66	78
811	500	300,847	0	0
911	2500	98,536	0	1
911	1500	103,560	3,252	1,096
911	500	346,100	0	0
1011	2500	93,406	1	0
1011	1500	96,916	467	213
1011	500	407,994	0	2
1111	2500	96,176	0	0
1111	1500	100,290	97	140
1111	500	310,036	0	0
1211	2500	96,262	1	1
1211	1500	100,607	538	275
1211	500	295,586	0	2
1311	2500	97,249	0	0
1311	1500	100,657	37	83
1311	500	329,996	0	0
1411	2500	93,734	1	0
1411	1500	98,112	246	145
1411	500	371,692	0	2
1511	2500	97,705	1	1
1511	1500	101,022	1,565	985
1511	500	431,443	0	0
1611	2500	96,218	0	0
1611	1500	102,405	2,067	–
1611	500	308,214	0	0
1711	2500	94,985	1	1
1711	1500	102,476	254	273
1711	500	431,999	0	1
1811	2500	97,949	0	0
1811	1500	102,572	125	76
1811	500	330,298	0	1
1911	2500	96,625	1	0
1911	1500	101,836	2,850	844
1911	500	344,443	0	1
2011	2500	94,449	1	1
2011	1500	100,263	571	171
2011	500	410,630	1	2
2111	2500	92,483	0	0
2111	1500	96,538	182	122
2111	500	246,917	0	1
2211	2500	95,594	1	1
2211	1500	–	–	–
2211	500	415,291	0	1
2311	2500	95,143	0	0
2311	1500	101,671	1,173	468
2311	500	267,682	0	0
2411	2500	94,064	1	0
2411	1500	99,372	497	741
2411	500	376,439	0	1
2511	2500	97,569	0	0
2511	1500	101,429	877	338
2511	500	435,475	0	0
2611	2500	94,352	1	0
2611	1500	99,709	1,480	406
2611	500	263,326	0	1
2711	2500	91,554	0	0
2711	1500	94,467	9	52
2711	500	316,372	1	5
2811	2500	97,935	0	0
2811	1500	102,398	328	472
2811	500	265,678	0	0
2911	2500	95,699	0	0
2911	1500	101,814	277	355
2911	500	353,600	0	0
		Average*	226	114
		Average all	248	142
		Nb solved	84	85

Our results first show that our metaheuristic provides good solutions. On average, these solutions deviate from the best known solution value by 0.30%, 0.26%, 0.32%, and 0.31% for $\lambda = \{0, 5, 10, 1000\}$, respectively, when $|N| \leq 500$, and deviate from the best known solution value by 0.04%, 0.11%, 0.10%, and 0.10% for $\lambda = \{0, 5, 10, 1000\}$, respectively, when $|N| \geq 600$. In addition, our heuristic is much faster than CPLEX. Within the prescribed time limit, CPLEX cannot solve instances with more than 600 nodes, but all other instances, except instance 4 with $r = 100$, are solved on average within 49 seconds. Our heuristic, solves all instances within less than 313 seconds. For the instances that are solved to optimality, it takes on average between five and 15 seconds, depending on the value of λ.

With respect to the total computation time, we observe that the time taken by our heuristic does not depend on the number of nodes in the instance, but rather on the number of located facilities (p). This can be explained by the fact that when the number of located facilities increases, more moves are performed by the local search operators. On the other hand, the total time remains very reasonable (on average between five and 74 seconds, depending on the value of λ and of N). In addition, the average time increases with λ. This can be explained by the initial solution obtained with the GRASP construction algorithm. Increasing λ adds more randomness to the initial construction algorithm; the initial solution is often better when λ is small. Because of the added randomness, more local search moves are performed, which results in an increased computational time. On the other hand, the added randomness can sometimes provide better solutions.

With respect to the solution value, our heuristic yields good results on average: for all instances the average deviation from the best known optimal solution for all instances solved to optimality ranges from 0.25% to 0.32%, and on all other instances, it ranges from 0.04% to 0.11%. On the other hand, there are some instances that seem harder to solve, in particular, for larger values of r, and a few instances where the number of facilities are larger seem to be more difficult (e.g., instances 15, 20 and 25).

In summary, for the p-median instances, the best two settings for our initial GRASP construction algorithm seem to be $\lambda = 0$ and $\lambda = 5$. For all instances with at most 500 nodes and solved optimally, when $\lambda = 5$, the algorithm tends to provide the best results and seems to be more stable than for the other tested values of λ. For this setting, the maximum deviation from the best known solution value is 2.11%, and is 0.26% on average with a standard deviation of 0.40%. In addition, the average computational time is six seconds, with a maximum of 52 seconds. For all instances with more than 500 nodes (not solved optimally), when $\lambda = 0$ the algorithm tends to yield the best results and seems to be more stable than for the other tested values of λ. For this setting, the maximum deviation from the best known solution value is 0.26%, and is 0.04% on average with a standard deviation of 0.08%. In addition, the average computational time is 22 seconds, with a maximum of 82 seconds. We conclude that the proposed heuristic provides good quality solutions within reasonable computational times.

Table 3: Computational results for the metaheuristic with the p-median instances with $|N| \leq 500$

						$\lambda = 0$		$\lambda = 5$		$\lambda = 10$		$\lambda = 1000$	
Inst.	$\|N\|$	p	r	Best	Sec.	Δ (%)	Sec.	Δ (%)	Sec.	Δ (%)	Sec.	Δ (%)	Sec.
1	100	5	150	5,915	1	0.00	0	0.00	0	0.00	0	0.00	0
1	100	5	100	5,976	1	0.00	0	0.00	0	0.00	0	0.00	0
1	100	5	80	6,168	4	0.00	0	2.11	0	1.69	0	1.69	0
2	100	10	150	4,298	–	0.00	0	0.00	0	0.00	0	0.51	0
2	100	10	100	4,346	2	0.00	0	0.00	0	0.00	0	0.00	0
2	100	10	80	4,419	2	0.05	0	0.05	0	0.05	0	0.05	0
3	100	10	150	4,282	1	0.00	0	0.00	0	0.00	0	0.00	0
3	100	10	100	4,307	1	0.00	0	0.16	0	0.56	0	0.00	0
3	100	10	80	4,582	7	4.45	0	0.00	0	4.54	0	0.00	0
4	100	20	150	3,034	1	0.00	0	0.53	0	0.40	0	0.13	0
4	100	20	100	*3,034*	–	0.00	0	0.00	0	0.40	0	0.00	0
4	100	20	80	3,192	4	0.53	0	0.53	0	0.00	0	0.53	0
5	100	33	150	1,355	1	0.00	0	0.00	0	0.00	0	0.00	0
5	100	33	100	1,359	1	0.00	0	0.00	0	0.00	0	0.00	0
5	100	33	80	1,432	1	0.70	0	0.00	0	0.70	0	1.82	0
6	200	5	150	8,288	19	0.00	0	0.00	0	0.00	0	0.00	0
6	200	5	100	8,288	16	0.34	0	0.00	0	0.00	0	0.00	0
6	200	5	80	8,288	19	0.00	0	0.00	0	0.00	0	0.00	0
7	200	10	100	5,826	7	0.00	0	0.02	0	0.00	0	0.02	0
7	200	10	80	5,826	5	0.00	0	0.02	0	0.00	0	0.00	0
7	200	10	50	5,843	4	0.09	0	0.00	0	0.00	0	0.00	0
8	200	20	100	4,491	5	0.00	0	0.00	0	0.20	0	0.49	1
8	200	20	80	4,510	4	0.71	0	0.71	0	0.91	0	0.51	1
8	200	20	50	4,618	3	0.00	0	0.00	0	0.00	0	0.00	1
9	200	40	100	2,734	5	0.48	1	0.15	1	0.48	1	1.21	2
9	200	40	80	2,758	4	0.80	1	0.47	1	0.00	1	0.36	2

Table 3 . . . continued

						$\lambda = 0$		$\lambda = 5$		$\lambda = 10$		$\lambda = 1000$	
Inst.	$\|N\|$	p	r	Best	Sec.	Δ (%)	Sec.	Δ (%)	Sec.	Δ (%)	Sec.	Δ (%)	Sec.
9	200	40	50	2,898	3	0.21	1	0.35	1	0.07	1	0.14	2
10	200	67	80	1,272	6	0.55	1	0.55	2	0.00	3	0.63	4
10	200	67	50	1,356	2	0.59	1	0.15	2	0.59	2	0.74	4
10	200	67	25	1,755	3	2.62	4	0.74	4	2.68	5	0.80	8
11	300	5	80	7,921	17	0.00	0	0.00	0	0.00	0	0.00	0
11	300	5	50	7,921	13	0.00	0	0.00	0	0.00	0	0.00	0
11	300	5	25	7,950	8	0.00	0	0.75	0	0.75	0	0.00	0
12	300	10	80	6,789	25	0.00	0	0.00	0	0.00	0	0.00	0
12	300	10	50	6,789	12	0.00	0	0.00	0	0.00	0	0.00	0
12	300	10	25	6,944	961	0.27	0	0.27	0	0.00	0	0.00	0
13	300	30	80	4,416	17	0.00	1	0.00	2	0.00	2	0.00	3
13	300	30	50	4,422	9	0.00	1	0.09	2	0.00	2	0.00	3
13	300	30	25	4,713	39	0.23	2	0.23	2	0.23	2	0.23	3
14	300	60	80	2,968	17	0.03	5	0.00	7	0.00	6	0.51	9
14	300	60	50	3,033	9	0.69	2	0.36	5	0.00	5	0.40	7
14	300	60	25	3,209	10	0.16	7	1.15	6	0.93	7	0.93	10
15	300	100	80	1,729	15	0.52	5	0.75	6	0.46	9	0.64	23
15	300	100	50	1,731	10	0.46	4	0.87	6	0.75	8	1.10	15
15	300	100	25	1,985	4	0.76	10	1.26	17	0.86	15	0.60	34
16	400	5	80	8,514	142	0.00	0	0.00	0	0.00	0	0.00	0
16	400	5	50	8,514	91	0.00	0	0.00	0	0.00	0	0.00	0
16	400	5	25	9,251	61	0.00	0	0.00	0	0.00	0	0.00	0
17	400	10	80	7,096	53	0.06	1	0.00	1	0.00	1	0.00	1
17	400	10	50	7,096	46	0.06	1	0.00	1	0.00	1	0.00	1
17	400	10	25	7,096	14	0.06	0	0.00	1	0.00	1	0.00	1
18	400	40	80	4,824	50	0.06	3	0.06	5	0.06	4	0.35	8
18	400	40	50	4,824	43	0.06	3	0.06	5	0.06	5	0.06	9
18	400	40	25	4,905	39	0.14	3	0.06	5	0.47	5	0.33	7
19	400	80	80	2,862	36	0.17	11	0.49	13	0.31	12	0.35	27
19	400	80	50	2,862	29	0.17	9	0.49	11	0.31	11	0.35	27
19	400	80	25	2,896	12	0.28	9	0.17	10	0.41	11	0.31	24
20	400	133	80	1,796	43	0.72	24	0.67	23	0.11	33	0.61	68
20	400	133	50	1,796	34	0.72	20	0.50	20	0.11	28	0.28	58
20	400	133	25	2,045	12	1.32	33	1.56	33	1.56	40	1.61	72
21	500	5	80	9,514	194	0.00	0	0.00	0	0.00	0	0.00	0
21	500	5	50	9,514	74	0.00	0	0.00	0	0.00	0	0.00	0
21	500	5	25	9,514	39	0.00	0	0.00	0	0.00	0	0.00	0
22	500	10	80	8,665	454	0.00	1	0.00	1	0.00	1	0.00	2
22	500	10	50	8,665	202	0.00	1	0.00	1	0.00	1	0.00	2
22	500	10	25	8,665	56	0.00	1	0.00	1	0.00	1	0.00	1
23	500	50	80	4,648	107	0.13	7	0.19	11	0.00	11	0.00	22
23	500	50	50	4,648	72	0.13	7	0.19	11	0.00	11	0.26	20
23	500	50	25	4,665	36	0.00	6	0.09	8	0.06	10	0.45	17
24	500	100	80	2,976	76	0.17	19	0.44	29	0.64	35	0.77	67
24	500	100	50	2,976	76	0.17	18	0.44	28	0.64	34	0.37	66
24	500	100	25	3,020	43	0.20	16	0.26	23	0.20	33	0.30	61
25	500	167	80	1,832	79	0.55	42	0.44	51	0.60	50	1.09	156
25	500	167	50	1,832	90	0.55	39	0.44	49	0.60	48	0.66	136
25	500	167	25	1,929	43	1.71	35	0.93	52	0.93	55	1.40	139
				Average	49	0.30	5	0.26	6	0.32	7	0.31	15
				Standard deviation	124	0.64	9	0.40	12	0.67	13	0.44	32
				Maximum	961	4.45	42	2.11	52	4.54	55	1.82	156

Table 4: Computational results for the metaheuristic with the p-median instances with $|N| \geq 600$

						$\lambda = 0$		$\lambda = 5$		$\lambda = 10$		$\lambda = 1000$	
Inst.	$\|N\|$	p	r	Best	Sec.	Δ (%)	Sec.	Δ (%)	Sec.	Δ (%)	Sec.	Δ (%)	Sec.
26	600	5	80	*10,067*	–	0.00	0	0.00	0	0.81	0	0.00	1
26	600	5	50	*10,067*	–	0.00	0	0.00	0	0.81	0	0.00	1
26	600	5	25	*10,067*	–	0.00	0	0.00	0	0.00	0	0.00	0
27	600	10	80	*8,388*	–	0.00	1	0.00	2	0.00	2	0.00	2
27	600	10	50	*8,388*	–	0.00	1	0.00	2	0.00	2	0.00	2
27	600	10	25	*8,388*	–	0.00	1	0.00	2	0.00	2	0.00	2
28	600	60	80	*4,499*	–	0.00	16	0.02	23	0.13	23	0.31	38
28	600	60	50	*4,522*	–	0.00	16	0.18	21	0.13	24	0.22	38
28	600	60	25	*4,522*	–	0.00	14	0.18	17	0.13	21	0.27	35
29	600	120	80	*3,052*	–	0.23	38	0.07	46	0.03	66	0.29	131
29	600	120	50	*3,052*	–	0.23	37	0.07	45	0.03	64	0.23	136
29	600	120	25	*3,106*	–	0.10	32	0.42	37	0.03	50	0.00	112
30	600	200	80	*2,001*	–	0.00	68	0.55	82	0.20	88	0.30	313
30	600	200	50	*2,001*	–	0.00	66	0.55	80	0.20	87	0.10	277
30	600	200	25	*2,071*	–	0.00	52	0.53	70	0.53	70	0.29	242
31	700	5	80	*10,365*	–	0.00	1	0.00	1	0.00	1	0.00	1
31	700	5	50	*10,365*	–	0.00	1	0.00	1	0.00	1	0.00	1
31	700	5	25	*10,365*	–	0.00	1	0.00	1	0.00	1	0.00	1
32	700	10	80	*9,515*	–	0.00	2	0.00	2	0.00	3	0.00	3
32	700	10	50	*9,515*	–	0.00	2	0.00	2	0.00	3	0.00	3
32	700	10	25	*9,515*	–	0.00	2	0.00	3	0.00	2	0.00	3
33	700	70	80	*4,703*	–	0.17	30	0.15	35	0.26	41	0.00	83
33	700	70	50	*4,710*	–	0.02	30	0.00	36	0.11	41	0.23	74
33	700	70	25	*4,706*	–	0.11	26	0.08	33	0.00	29	0.08	69
34	700	140	80	*3,019*	–	0.26	82	0.33	90	0.03	116	0.20	276
34	700	140	50	*3,019*	–	0.26	81	0.33	90	0.03	115	0.30	252
34	700	140	25	*3,075*	–	0.07	50	0.46	72	0.00	93	0.07	201
35	800	5	80	*10,594*	–	0.00	1	0.00	1	0.00	1	0.00	1
35	800	5	50	*10,594*	–	0.00	1	0.00	1	0.00	1	0.00	1
35	800	5	25	*10,594*	–	0.00	1	0.00	1	0.00	1	0.00	1
36	800	10	80	*10,051*	–	0.12	2	0.12	3	0.12	4	0.00	3
36	800	10	50	*10,051*	–	0.12	2	0.12	3	0.12	4	0.00	3
36	800	10	25	*10,051*	–	0.12	2	0.12	3	0.12	4	0.12	4
37	800	80	80	*5,066*	–	0.00	37	0.16	65	0.14	66	0.10	143
37	800	80	50	*5,066*	–	0.00	37	0.16	65	0.14	66	0.53	121
37	800	80	25	*5,070*	–	0.00	36	0.10	59	0.34	64	0.02	115
38	900	5	80	*11,494*	–	0.00	1	0.00	1	0.00	1	0.00	1
38	900	5	50	*11,494*	–	0.00	1	0.00	1	0.00	1	0.00	1
38	900	5	25	*11,494*	–	0.00	1	0.00	1	0.00	1	0.00	1
39	900	10	80	*9,628*	–	0.01	3	0.00	4	0.00	4	0.00	5
39	900	10	50	*9,628*	–	0.01	3	0.00	4	0.00	4	0.00	5
39	900	10	25	*9,628*	–	0.01	3	0.00	4	0.00	4	0.00	5
40	900	90	80	*5,151*	–	0.04	67	0.00	89	0.06	109	0.25	209
40	900	90	50	*5,151*	–	0.04	69	0.00	89	0.06	108	0.10	210
40	900	90	25	*5,155*	–	0.06	61	0.23	72	0.04	99	0.33	189
				Average		0.04	22	0.11	28	0.10	33	0.10	74
				Standard deviation		0.08	26	0.17	33	0.19	40	0.14	97
				Maximum		0.26	82	0.55	90	0.81	116	0.53	313

4.5 Computational results for the metaheuristic with the Euclidean instances

Table 5 presents detailed results for the Euclidean instances. The results are reported as in Tables 3 and 4. For these instances, we do not report the value of $|N|$ because each instance contains exactly 100 nodes and we do not report a value of p as there is no limit on the number of located facilities, but rather a cost of facility location as explained previously.

Again, our metaheuristic provides good solutions. On average, these solutions deviate from 0.51%, 0.53%, 0.55%, and 0.70% from the best known solution values with $\lambda = \{0, 5, 10, 1000\}$, respectively. In addition, our heuristic is much faster than CPLEX which cannot solve two instances (instances 1611 and 2211 with $r = 1500$) within the prescribed time limit, while all other instances are solved within 142 seconds on average. When $r = 1500$, all instances are harder to solve and take more time, on average 445 seconds. Our heuristic solves all instances within less than one second, independently on the values of r and λ.

With respect to the solution value, our heuristic provides good results on average: for all instances the average deviation from the best known optimal solution ranges from 0.51% to 0.70%. On the other hand, there are a few instances that seem harder to solve, in particular, when $r = 1500$ (e.g., instances 311, 511, and 711).

For the Euclidean instances, the best two settings for our initial GRASP construction algorithm seem to be $\lambda = 0$ and $\lambda = 10$. When $\lambda = 0$, the maximum deviation from the best known solution value is 4.34%, and it is 0.51% on average with a standard deviation of 0.88%. In addition, all instances are solved within less than one second. When $\lambda = 10$, the maximum deviation from the best known solution value is 3.42%, and it is 0.55% on average with a standard deviation of 0.79%. Again, all instances are solved within less than one second. When $\lambda = 10$, the solution quality seems to be more stable for the Euclidean instances. In conclusion, the proposed heuristic provides good quality solutions within reasonable computational times.

Table 5: Computational results for the metaheuristic with the Euclidean instances

				$\lambda = 0$		$\lambda = 5$		$\lambda = 10$		$\lambda = 1000$	
Inst.	r	Best	Sec.	Δ (%)	Sec.	Δ (%)	Sec.	Δ (%)	Sec.	Δ (%)	Sec.
111	2500	93,415	0	0.00	0	0.00	0	0.00	0	0.07	0
111	1500	98,400	1,639	2.74	0	1.42	0	0.53	0	1.42	0
111	500	341,409	0	0.00	0	0.00	0	0.00	0	0.00	0
211	2500	95,607	1	0.00	0	0.02	0	0.01	0	0.60	0
211	1500	99,473	377	0.01	0	0.80	0	0.92	0	2.90	0
211	500	316,046	2	0.00	0	0.00	0	0.00	0	0.00	0
311	2500	93,293	0	0.00	0	0.00	0	0.06	0	0.00	0
311	1500	99,043	83	4.34	0	4.21	0	1.92	0	2.77	1
311	500	368,330	1	0.00	0	0.00	0	0.00	0	0.00	0
411	2500	99,960	1	0.08	0	0.24	0	0.50	0	0.62	0
411	1500	103,611	899	1.19	0	1.75	0	2.14	0	3.26	0
411	500	392,389	2	0.00	0	0.00	0	0.00	0	0.00	0
511	2500	96,553	0	0.10	0	0.19	0	0.29	0	0.29	0
511	1500	100,105	561	1.38	0	2.02	0	3.42	0	3.18	0
511	500	396,534	2	0.00	0	0.00	0	0.00	0	0.00	0
611	2500	95,394	1	0.35	0	0.35	0	0.44	0	0.53	0
611	1500	100,227	182	0.13	0	1.33	0	0.04	0	1.75	0
611	500	309,889	1	0.00	0	0.00	0	0.00	0	0.00	0
711	2500	96,415	0	0.00	0	0.00	0	0.00	0	0.00	0
711	1500	101,922	952	1.64	0	0.13	0	0.97	0	0.64	0
711	500	347,487	0	0.00	0	0.00	0	0.00	0	0.00	0
811	2500	98,637	0	0.46	0	0.00	0	0.46	0	0.46	0
811	1500	102,212	78	1.10	0	0.37	0	0.57	0	0.41	0
811	500	300,847	0	0.00	0	0.00	0	0.00	0	0.00	0
911	2500	98,536	1	0.13	0	0.59	0	0.66	0	0.20	0
911	1500	103,560	1,096	1.18	0	1.54	0	0.67	0	0.82	0
911	500	346,100	0	0.00	0	0.00	0	0.00	0	0.00	0
1011	2500	93,406	0	0.00	0	0.00	0	0.11	0	0.11	0
1011	1500	96,916	213	1.46	0	1.08	0	1.78	0	0.06	0
1011	500	407,994	2	0.00	0	0.00	0	0.00	0	0.00	0
1111	2500	96,176	0	0.00	0	0.48	0	0.02	0	0.49	0
1111	1500	100,290	140	1.11	0	1.01	0	1.94	0	2.64	1
1111	500	310,036	0	0.00	0	0.00	0	0.00	0	0.00	0
1211	2500	96,262	1	0.00	0	0.00	0	0.00	0	0.00	0
1211	1500	100,607	275	1.23	0	1.35	0	1.82	0	1.67	0
1211	500	295,586	2	0.00	0	0.00	0	0.00	0	0.00	0

Table 5 . . . continued

				$\lambda = 0$		$\lambda = 5$		$\lambda = 10$		$\lambda = 1000$	
Inst.	r	Best	Sec.	Δ (%)	Sec.	Δ (%)	Sec.	Δ (%)	Sec.	Δ (%)	Sec.
1311	2500	97,249	0	0.00	0	0.52	0	0.52	0	1.02	0
1311	1500	100,657	83	3.51	0	1.90	0	1.12	0	1.15	0
1311	500	329,996	0	0.00	0	0.00	0	0.00	0	0.00	0
1411	2500	93,734	0	0.00	0	0.00	0	0.00	0	0.00	0
1411	1500	98,112	145	0.20	0	1.62	0	2.58	0	2.71	0
1411	500	371,692	2	0.00	0	0.00	0	0.00	0	0.00	0
1511	2500	97,705	1	0.49	0	0.00	0	0.36	0	0.00	0
1511	1500	101,022	985	3.19	0	3.06	0	2.20	0	2.68	0
1511	500	431,443	0	0.00	0	0.00	0	0.00	0	0.00	0
1611	2500	96,218	0	0.44	0	0.49	0	0.75	0	0.59	0
1611	1500	*102,405*	–	2.81	0	0.96	0	2.19	0	2.61	1
1611	500	308,214	0	0.00	0	0.00	0	0.00	0	0.00	0
1711	2500	94,985	1	0.15	0	0.19	0	0.02	0	0.00	0
1711	1500	102,476	273	0.56	0	0.53	0	0.88	0	0.86	1
1711	500	431,999	1	0.00	0	0.00	0	0.00	0	0.00	0
1811	2500	97,949	0	0.50	0	0.00	0	0.00	0	0.55	0
1811	1500	102,572	76	0.39	0	1.30	0	1.30	0	0.47	0
1811	500	330,298	1	0.00	0	0.00	0	0.00	0	0.00	0
1911	2500	96,625	0	0.32	0	1.12	0	0.32	0	0.32	0
1911	1500	101,836	844	0.15	0	0.00	0	0.81	0	0.00	0
1911	500	344,443	1	0.00	0	0.00	0	0.00	0	0.00	0
2011	2500	94,449	1	0.34	0	0.63	0	0.72	0	0.72	0
2011	1500	100,264	171	0.36	0	0.65	0	0.44	0	0.44	1
2011	500	410,630	2	0.00	0	0.00	0	0.00	0	0.00	0
2111	2500	92,483	0	0.00	0	0.00	0	0.00	0	0.00	0
2111	1500	96,538	122	1.48	0	1.27	0	1.84	0	1.88	0
2111	500	246,917	1	0.00	0	0.00	0	0.00	0	0.00	0
2211	2500	95,594	1	0.41	0	0.55	0	0.00	0	0.68	0
2211	1500	*103,998*	–	0.81	0	0.59	0	0.00	0	0.77	1
2211	500	415,291	1	0.00	0	0.00	0	0.00	0	0.00	0
2311	2500	95,143	0	0.02	0	0.02	0	0.65	0	0.65	0
2311	1500	101,671	468	0.05	0	1.98	0	1.52	0	1.98	0
2311	500	267,682	0	0.00	0	0.00	0	0.00	0	0.00	0
2411	2500	94,064	0	0.00	0	0.35	0	0.52	0	0.52	0
2411	1500	99,372	741	1.17	0	1.22	0	1.06	0	1.57	0
2411	500	376,439	1	0.00	0	0.00	0	0.00	0	0.00	0
2511	2500	97,569	0	0.01	0	0.00	0	0.44	0	0.01	0
2511	1500	101,429	338	0.65	0	2.02	0	1.30	0	1.00	1
2511	500	435,475	0	0.00	0	0.00	0	0.00	0	0.00	0
2611	2500	94,352	0	0.72	0	0.64	0	0.00	0	0.64	0
2611	1500	99,709	406	1.01	0	0.85	0	0.46	0	0.60	0
2611	500	263,326	1	0.00	0	0.00	0	0.00	0	0.00	0
2711	2500	91,554	0	0.53	0	0.00	0	0.49	0	0.49	0
2711	1500	94,467	52	0.98	0	2.39	0	0.35	0	4.01	0
2711	500	316,372	5	0.00	0	0.00	0	0.00	0	0.00	0
2811	2500	97,935	0	0.36	0	0.02	0	0.29	0	0.29	0
2811	1500	102,398	472	2.05	0	1.01	0	2.23	0	2.28	1
2811	500	265,678	0	0.00	0	0.00	0	0.00	0	0.00	0
2911	2500	95,699	0	0.00	0	0.59	0	0.61	0	0.61	0
2911	1500	101,814	355	2.24	0	1.12	0	2.72	0	3.53	0
2911	500	353,600	0	0.00	0	0.00	0	0.00	0	0.00	0
		Average	142	0.51	0	0.53	0	0.55	0	0.70	0
		Standard deviation	301	0.88	0	0.79	0	0.79	0	1.00	0
		Maximum	1,639	4.34	0	4.21	0	3.42	0	4.01	1

5 Conclusions

We have introduced and modeled median and covering problems with interconnected facilities. Such problems arise, for example, in contexts where forest fire-fighters must be within a prescribed distance of a central office or of each other in order to be able to communicate by short-range radio. We have proposed three models for median problems and three others for covering models. We have then developed a greedy randomized adaptive search procedure (GRASP) applicable to both classes of problems. Extensive tests performed on two formulations of the median problems show that the relative performance of the models depends on the instance type. With a few exceptions, the GRASP consistently yields optimal or near-optimal solutions on instances for which the optimum is known.

References

Church, R., ReVelle, C., 1974. The maximal covering location problem. Papers of the Regional Science Association 32, 101–118.

Contreras, I., Fernndez, E., 2012. General network design: A unified view of combined location and network design problems. European Journal of Operational Research 219, 680–697.

Contreras, I., Fernndez, E., Marn, A., 2010. The tree of hubs location problem. European Journal of Operational Research 202, 390–400.

Demaine, E., Hajiaghayi, M., Sayedi-Roshkhar, A., Zadinoghaddam, M., 2009. Minimizing movement. ACM Transactions on Algorithms 5(3), 28.

Feo, T., Resende, M., 1995. Greedy randomized adaptive search procedures. Journal of Global Optimization 6(2), 109–133.

Hakimi, S., 1964. Optimum location of switching centers and the absolute centers and the medians of a graph. Operations Research 12, 450–459.

Kolen, A., Tamir, A., 1990. Covering problems. In: Mirchandani, P., Francis, R. (Eds.), Discrete Location theory. Wiley, New York, pp. 1–2.

Kuehn, A., Hamburger, M., 1963. A heuristic program for locating warehouses. Management Science 9, 645–666.

Landete, M., Marn, A., 2014. Looking for edge-equitable spanning trees. Operations Research 41, 44–52.

Laporte, G., Rodrguez Martn, I., 2007. Locating a cycle in a transportation or a telecommunications network. Networks 50(1), 92–108.

Resende, M., Ribeiro, C., 2010. Greedy randomized adaptive search procedures: Advances, hybridizations, and applications. In: Handbook of Metaheuristics. Springer, pp. 283–319.

Resende, M., Werneck, R., 2004. A hybrid heuristic for the p-median problem. Journal of Heuristics 10(1), 59–88.