INFORMS Journal on Computing

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

A New Formulation and Resolution Method for the p-Center Problem

Sourour Elloumi, Martine Labbé, Yves Pochet,

To cite this article:
Sourour Elloumi, Martine Labbé, Yves Pochet, (2004) A New Formulation and Resolution Method for the p-Center Problem. INFORMS Journal on Computing 16(1):84-94. http://dx.doi.org/10.1287/ijoc.1030.0028

INFORMS Journal on Computing
Vol. 16, No. 1, Winter 2004, pp. 84–94
ISSN 0899-1499 | EISSN 1526-5528 | 04 | 1601 | 0084

DOI 10.1287/ijoc.1030.0028

A New Formulation and Resolution Method for the *p*-Center Problem

Sourour Elloumi
CEDRIC-CNAM, 292 Rue Saint-Martin, F-75141 Paris Cedex 03, France, elloumi@cnam.fr

Martine Labbé
Service d'Optimisation, Institut de Statistique et de Recherche Opérationnelle, Université Libre de Bruxelles, Boulevard du Triomphe CP 210/01, B-1050 Bruxelles, Belgium, mlabbe@smg.ulb.ac.be

Yves Pochet
CORE and IAG, Université catholique de Louvain, Voie du Roman Pays 34, B-1348 Louvain-La-Neuve, Belgium, pochet@core.ucl.ac.be

The *p*-center problem consists of choosing p facilities among a set of M possible locations and assigning N clients to them in order to minimize the maximum distance between a client and the facility to which it is allocated. We present a new integer linear programming formulation for this min-max problem with a polynomial number of variables and constraints, and show that its LP relaxation provides a lower bound tighter than the classical one. Moreover, we show that an even better lower bound LB^*, obtained by keeping the integrality restrictions on a subset of the variables, can be computed in polynomial time by solving at most $O(\log_2(NM))$ linear programs, each having N rows and M columns. We also show that, when the distances satisfy triangle inequalities, LB^* is at least one third of the optimal value. Finally, we use LB^* in an exact solution method and report extensive computational results on test problems from the literature. For instances where the triangle inequalities are satisfied, our method outperforms the running time of other recent exact methods by an order of magnitude. Moreover, it is the first one to solve large instances of size up to $N = M = 1{,}817$.

Key words: min-max objective; facility location; *p*-center; mathematical programming
History: Accepted by Jan Karel Lenstra; received November 2001; revised June 2002, December 2002; accepted December 2002.

1. Introduction

Let N be the number of clients, called $C_1, C_2, \ldots, C_N$, let M be the number of potential sites or facilities, called $F_1, F_2, \ldots, F_M$, and let d_{ij} be the distance from C_i to F_j. The *p*-center problem consists of opening at most p facilities and assigning each client to its closest open facility, in order to minimize the radius, i.e., the maximum distance between a client and its closest facility. More formally, the problem is to find a subset S of $\{1, \ldots, M\}$ such that:

- $|S| \le p$
- $radius(S) = \max_{i=1,\ldots,N}(\min_{j\in S} d_{ij})$ is minimized.

Many applications of this problem arise in the public sector, such as locating fire stations or ambulance depots; see Daskin (1995) and Marinov and ReVelle (1995).

This problem covers the case where a nonnegative weight w_i is associated with each client C_i and where the radius of a solution S is $\max_{i=1,\ldots,N}(w_i \min_{j\in S} d_{ij})$. To consider such weights in an unweighted instance, it suffices to modify the distances by setting $d'_{ij} = w_i d_{ij}$. Note, however, that such a transformation may not preserve triangle inequalities and will thus make the problem harder to solve.

The 1-center and the $(M-1)$-center problems are trivial and, for a fixed p, the *p*-center problem is polynomial since no more than M^p different ways to choose locations exist. In Megiddo et al. (1981), the *p*-center problem is proved to be polynomial if the d_{ij}'s represent distances in a tree network (path distances in the underlying tree) but in general, the *p*-center problem is NP-hard; see Kariv and Hakimi (1979) and Masuyama et al. (1981).

Many authors consider the particular case where the facilities are to be located among the clients, i.e., $N = M$, and distances are symmetric and satisfy triangle inequalities. We call this particular case the *symmetric p-center* problem.

Many exact solution techniques are based on the idea that the optimal radius can be computed by solving different auxiliary problems. For instance, a strong relation exists between the *p*-center problem and the *set-covering* auxiliary problem, and with the *dominating-set* auxiliary problem in the symmetric case. To see these relations, consider the following

problem. For a given radius, minimize the number of facilities needed to serve all clients within this radius. This auxiliary subproblem is a minimal-cardinality dominating-set problem in the symmetric case; see Hochbaum and Shmoys (1985). In the general case, this problem is a minimal-cardinality set-covering problem; see ReVelle et al. (1971). Daskin (2000) considers a different auxiliary subproblem, consisting of maximizing the number of covered clients by no more than p facilities, within a given radius. He calls this the *maximal covering* problem. Ilhan and Pinar (2001) consider yet a different subproblem that is a feasibility problem, checking whether or not it is possible to serve all customers with no more than p facilities within a given radius.

In an exact solution approach, different authors solve the p-center problem by finding the smallest radius such that the optimal solution of the considered auxiliary problem gives a feasible solution to the p-center problem, i.e., no more than p facilities are needed. In an approximate solution approach, several authors use a heuristic algorithm to solve the auxiliary problem and use it to search the smallest radius such that the heuristic solution gives a feasible solution to the p-center problem. For example, Hochbaum and Shmoys (1985) consider the symmetric p-center problem and use a greedy heuristic to build a dominating set in the square of the initial graph. Their algorithm leads to a solution to the p-center problem with worst-case ratio 2. According to the results of Hsu and Nemhauser (1979), this 2-approximation algorithm is optimal in the sense that, unless $P = NP$, it is impossible to find a δ-approximation polynomial time algorithm with $\delta < 2$. Hochbaum and Shmoys (1986) generalize these results to obtain a 3-approximation algorithm when distances satisfy the triangle inequalities.

In this paper we introduce a new formulation for the p-center problem that is based on its relationship with the set-covering problem. In the next section we first recall the classical formulation (PC). Then we present our new formulation ($PC-SC$). We show that the LP relaxation of ($PC-SC$) provides a better lower bound to the p-center optimal value than does the LP relaxation of (PC). Furthermore, we show, with a few experimental results, how formulation ($PC-SC$) takes advantage of the knowledge of tight lower and upper bounds. In §3 we present a method to compute a lower bound LB^* and an upper bound UB^*. We define LB^* as the optimal value of a MILP obtained by relaxing the integrality restrictions of some variables of ($PC-SC$). Nevertheless, we show that LB^* can be computed by a polynomial-time algorithm consisting of solving a series of linear set-covering problems. UB^* is simply the best solution to the p-center problem obtained while computing LB^*. In §4 we show that our lower bound LB^* cannot be less than one third of the optimal radius when the distances satisfy the triangle inequalities, and not less than one half of the optimal radius for the symmetric p-center. In §5 we describe our exact algorithm and we report experimental results. Throughout this paper, our computational results are mainly based on instances coming either from OR-Lib or from TSPLIB. We give a more precise description of these instances in §5. Furthermore, we use a PC with 384 MB of RAM and a 400 MHz PentiumII processor, C++, and CPLEX 7.0.

In §6, we outline a generalization of our formulation and lower bound LB^* to a generalization of the p-center problem. We conclude in §7 with some possible extensions of this approach to related problems.

2. The Classical Formulation (PC) and the New Formulation ($PC-SC$)

The p-center problem is usually formulated by (1)–(6) (see, for example, Daskin 1995). For any feasible solution (x, y, z), $y_j = 1$ if and only if facility F_j is open, $x_{ij} = 1$ if and only if client C_i is assigned to facility F_j, and z is an upper bound on the radius of the feasible solution.

Formulation (PC)

$$\min z \tag{1}$$

subject to

$$\sum_{j=1}^{M} y_j \le p \tag{2}$$

$$\sum_{j=1}^{M} x_{ij} = 1 \quad i = 1, \ldots, N \tag{3}$$

$$x_{ij} \le y_j \quad i = 1, \ldots, N;\ j = 1, \ldots, M \tag{4}$$

$$\sum_{j=1}^{M} d_{ij} x_{ij} \le z \quad i = 1, \ldots, N \tag{5}$$

$$x_{ij}, y_j \in \{0, 1\} \quad i = 1, \ldots, N;\ j = 1, \ldots, M. \tag{6}$$

Constraint (2) limits the number of open facilities to at most p; constraints (3) assign each client to exactly one facility; constraints (4) forbid the assignment of a client to a closed facility; and constraints (5) force z to be larger than the distance from any client to the assigned facility. In an optimal solution, $z = \max_{i=1,\ldots,N} \sum_{j=1}^{M} d_{ij} x_{ij}$ is the optimal radius.

Our new formulation of the p-center problem is based on its well-known relation to the set-covering problem. Let D^{min} (resp. $D^{\max}$) be the smallest (resp. largest) value in the distance matrix, and let $D^{\min} = D^0 < D^1 < D^2 < \cdots < D^K = D^{\max}$ be the sorted different values in that matrix. We now present a new formulation ($PC-SC$) using variables y_j, $j = 1, \ldots, M$, and z^k, $k = 1, \ldots, K$. In this formulation, for any feasible

solution (y, z), $y_j = 1$ if and only if facility F_j is open, and $z^k = 0$ only if it is possible to choose p facilities and cover all the clients within the radius D^{k-1}. In an optimal solution, note that $z^k = 0$ implies $z^{k+1} = \cdots = z^K = 0$ and the objective function is strictly lower than D^k. Similarly, $z^k = 1$ implies $z^{k-1} = \cdots = z^1 = 1$.

Formulation $(PC - SC)$

$$\min D^0 + \sum_{k=1}^{K} (D^k - D^{k-1}) z^k \tag{7}$$

$$\text{subject to} \quad \sum_{j=1}^{M} y_j \geq 1 \tag{8}$$

$$\sum_{j=1}^{M} y_j \leq p \tag{9}$$

$$z^k + \sum_{j: d_{ij} < D^k} y_j \geq 1 \qquad i = 1, \ldots, N, \ k = 1, \ldots, K \tag{10}$$

$$z^k \in \{0, 1\} \quad k = 1, \ldots, K \tag{11}$$

$$y_j \in \{0, 1\} \quad j = 1, \ldots, M. \tag{12}$$

Constraint (8) discards solutions with no open facility. This constraint is automatically satisfied in (PC) and is needed in order to compare (PC) and $(PC - SC)$. Constraint (9) is the same as (2). Note that the coefficients of the objective function (7) are positive. Hence, in an optimal solution, constraints (10) say that, for a given k, $z^k = 0$, if and only if all clients can be served at a distance strictly lower than D^k. Therefore, when $z^k = 1$, a distance $(D^k - D^{k-1})$ is added to the radius in the objective (7).

We investigate now the relation between $(PC - SC)$ and some auxiliary set-covering problems. We can have $z^k = 0$ in an optimal solution to $(PC - SC)$ if and only if the optimal value of the set-covering problem $SC_{D^{k-1}}$ is less than or equal to p.

Problem (SC_δ)

$$\begin{aligned} &\min \sum_{j=1}^{M} y_j \\ &\text{subject to} \quad \sum_{j: d_{ij} \leq \delta} y_j \geq 1 \quad i = 1, \ldots, N \\ &\qquad\qquad y_j \in \{0, 1\} \quad j = 1, \ldots, M \end{aligned} \tag{13}$$

We define $v(SC_\delta)$ as the optimal objective-function value of SC_δ, where $v(SC_\delta) = \infty$ when SC_δ contains no feasible solution. Since a feasible solution for (SC_{D^k}) is also feasible for $(SC_{D^{k-1}})$, we have that $v(SC_{D^0}) \geq v(SC_{D^1}) \geq \cdots \geq v(SC_{D^K})$. Now, we make precise the relation between the optimal solutions of problems SC_{D^k} and $(PC - SC)$. Let k^* be the smallest k in $\{0, \ldots, K\}$ such that $v(SC_{D^k}) \leq p$, and let $\tilde{y}$ be an optimal solution to $SC_{D^{k^*}}$. Then an optimal solution $(\tilde{y}, \tilde{z})$ exists for $(PC - SC)$ where $\tilde{z}^1 = \cdots = \tilde{z}^{k^*} = 1$ and $\tilde{z}^{k^*+1} = \cdots = \tilde{z}^K = 0$. Therefore, solving $(PC - SC)$ can be done by determining the value of k^*, i.e., by solving at most $(K + 1)$ set-covering problems. Note that formulation $(PC - SC)$ does not give the allocation of the clients in an explicit way. One has to determine the allocation of a client C_i to the closest open facility by looking for a facility F_{j_0} such that $d_{ij_0} = \min_{j: \tilde{y}_j = 1} d_{ij}$.

To compare the two formulations, we analyze their LP-relaxations. Let (LPC) and $(LPC - SC)$ be the two linear programs obtained by relaxing the integrality constraints in problems (PC) and $(PC - SC)$, and let $v(LPC)$ and $v(LPC - SC)$ be their respective optimal objective values.

THEOREM 1. $v(LPC - SC) \geq v(LPC)$.

PROOF. Let $(\tilde{y}, \tilde{z})$ be an optimal solution to $(LPC - SC)$. Such a solution always exists because problem $(LPC - SC)$ is feasible and bounded from below by D^0. We build a feasible solution $(\hat{x}, \hat{y}, \hat{z})$ to (LPC), whose value $\hat{z}$ is not larger than $v(LPC - SC)$.

- $\hat{y} = \tilde{y}$
- For any client C_i, $i = 1, \ldots, N$, we consider the following partition of $\{1, \ldots, M\}$. Let $J_i^0, \ldots, J_i^K$ be the set of facilities at distance $D^0, \ldots, D^K$, respectively, from client C_i. That is, $J_i^k = \{j \in \{1, \ldots, M\}: d_{ij} = D^k\}$, for $k = 0, \ldots, K$. In order to define the values of the assignment variables $\hat{x}_{ij}$, $j = 1, \ldots, M$, we consider the two following cases:

1. Client C_i satisfies $\sum_{j \in J_i^0} \tilde{y}_j \geq 1$. In this case, the linear system

$$\begin{aligned} &\sum_{j \in J_i^0} x_{ij} = 1 \\ &0 \leq x_{ij} \leq \tilde{y}_j \quad j \in J_i^0 \end{aligned} \tag{14}$$

is consistent. Let $\bar{x}$ be any feasible solution to (14). We now define the assignment variables as:

$$\hat{x}_{ij} = \begin{cases} \bar{x}_{ij} & \text{for } j \in J_i^0 \\ 0 & \text{otherwise.} \end{cases}$$

2. Client C_i satisfies $\sum_{j \in J_i^0} \tilde{y}_j < 1$. Define $\alpha_i = \max\{\alpha \in \mathbb{N}: \sum_{j: d_{ij} \leq D^\alpha} \tilde{y}_j < 1\}$. This α_i always exists and $\alpha_i \leq (K - 1)$ follows from (8). By definition of α_i, $\sum_{j: d_{ij} \leq D^{\alpha_i + 1}} \tilde{y}_j \geq 1$ and then the linear system

$$\begin{aligned} &\sum_{j \in J_i^{\alpha_i + 1}} x_{ij} = 1 - \sum_{j: d_{ij} \leq D^{\alpha_i}} \tilde{y}_j \\ &0 \leq x_{ij} \leq \tilde{y}_j \quad j \in J_i^{\alpha_i + 1} \end{aligned} \tag{15}$$

is consistent. Let $\bar{x}$ be any feasible solution to (15). We now define the assignment variables as:

$$\hat{x}_{ij} = \begin{cases} \tilde{y}_j & \text{for } j \in J_i^k, \ 0 \leq k \leq \alpha_i \\ \bar{x}_{ij} & \text{for } j \in J_i^k, \ k = \alpha_i + 1 \\ 0 & \text{for } j \in J_i^k, \ \alpha_i + 1 < k \leq K. \end{cases}$$

It is straightforward to see that $(\hat{x}, \hat{y})$ satisfies (2)–(4).

• $\hat{z} = \max_{i=1,N} \sum_{j=1}^{M} d_{ij}\hat{x}_{ij}$. It is easy to check that $\hat{z}$ satisfies constraints (5).

Let us now compare the value $\hat{z}$ of the solution $(\hat{x}, \hat{y}, \hat{z})$ of (LPC), to the optimal solution value of $(LPC-SC)$, $v(LPC-SC) = D^0 + \sum_{k=1}^{K}(D^k - D^{k-1})\tilde{z}^k$. For this, we compare $v(LPC-SC)$ to $\sum_{j=1}^{M} d_{ij}\hat{x}_{ij}$ for each i and again consider two cases:

1. If i satisfies $\sum_{j\in J_i^0} \tilde{y}_j \geq 1$, then

$$v(LPC-SC) \geq D^0 = D^0 \sum_{j\in J_i^0} \hat{x}_{ij} = \sum_{j\in J_i^0} d_{ij}\hat{x}_{ij} = \sum_{j=1}^{M} d_{ij}\hat{x}_{ij}.$$

2. If i satisfies $\sum_{j\in J_i^0} \tilde{y}_j < 1$, then

$$\begin{aligned}
&v(LPC-SC)\\
&\geq D^0 + \sum_{k=1}^{\alpha_i+1}(D^k - D^{k-1})\tilde{z}^k\\
&\geq D^0 + \sum_{k=1}^{\alpha_i+1}(D^k - D^{k-1})\left(1 - \sum_{j:\, d_{ij}<D^k} \tilde{y}_j\right)\\
&\qquad \text{(by constraints (10))}\\
&= D^0 \sum_{j:\, d_{ij}=D^0} \tilde{y}_j + \cdots + D^{\alpha_i} \sum_{j:\, d_{ij}=D^{\alpha_i}} \tilde{y}_j\\
&\quad + D^{\alpha_i+1}\left(1 - \sum_{j:\, d_{ij}<D^{\alpha_i+1}} \tilde{y}_j\right)\\
&= D^0 \sum_{j\in J_i^0} \tilde{y}_j + \cdots + D^{\alpha_i} \sum_{j\in J_i^{\alpha_i}} \tilde{y}_j + D^{\alpha_i+1}\left(1 - \sum_{j:\, d_{ij}\leq D^{\alpha_i}} \tilde{y}_j\right)\\
&= D^0 \sum_{j\in J_i^0} \hat{x}_{ij} + \cdots + D^{\alpha_i} \sum_{j\in J_i^{\alpha_i}} \hat{x}_{ij}\\
&\quad + D^{\alpha_i+1} \sum_{j\in J_i^{\alpha_i+1}} \hat{x}_{ij} \text{ [as } \hat{x}_{ij},\ j \in J_i^{\alpha_i+1} \text{ satisfy (15)]}\\
&= \sum_{j=1}^{M} d_{ij}\hat{x}_{ij}.
\end{aligned}$$

Hence, $v(LPC-SC) \geq \max_{i=1,N} \sum_{j=1}^{M} d_{ij}\hat{x}_{ij} = \hat{z}$. □

The following example shows that the domination of $(LPC-SC)$ over (LPC) may be strict.

Example 1. Consider a symmetric instance with $N = M = 3$, $p = 2$, $d_{ii} = 0$ $(i = 1, \ldots, 3)$, $d_{12} = d_{13} = 2$, and $d_{23} = 1$. Take the obvious bounds $LB = 0$ and $UB = 2$. Here, $D^0 = 0$, $D^1 = 1$, and $D^2 = 2$. An optimal solution to (LPC) is $\hat{z} = \frac{2}{5}$, $\hat{y}_1 = \frac{4}{5}$, $\hat{y}_2 = \hat{y}_3 = \frac{3}{5}$, $\hat{x}_{11} = \frac{4}{5}$, $\hat{x}_{13} = \frac{1}{5}$, $\hat{x}_{22} = \hat{x}_{33} = \frac{3}{5}$, and $\hat{x}_{23} = \hat{x}_{32} = \frac{2}{5}$. Hence, $v(LPC) = \frac{2}{5}$. An optimal solution to $(LPC-SC)$ is $\tilde{z}^1 = \frac{1}{2}$, $\tilde{z}^2 = 0$, $\tilde{y}_1 = 1$, and $\tilde{y}_2 = \tilde{y}_3 = \frac{1}{2}$. Hence $v(LPC-SC) = \frac{1}{2}$. Note that for this example, the optimal integer value is equal to 1, and the lower bound LB^* we describe below in §3 is also equal to 1. □

Moreover, for the majority of the instances for which we computed the optimal values of problems (LPC) and $(LPC-SC)$, we observed a strict domination of $(LPC-SC)$ (see Tables 1 and 2 for instance).

The two MILPs (PC) and $(PC-SC)$ are polynomial in size. Problem (PC) has $(N+1)\times M$ binary variables, one continuous variable, and $N\times(M+2)+1$ constraints. Problem $(PC-SC)$ has $M+K$ binary variables and $K\times N+2$ constraints. Recall that $(K+1)$ is the number of different values in the distance matrix and is therefore bounded by $N\times M$. In the worst case where K is equal to $N\times M-1$, the two problems have almost the same number of variables but problem $(PC-SC)$ has about N times more constraints.

Observation 1. In formulation (PC), if a lower bound LB and an upper bound UB are known on z, then we can reduce the problem size. Indeed,

$$d_{ij} > UB \text{ implies } x_{ij} = 0 \quad i = 1, \ldots, N, j = 1, \ldots, M. \quad \square$$

The knowledge of bounds on the optimal value is even more useful when considering $(PC-SC)$.

Observation 2. In formulation $(PC-SC)$, if a lower bound LB and an upper bound UB are known then

$$\text{for all } k\text{: } D^k \leq LB \text{ implies } z^k = 1 \text{ and}$$
$$\text{for all } k\text{: } D^k > UB \text{ implies } z^k = 0.$$

This observation allows one to reduce the size of problem $(PC-SC)$ since fixing a z^k variable either to 0 or 1 amounts to reducing the value of K and thus the number of constraints (10). □

Table 1 **Solving (PC) and $(PC-SC)$ with $LB = LB_0$ and $UB = UB_0$**

						(PC) Used				$(PC-SC)$ Used			
Instance	$N=M$	p	Opt	LB_0	UB_0	LP	No. of Nodes	Gap %	CPU	LP	No. of Nodes	Gap %	CPU
Pmed1	100	5	127	0	186	91	28	—	3,600	108	38	—	3,600
Pmed2	100	10	98	0	178	64	51	—	3,600	85	82	20	3,600
Pmed3	100	10	93	0	205	63	107	—	3,600	82	50	11	3,600
Pmed4	100	20	74	0	204	42	160	—	3,600	61	96	22	3,600
Pmed5	100	33	48	0	169	20	2,598	10	3,600	35	214	0	2,500

Note. — = no integer feasible solution found after 3,600 seconds of CPU time.

Table 2 **Solving (PC) and $(PC-SC)$ with the Tighter Bounds $LB = LB_1$ and $UB = UB_1$**

						(PC) Used				$(PC-SC)$ Used			
Instance	$N=M$	p	Opt	LB_0	UB_0	LP	No. of Nodes	Gap %	CPU	LP	No. of Nodes	Gap %	CPU
Pmed1	100	5	127	59	138	98	1,733	22	3,600	108	1,065	0	3,200
Pmed2	100	10	98	56	117	77	3,184	0	3,500	86	66	0	340
Pmed3	100	10	93	55	146	74	1,114	32	3,600	84	105	0	1,100
Pmed4	100	20	74	41	111	55	8,081	0	2,100	65	360	0	1,300
Pmed5	100	33	48	23	92	31	1,748	0	450	38	144	0	260

As suggested by Observations 1 and 2, the size of the two formulations can be reduced if a lower bound and an upper bound are known. In the remainder of this section, we present a simple lower bound LB_0 and a simple upper bound UB_0. We also give slightly more sophisticated bounds LB_1 and UB_1. Finally, we give experimental results showing the impact of tighter bounds on the behavior of each of the formulations (PC) and $(PC-SC)$.

The bound LB_0 comes from the observation that the optimal radius is a decreasing function of p. Let LB_0 be the solution value of the M-center problem. In this problem, all of the facilities are opened, and each client is allocated to the closest facility, so the radius is $LB_0 = \max_{i=1,\dots,N}(\min_{j=1,\dots,M} d_{ij})$.

Now, let $\gamma_j = \max_{i=1,\dots,N}(\min_{h\neq j} d_{ih})$ for any facility F_j. This expression can be viewed as a lower bound on the values of all solutions with facility F_j closed. Furthermore, we know that in any feasible solution, at least $M-p$ facilities are closed, so the $(M-p)$th smallest value in γ is a lower bound for the p-center problem. Let $\gamma_{j_1} \leq \gamma_{j_2} \leq \dots \leq \gamma_{j_M}$ be the sorted values of γ, then $LB_1 = \gamma_{j_{M-p}}$.

The first upper bound UB_0 is the solution value of the 1-center problem, i.e.,

$$UB_0 = \min_{j=1,\dots,M}\Big(\max_{i=1,\dots,N} d_{ij}\Big).$$

The second upper bound UB_1 is obtained by the greedy heuristic described by Mladenovic et al. (2000). Once the 1-center problem is solved, a second facility that minimizes the resulting radius is opened. The same procedure can be repeated until p facilities are opened.

Tables 1 and 2 show experimental results for five instances from OR-Lib (see description in §5) of size $N = M = 100$. The MIPs are solved by using CPLEX 7.0. The first three columns of each table give the characteristics of the instance, and the fourth column gives its optimal value. The next two columns give the values of the lower bound and upper bound used, LB_0, UB_0 in Table 1 and LB_1, UB_1 in Table 2. Columns 7 through 10 (resp. 11 through 14) indicate the results obtained by using the same MIP solver on formulation (PC) (resp. $(PC-SC)$) together with a preprocessing phase based on Observation 1 (resp. Observation 2). Column *LP* contains the rounded up value of the linear relaxation of the formulation; Column *No. of Nodes* contains the number of branch-and-bound nodes. Column *Gap* contains the relative gap at the end of the branch and bound, i.e., either 0% if the problem is solved to optimality within 1 hour of CPU time or the gap between the best lower bound and best integer solution value found after one hour. Column *CPU* indicates the total CPU time in seconds devoted to that instance. The two tables show that, for given values of the lower bound LB and the upper bound UB, formulation $(PC-SC)$ performs better than does formulation (PC). They also indicate how each formulation takes advantage of the knowledge of good lower and upper bounds. Formulation $(PC-SC)$, used together with the tighter bounds LB_1 and UB_1, is the only one that can solve the five instances within one hour of CPU time. In the next section, we will focus on the computation of a tight lower bound LB^* by a polynomial-time algorithm. This algorithm also computes a better upper bound UB^*. We will show that when these two bounds are used, formulation $(PC-SC)$ can solve all 40 OR-Lib instances.

3. A Polynomial Algorithm for Computing a Tighter Lower Bound LB^*

Let $(PC-SC-R)$ be the MILP obtained from $(PC-SC)$ by relaxing the integrality constraints on the y variables only, that is, formulation (7)–(11) with $0 \leq y_j \leq 1$ for all $j = 1,\dots,M$. Let $LB^* = v(PC-SC-R)$ be the optimal value of $(PC-SC-R)$.

Proposition 1. *There exists h^* such that $LB^* = D^{h^*}$ and LB^* can be computed in polynomial time.*

Proof. Let (LSC_δ) be the LP-relaxation of problem (SC_δ). For problem $(PC-SC-R)$, $z^k = 0$ in an optimal solution if and only if the optimal solution value to problem $(LSC_{D^{k-1}})$ is less than or equal to p. Furthermore, the relation $z^{k+1} \leq z^k$ still holds for the optimal solution of problem $(PC-SC-R)$. Hence, in an optimal solution $(\tilde{y}, \tilde{z})$ to $(PC-SC-R)$ and even if $\tilde{y}$ is fractional, $\tilde{z}$ has the form $\tilde{z}^1 = \dots = \tilde{z}^{h^*} = 1$, $\tilde{z}^{h^*+1} = \dots = \tilde{z}^K = 0$ where h^* is the smallest $h \in \{0,\dots,K-1\}$ such that $v(LSC_{D^h}) \leq p$. This is equivalent to $LB^* = D^{h^*}$ and the computation of LB^* can be

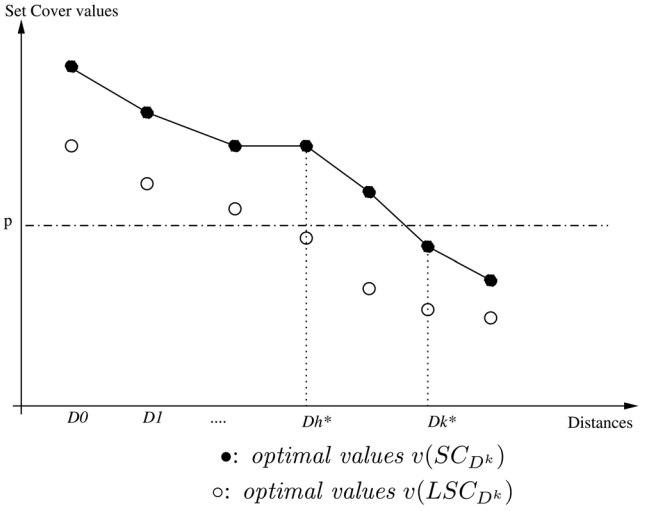

Figure 1 **Illustration of h^* and k^***

done by solving at most $K \leq N \times M$ linear set-covering problems (LSC_δ). □

Recall that k^* is the smallest k in $\{0, \ldots, K-1\}$ such that $v(SC_{D^k}) \leq p$, or, equivalently $v(PC-SC) = D^{k^*}$ is the optimal radius. Obviously, $h^* \leq k^*$. Whenever the two values are equal for a given instance, it means that there is no integrality gap between problems $(PC-SC-R)$ and $(PC-SC)$. Figure 1 illustrates the definitions and relation between h^* and k^*.

Using an LP solver to compute $v(LSC_\delta)$, one can easily compute LB^* by searching h^* in the interval $[0, K]$, or in a smaller interval if a better upper bound or a lower bound on the optimal radius is known. Binary search can be performed to reduce the maximal number of (LSC_δ) to be solved to $\lceil \log_2(K+1) \rceil$. Our algorithm *Bsearch* takes as input a p-center instance and the value of the solution computed by the greedy algorithm, UB_1. Note that the algorithm works with any initial upper bound. It computes LB^* and tries to find a better solution value UB^*. This bound UB^* is initialized to the value of the greedy solution and then updated every time a better solution to the p-center problem is found. At each iteration of the binary search, for a given h and the corresponding distance D^h, the main objective is to get LB^* by comparing $v(LSC_{D^h})$ to p, and the second objective is to get a better solution value UB^* for the p-center problem. At any time in the algorithm, $h^* \in \{head+1, \ldots, tail\}$. The algorithms starts by giving initial values to *head* and *tail* such that $D^{head+1} \leq LB^* \leq D^{tail}$.

Algorithm *Bsearch*

Input: N, M, P, UB_1, distances $D^0, \ldots, D^K = UB_1$, and distances d_{ij}, $i = 1, \ldots, N$, $j = 1, \ldots, M$.

Output: LB^*, UB^*.

Init. $head = -1$; $tail = K$;
Step 0. **If** $(head \geq tail - 1)$, **then** $\{LB^* = D^{tail}$; **STOP**$\}$
Step 1. $h = \lfloor (head + tail)/2 \rfloor$;
Step 2. Find a greedy solution y^G to (SC_{D^h});
Step 3. **If** $(\sum_{j=1}^{M} y_j^G \leq p)$, **then** $\{tail = h$; $UB^* = D^h$; **goto** Step 0.$\}$
Step 4. Deduce a heuristic solution to the p-center from y^G; update UB^* if necessary;
Step 5. Find an optimal solution $\tilde{y}$ to (LSC_{D^h}); keep the reduced costs rc_j, $j = 1, \ldots, M$;
Step 6. **If** $v(LSC_{D^h}) = \sum_{j=1}^{M} \tilde{y}_j > p$, **then** $\{head = h$; **goto** Step 0.$\}$ **else** $\{tail = h\}$
Step 7. Perform a reduced cost fixing phase;
Step 8. Compute a heuristic solution $\tilde{y}^{01}$ to (SC_{D^h}) based on the fractional solution $\tilde{y}$;
Step 9. Deduce a heuristic solution to the p-center from $\tilde{y}^{01}$; update UB^* if necessary;
Step 10. **while** $(UB^* < D^{tail})$ **do** $tail = tail - 1$;
Step 11. **goto** Step 0.

In Step 2, we use a classical greedy heuristic for the set-covering problem, described by Balas and Ho (1980). Let n_i be the number of facilities that can cover client C_i within distance D^h. The clients are sorted by increasing n_i. In that order, each uncovered client C_i is covered by opening a new facility that minimizes the number of uncovered clients. Once all clients are covered, we close redundant facilities, i.e., facilities that can be closed without uncovering any client. Whenever a greedy solution with no more than p facilities is found, the current iteration is stopped at Step 3. Steps 4 and 9 are performed to deduce a solution to the p-center problem from a solution to the set-covering problem (SC_{D^h}). If the number of open facilities in the solution to the set-covering problem is at most p, then the solution for the set-covering defines also a solution for the p-center. Otherwise, we close an open facility that minimizes the increase of the radius, and then repeat this process until no more than p facilities remain open. This gives a solution to the p-center problem, with a radius larger than D^h but that may be smaller than UB^*. Steps 5 and 6 constitute the heart of each iteration since the LP-relaxed set-covering problem (LSC_{D^h}) is solved and its optimal value compared to p. Step 7 is performed if the optimal value of (LSC_{D^h}) is at most p, i.e. $v(LSC_{D^h}) = \sum_{j=1}^{M} \tilde{y}_j \leq p$. Then a reduced cost fixing phase can be applied, using the reduced costs computed at Step 5: Fix y_j to zero for all facilities F_j such that $v(LSC_{D^h}) + rc_j > p$, where rc_j is the reduced cost associated with variable y_j. Indeed, every feasible solution to (SC_{D^h}) with $y_j = 1$ would require more than $v(LSC_{D^h}) + rc_j$ open facilities. It follows from the structure of Problem (LSC_{D^h}) that $0 \leq rc_j \leq 1$, so this phase is useless if $v(LSC_{D^h}) \leq p - 1$. Note also that this variable fixing is valid for all the set-covering problems (SC_δ) with $\delta \leq D^h$ but in our

Table 3 **Results of Algorithm *Bsearch* for the 40 OR-Lib Instances**

Instance	$N = M$	P	Opt	LB^*	UB^*	CPU
Pmed1	100	5	127	121	127	0.2
Pmed2	100	10	98	98	98	0.2
Pmed3	100	10	93	93	93	0.1
Pmed4	100	20	74	74	74	0.1
Pmed5	100	33	48	48	48	0.1
Pmed6	200	5	84	83	84	0.8
Pmed7	200	10	64	64	64	0.5
Pmed8	200	20	55	55	55	0.4
Pmed9	200	40	37	37	37	0.1
Pmed10	200	67	20	20	20	0.3
Pmed11	300	5	59	59	61	1.1
Pmed12	300	10	51	51	51	1.3
Pmed13	300	30	36	36	36	0.8
Pmed14	300	60	26	26	26	0.9
Pmed15	300	100	18	18	18	1.0
Pmed16	400	5	47	47	47	1.6
Pmed17	400	10	39	39	39	2.1
Pmed18	400	40	28	28	28	1.4
Pmed19	400	80	18	18	19	0.4
Pmed20	400	133	13	13	13	1.8
Pmed21	500	5	40	40	40	5.2
Pmed22	500	10	38	38	39	6.9
Pmed23	500	50	22	22	23	2.1
Pmed24	500	100	15	15	15	4.5
Pmed25	500	167	11	11	11	2.7
Pmed26	600	5	38	37	39	8.8
Pmed27	600	10	32	32	32	8.2
Pmed28	600	60	18	18	18	2.1
Pmed29	600	120	13	13	13	5.1
Pmed30	600	200	9	9	9	5.4
Pmed31	700	5	30	30	30	8.1
Pmed32	700	10	29	28	30	13.2
Pmed33	700	70	15	15	16	4.3
Pmed34	700	140	11	11	11	6.5
Pmed35	800	5	30	30	30	13.7
Pmed36	800	10	27	27	29	21.2
Pmed37	800	80	15	15	15	2.0
Pmed38	900	5	29	29	29	18.5
Pmed39	900	10	23	23	24	21.2
Pmed40	900	90	13	13	13	7.8
Average						4.5

algorithm, we do not use this property, i.e. all the variables are unfixed when going back to Step 0. In Step 8, another simple greedy heuristic is used to compute a feasible solution for (SC_{D^h}), based on the fractional optimal solution $\tilde{y}$ of its LP-relaxation. This heuristic is identical to the greedy heuristic of Step 2, except for the choice of a new facility to open. Here, to cover the next client C_i, a facility with a largest $\tilde{y}_j$ among the facilities within distance D^h from C_i is selected.

For an informal proof of the correctness of algorithm *Bsearch*, one can observe that the following properties are satisfied every time Steps 1 through 11 are performed:

1. $head \leq tail - 1$,
2. $v(LSC_{D^{tail}}) \leq p$, and
3. $head = -1$ or $v(LSC_{D^{head}}) > p$.

Table 3 reports the results of the execution of algorithm *Bsearch* on the 40 OR-Lib instances (see description in §5). Columns 1 through 3 give the characteristics of the instance, and the optimal radius is in column 4. The following two columns give the values LB^* and UB^* computed by algorithm *Bsearch* and the last column reports the CPU time required by this algorithm. One can observe that a positive integrality gap (difference between Opt and LB^*) exists only for the four instances *Pmed1, Pmed6, Pmed26*, and *Pmed32*. For 29 of the 36 remaining instances, $LB^* = UB^*$ i.e., algorithm *Bsearch* is sufficient to solve those instances to optimality. In conclusion, the lower bound LB^* is tight and algorithm *Bsearch* is very efficient in computing it. To compare, we tried to compute LB^* by solving $(PC-SC-R)$, using a direct MIP approach rather than *Bsearch*, for the *Pmed1* instance. Although we applied a preprocessing phase using the upper bound computed by the greedy algorithm and Observation 2, we got a MILP having 137 binary variables, 100 continuous variables, and 13,901 constraints. Cplex needed 40 minutes and 178 *Branch & Bound* nodes (cutting planes were turned off) to compute LB^* i.e. algorithm *Bsearch* is about 6,000 times faster for that instance.

4. Worst-Case Performance of LB^*

Hochbaum and Shmoys (1986) describe a 3-approximation algorithm for the p-center problem satisfying the triangle inequalities and show that no better approximation algorithm can exist unless $P = NP$. In the case of what we call in this paper the symmetric p-center, this result specializes to a 2-approximation algorithm that had already been shown to be optimal by Hsu and Nemhauser (1979). Using mainly the same ideas, we derive a worst-case performance for our lower bound LB^*. However, for clarity reasons, we adopt a different and more specific presentation than do Hochbaum and Shmoys (1986). This also makes our presentation self-contained.

For a given distance δ, let the "neighborhood graph" $\Delta_\delta = (V, E)$ be the graph where V is the set of clients and an edge $\{C_i, C_{i'}\} \in E$ if and only if there exists a facility F_j that can cover both the clients within distance δ. The *maximum stable set* problem (SS_δ) on graph Δ_δ can be formulated by the following IP:

Problem (SS_δ)

$$\max \sum_{i=1}^{N} x_i$$

$$\text{subject to} \quad \sum_{i:\, d_{ij} \leq \delta} x_i \leq 1 \quad j = 1, \ldots, M$$

$$x_i \in \{0, 1\} \quad i = 1, \ldots, N$$

where, in any optimal solution, $x_i = 1$ if and only if client C_i is in the optimal stable set. It is straightforward to observe that the linear relaxation of (SS_δ)

is the dual of (LSC_δ), the linear relaxation of (SC_δ). This implies that the cardinality of any stable set in graph Δ_δ is a lower bound of $v(SC_\delta)$. Exploiting this duality relationship, we prove the following proposition:

PROPOSITION 2. *For a given distance* D^k,
(i) *if* $v(SS_{D^k}) > p$ *then* $D^{k+1} \le LB^*$
(ii) *if* $v(SS_{D^k}) \le p$ *and all distances between clients and facilities satisfy the triangle inequalities then* $D^{k^*} \le 3D^k$ (*this was shown by Hochbaum and Shmoys* (1986)).

PROOF.
(i) if $v(SS_{D^k}) > p$, then $v(LSC_{D^k}) > p$ and, by definition of LB^*, $D^{k+1} \le LB^*$.
(ii) if $v(SS_{D^k}) \le p$, we construct a feasible solution to the p-center problem, with $v(SS_{D^k})$ open facilities and with a radius at most $3D^k$.

Let $S \subset V$ be a maximum stable set for graph Δ_{D^k}. Then $|S| = v(SS_{D^k})$. For $C_i \in S$, let $\phi(i)$ be a facility within distance D^k from C_i. By definition of graph Δ_{D^k}, since S is a stable set for Δ_{D^k} then $\phi(i') \ne \phi(i)$ for all $C_i \in S$, $C_{i'} \in S$, and $i' \ne i$. Hence, one needs at least $|S|$ centers to cover all the clients of S, within distance D^k. Let $\Phi(S) = \{\phi(i)\colon C_i \in S\}$. One has $|\Phi(S)| = |S|$. We consider the following two cases:

1. $\Phi(S)$ is a feasible solution to problem (SC_{D^k}), i.e., each client C_i can be covered, by a facility in $\Phi(S)$ within distance D^k. In this case, $\Phi(S)$ is a feasible p-center solution with a radius at most D^k. Hence $D^{k^*} \le D^k$.

2. $\Phi(S)$ is not a feasible solution to problem SC_{D^k}, i.e. at least one client C_i cannot be covered by any facility in $\Phi(S)$ within distance D^k. But, since S is a maximum stable set, C_i is adjacent, in graph Δ_{D^k}, to a client $C_{i'}$ in S. Hence, there exists a facility $F_j \notin \Phi(S)$ that covers both C_i and $C_{i'}$ within distance D^k.

$$C_i \xrightarrow{\le D^k} F_j \xrightarrow{\le D^k} C_{i'} \xrightarrow{\le D^k} \phi(i')$$

If all distances between clients and facilities satisfy the triangle inequalities, client C_i can be covered by facility $\phi(i')$ within distance $3D^k$. Therefore $\Phi(S)$ constitutes a feasible p-center solution, with a radius of at most $3D^k$. □

COROLLARY 1. *If all distances between clients and facilities satisfy the triangle inequalities, then* $LB^* \ge \frac{1}{3}D^{k^*}$.

PROOF. For the same reasons as the set-covering problems and their LP-relaxations, the optimal value of the stable set problem (SS_{D^δ}) is a non-increasing function of the distance δ. Then either $v(SS_{D^0}) \le p$ or there exists $\tilde{k}$ such that $v(SS_{D^{\tilde{k}-1}}) > p$ and $v(SS_{D^{\tilde{k}}}) \le p$. It follows from Proposition 2 that either $D^{k^*} \le 3D^0$, or $D^{\tilde{k}} \le LB^*$ and $D^{k^*} \le 3D^{\tilde{k}}$. It then suffices to recall that $D^{k^0} \le LB^*$ to complete the proof. □

The specialization of Proposition 2 and Corollary 1 to the symmetric p-center is straightforward.

COROLLARY 2. *For the symmetric p-center,* $LB^* \ge \frac{1}{2}D^{k^*}$.

5. The Exact Solution Method

In order to compute the optimal radius, once the bounds LB^* and UB^* have been computed by algorithm *Bsearch*, one can apply Observation 2 to problem $(PC - SC)$ and then solve it using an MIP solver, in the same way as Tables 1 and 2 were obtained. But one can again take advantage from the decomposition property of the p-center problem into set-covering problems. We call our exact solution algorithm *BsearchEx*. Its objective is to find k^* such that D^{k^*} is the optimal radius. It proceeds by binary search on the distances $LB^* = D^{h^*} \le D^{h^*+1} \le \cdots \le D^{K'} = UB^*$. Algorithm *BsearchEx* is similar to our algorithm *Bsearch* of §3, except in the following steps:

- The Init. step is replaced by

  Init.ex $head = h^* - 1$; $tail = K'$;

- Step 6 is removed
- Step 8 and Step 9 are replaced by:

  Step 8.ex. Compute an optimal solution y^{01} to (SC_{D^h});

  Step 9.ex. If $v(SC_{D^h}) = \sum_{j=1}^{M} y_j^{01} > p$,
  then $\{head = h\}$,
  else $\{tail = h$; update UB^* if necessary$\}$.

At any time in the new algorithm, $k^* \in \{head + 1, \ldots, tail\}$. At Step 8.ex, an optimal solution rather than a heuristic one is computed for problem (SC_{D^h}). In fact, we do not need to solve this problem to optimality but only to get an answer to the question "Does it have a solution of value less or equal to p?" For this, we use the MIP solver of CPLEX 7.0 and change some of its default parameters:

- Change *mip tolerances uppercutoff* to $(p + 0.001)$, i.e., integer solutions with a value larger than p are discarded;
- Change *mip limits solutions* to 1, i.e., the resolution stops as soon as an integer solution (with a value at most p) is found; and
- Change *time limit* to 3,600, i.e., the resolution stops if no integer solution is found after one hour of CPU time.

Hence, the resolution process stops either because a solution of value of at most p is found, or because the solver proves that p is a strict lower bound for the optimal value of the current problem (SC_{D^h}). Moreover, when the MIP solver is still unable to answer after one hour, we consider p to be a strict lower bound. If this happens we are no longer sure that our algorithm *BsearchEx* computes the optimal solution value for the p-center problem, but an upper bound on it. Note that we tried to improve the solver performances by changing some other settings, mainly in the presolve and cut-generation phases. But none of our changes led to significant improvement.

Here we describe all the test instances used in this paper:

OR-Lib Instances. There are 40 instances in OR-Lib (Beasley 1990) that were initially used for the *p*-median problem. These instances have also been used for the *p*-center problem to compare heuristic solution methods in Mladenovic et al. (2000) and to test an exact solution method in Daskin (2000). All of them are symmetric *p*-center instances. Here, $N = M$ and varies in the range from 100 to 900, and p varies from 5 to 90. A first set of distances is given, defining a connected network of the clients, and one has to compute a shortest path between each pair of clients to get the whole distance matrix.

TSPLIB Instances. These instances come from TSPLIB (Reinelt 1991) and are usually devoted to the travelling salesman problem. If N is the number of cities (clients), we set $M = N$ and consider different values of p, typically $10, 20, 30$, etc. Given the coordinates of the clients in the plane, the Euclidean distance is computed and rounded to the nearest integer, for every pair of clients. We derive 15 (resp. 10, 15) instances from the u1060 (resp. rl1323, u1817) TSP instances. These instances are symmetric *p*-center instances, and the idea of using them comes from Mladenovic et al. (2000).

Random Euclidean Instances. We also generated a few random instances. Here, $N = M = 100$; coordinates of the points in the plane are randomly generated in $[0, 100]$; and Euclidean distances are computed between the points. Based on these common characteristics, three instances are considered with different values for p: 5, 10, and 15. These instances are named *Euc100*.

Random Instances. Again, $N = M = 100$ but now, distances are randomly and uniformly generated in $[0, 100]$ and satisfy $d_{ii} = 0$ and $d_{ij} = d_{ji}$. We also build three instances with $p = 5$, 10, and 15. These instances are named *Rand100* and are the only instances among all those we test for which the triangle inequalities are not satisfied.

Table 4 gives results of our algorithm *BsearchEx* for the 11 OR-Lib instances for which $LB^* < UB^*$ (see Table 3) and for the TSPLIB instances. The first three columns characterize the instance. Column 4 gives either the optimal value or the best found solution value. Columns 5 through 8 report the results of Algorithm *Bsearch*. Column $\#LSC_\delta$ gives the number of problems (LSC_δ) that have been solved, and *cpu1* is the CPU time devoted to Algorithm *Bsearch*. Column 9 gives $\#SC_\delta$, the number of problems (SC_δ) that have been solved by Algorithm *BsearchEx*. The last column, *cpu2*, is the total CPU time devoted to solving the instance, i.e., to read the instance data, perform the greedy algorithm, sort the distances, and perform Algorithms *Bsearch* and *BsearchEx*.

Table 4 Results of Our Exact Solution Method for Instances from OR-Lib and TSPLIB

Instance	$N = M$	p	Opt	LB^*	UB^*	$\#LSC_\delta$	*cpu1*	$\#SC_\delta$	*cpu2*
Pmed1	100	5	127	121	127	7	0.2	3	0.7
Pmed6	200	5	84	83	84	6	0.8	1	0.3
Pmed11	300	5	59	59	61	4	1.1	1	1.0
Pmed19	400	80	18	18	19	4	0.4	1	0.4
Pmed22	500	10	38	38	39	5	6.9	1	4.3
Pmed23	500	50	22	22	23	4	2.1	1	1.2
Pmed26	600	5	38	37	39	6	8.8	1	6.1
Pmed32	700	10	29	28	30	5	13.2	2	45.2
Pmed33	700	70	15	15	16	3	4.3	1	3.1
Pmed36	800	10	27	27	29	5	21.2	1	34.5
Pmed39	900	10	23	23	24	5	21.2	1	27.3
u1060	1,060	10	2,273	2,273	2,273	6	27	0	53
u1060	1,060	20	1,581	1,556	1,768	9	63	6	2,778
u1060	1,060	30	1,208	1,205	1,275	8	50	4	298
u1060	1,060	40	1,021	1,013	1,079	9	35	4	366
u1060	1,060	50	905	895	963	6	21	4	383
u1060	1,060	60	781	765	807	8	21	4	233
u1060	1,060	70	711	707	761	6	17	3	135
u1060	1,060	80	652	652	711	6	18	2	60
u1060	1,060	90	608	604	636	6	19	3	38
u1060	1,060	100	570	570	570	5	18	0	29
u1060	1,060	110	539	539	539	5	18	0	30
u1060	1,060	120	510	510	538	7	29	1	44
u1060	1,060	130	500	495	510	6	28	1	44
u1060	1,060	140	452	452	500	5	28	3	46
u1060	1,060	150	447	430	447	6	34	3	50
rl1323	1,323	10	3,077	3,062	3,329	11	106	8	1,380
rl1323	1,323	20	2,016	2,008	2,152	11	115	5	480
rl1323	1,323	30	1,632	1,611	1,797	11	99	7	900
rl1323	1,323	40	1,352	1,334	1,521	10	76	7	3,000
rl1323	1,323	50	1,187	1,165	1,300	9	61	7	8,580
rl1323	1,323	60	1,063	1,047	1,194	11	55	8	9,120
rl1323	1,323	70	972	959	1,040	10	42	7	1,740
rl1323	1,323	80	895	889	948	10	37	5	420
rl1323	1,323	90	832	830	857	10	30	4	120
rl1323	1,323	100	787	777	803	9	26	5	120
u1817	1,817	10	458	455	467	8	611	4	2,700
u1817	1,817	20	310?	306	342	9	660	5	4,920
u1817	1,817	30	250?	240	287	8	355	6	16,500
u1817	1,817	40	210?	205	234	6	247	4	6,420
u1817	1,817	50	187?	180	205	8	242	4	9,840
u1817	1,817	60	163	163	183	6	177	4	1,260
u1817	1,817	70	148	148	152	6	166	1	420
u1817	1,817	80	137	137	148	5	150	3	1,140
u1817	1,817	90	130?	127	148	6	161	3	7,202
u1817	1,817	100	127	127	130	6	159	1	300
u1817	1,817	110	109	108	127	5	119	4	420
u1817	1,817	120	108	108	108	6	131	0	120
u1817	1,817	130	108?	105	109	4	121	1	3,720
u1817	1,817	140	105?	102	108	5	121	2	4,020
u1817	1,817	150	94?	92	108	5	144	3	5,640

Note. "?" = opt is the value of the best found solution for that instance.

For the 11 OR-Lib instances Pmedxx, *BsearchEx* needed to solve 14 set-covering problems and took 123 seconds of CPU time. To compare, we also used the first method suggested in this section i.e., solve $(PC - SC)$ directly, and observed that it was about 30 times slower than *BsearchEx*.

Daskin (2000) gives experimental results of his exact solution method for the 40 OR-Lib instances of Table 3. Even if it is not straightforward to compare CPU times on different machines, we can give as indication the fact that our average CPU total time for all these instances is 9.1 seconds with a maxi-

Table 5 Results of Our Exact Solution Method for Random Instances

Instance	$N=M$	p	Opt	LB^*	UB^*	$\#LSC_\delta$	$cpu1$	$\#SC_\delta$	$cpu2$
Euc100	100	5	29	29	29	3	0.1	0	0.1
Euc100	100	10	20	20	20	3	0.1	0	0.1
Euc100	100	15	15	15	15	3	0.1	0	0.1
Rand100	100	5	28	20	39	6	0.4	4	37.2
Rand100	100	10	13	10	17	5	0.2	3	42.5
Rand100	100	15	7	6	10	4	0.2	2	4.9

mum of 59.1 seconds, while Daskin's results show 91.5 seconds with a maximum of 1402.5 seconds. Daskin's results are obtained using a PC with 128MB of RAM and a Pentium III processor at 650Mhz.

Another comparison can be done with the work by Mladenovic et al. (2000), whose objective was to design and test heuristic methods for the p-center problem. We can now test the average deviation of their best solution, computed by a *variable neighborhood search* heuristic, the best among the other tested heuristics. This deviation is 2.37% for the 40 OR-Lib instances (average CPU time of 133.6 seconds for the *VNS* heuristic on a SUN Sparc 10) and 0.02% (average CPU time of 300 seconds) for the 15 TSPLIB instances with $N=M=1060$.

We also wanted to check how the quality of our lower bound LB^* varies if the distances do not satisfy the triangle inequalities. For this, we constructed the random instances described above. Table 5 reports the results obtained for these instances. It appears that the lower bound LB^* is much tighter for the Euclidean instances. In fact, the relative gap $(\text{Opt}-LB^*)/\text{Opt}$ does not exceed 5% for any of the symmetric p-center instances we tested, while it is equal to 28% for problem *Rand100* with $p=5$.

Finally, we did not test non-symmetric instances systematically. We only solved the 40 OR-Lib instances with $M=N/2$ to make the instances non-symmetric, and observed that LB^* is equal to the optimal radius for all those instances. The average CPU time needed to solve the 40 instances was less than for the case $M=N$. Consequently, we do not expect non-symmetric instances, for which triangle inequalities are satisfied, to be more difficult to solve.

6. Extension to the Fault-Tolerant p-Center Problem

Many variations and generalizations of the p-center problem exist. One of them is the *fault-tolerant* p-center introduced in Krumke (1995) and its variations in Khuller et al. (2000). Given an integer $\alpha \le p$, each client is to be assigned to α of the p open facilities, and the objective is to minimize the maximal distance between any client and any of the facilities to which it is assigned. Our formulation $(PC-SC)$ can be generalized to the *fault-tolerant* p-center by replacing constraints (10) by

$$\alpha z^k + \sum_{j:d_{ij}<D^k} y_j \ge \alpha \quad i=1,\dots,N;\ k=1,\dots,K. \tag{16}$$

The binary search algorithms can be extended, both for an exact solution method and for the computation of a generalization of the lower bound LB^*. For this, one has to replace the set-covering sub-problem (SC_δ) for a given distance δ by the following *set-α-covering problem* obtained by replacing constraints (13) by

$$\sum_{j:d_{ij}\le\delta} y_j \ge \alpha \quad i=1,\dots,N.$$

A short computational experiment, on the 40 problems from OR-Lib, is reported in Elloumi et al. (2002), showing that the bound LB^* remains of very high quality for $\alpha=3$. In particular, we were able to solve all the OR-Lib instances and all the u1060 instances with $\alpha=3$ and observe that LB^* is equal to the optimal value for all those instances.

7. Conclusion

In this paper, we introduced a new MIP formulation for the general p-center problem. We proved that this formulation is better than the previous one because its LP relaxation gives a better lower bound. Moreover, it guided us to the lower bound LB^*, which is the optimal value of the problem obtained by the continuous relaxation of a subset of the variables in the new formulation. Another advantage of this formulation is that it can be easily generalized to more complex p-center problems such as the *fault tolerant* p-center. Although we have not yet investigated this question precisely, we think that this approach can also be used for other problems with a min-max objective.

We give a more combinatorial interpretation of bound LB^*, together with an efficient algorithm to compute it, based on the resolution of a series of LP problems. A by-product of this algorithm is an upper bound UB^*. Then, we use a quite elementary exact solution method, based on the resolution of a series of minimum-cardinality set-covering problems. Our experimental results aimed at showing the quality of the lower bound LB^* and getting an idea on how it can improve the resolution of the p-center problem. One can probably get better computational results by using state-of-the art methods for the set-covering problem. Possibly, the usage of the lower bound LB^* in a bounding phase of a branch-and-bound algorithm can lead to better global performance.

Our computational results show that LB^* is a very tight lower bound for the symmetric instances. For

non-symmetric instances, LB^* is not tight but algorithm *Bsearch* is fast enough to be used as a preprocessing phase in a different exact or heuristic solution method. A polyhedral investigation of the new formulation is probably an interesting and useful topic for further research. Indeed, the addition of strong valid inequalities to the new formulation or to the related set-covering problems might yield a tighter lower bound.

While finishing the writing of this paper, we became aware of very recent work by Ilhan and Pinar (2001). An exact solution method for the *p*-center problem is proposed. Their solution method is composed of two phases: the first phase, called the *LP-Phase*, turns out to compute the same value as our bound LB^*. The main difference is that they solve a series of feasibility problems rather than our minimal-cardinality set-covering problems. Their feasibility problems are our problems (SC_δ) with no objective function but with an additional constraint on the number of open facilities. Our first implementation of algorithm *Bsearch* also considered feasibility problems, but we obtained substantially better computational times when we changed to minimum-cardinality set-covering problems. Having a look at their results for instances from OR-Lib, one can however notice that our bound LB^* is often slightly better than their *LP-Phase* bound. We cannot explain this difference, maybe due to numerical precision in the solution of the LP feasibility problems.

Acknowledgments

The authors thank an anonymous referee for extensive comments that substantially improved the readability of the paper. They also thank Ismael de Farias for a very helpful reading and comments. Sourour Elloumi's research was partially supported by the Research in Brussels 2002 programme of The Minister-President of the Brussels-Capital Region. Martine Labbé's research was partially supported by the Ministerio de Educación, Cultura y Desportes of Spain and by the European Social Fund (SAB2000-0069). Yves Pochet's research was carried out with financial support of the project TMR-DONET ERB FMX-CT98-0202 of the European Community and within the framework of the Belgian Program on Interuniversity Poles of Attraction initiated by the Belgian State, Prime Minister's Office, Science Policy Programming.

References

Balas, E., A. Ho. 1980. Set-covering algorithms using cutting planes, heuristics and subgradient optimization: A computational study. *Math. Programming Stud.* **12** 37–60.

Beasley, J. E. 1990. OR-Library: distributing test problems by electronic mail. *J. Oper. Res. Soc.* **41** 1069–1072.

Daskin, M. 1995. *Network and Discrete Location*. Wiley, New York.

Daskin, M. 2000. A new approach to solving the vertex *p*-center problem to optimality: Algorithm and computational results. *Comm. Oper. Res. Soc. Japan* **45** 428–436.

Elloumi, S., M. Labbé, Y. Pochet. 2002. Generalisations of the *p*-center problem: Formulations and solution methods. *XV Conf. Eur. Chap. Combin. Optim.*, Lugano, Switzerland, May 2002.

Hochbaum, D., D. Shmoys. 1985. A best possible heuristic for the *k*-center problem. *Math. Oper. Res.* **10** 180–184.

Hochbaum, D., D. Shmoys. 1986. A unified approach to approximation algorithms for bottleneck problems. *J. Assoc. Comput. Machinery* **33** 533–550.

Hsu, W. L., G. L. Nemhauser. 1979. Easy and hard bottleneck location problems. *Discrete Appl. Math.* **I** 209–216.

Ilhan, T., M. C. Pinar. 2001. An efficient exact algorithm for the vertex *p*-center problem. http://www.optimization-online.org/DB_HTML/2001/09/376.html.

Kariv, O., S. L. Hakimi. 1979. An algorithmic approach to network location problems. Part 1: The *p*-centers. *SIAM J. Appl. Math.* **37** 513–538.

Khuller, S., R. Pless, Y. Sussmann. 2000. Fault tolerant *k*-center problems. *Theoret. Comput. Sci.* **242** 237–245.

Krumke, S. O. 1995. On a generalization of the *p*-center problem. *Inform. Processing Lett.* **56** 67–71.

Marianov, V., C. ReVelle. 1995. Siting emergency services. Z. Drezner, ed. *Facility Location: A Survey of Applications and Methods*, Springer *Series in Operations Research.* Springer-Verlag, New York, 199–222.

Masuyama, S., T. Ibaraki, T. Hasegawa. 1981. The computational complexity of the *m*-center problems on the plane. *Trans. IECE Japan* **E-64** 57–64.

Megiddo, N., A. Tamir, E. Zemel, R. Chandrasekaran. 1981. An $O(n \log^2 n)$ algorithm for the *k*th longest path in a tree with applications to location problems. *SIAM J. Comput.* **10** 328–337.

Mladenovic, N., M. Labbé, P. Hansen. 2000. Tabu search and variable neighborhood search in solving the *p*-center problem. *ULB-ISRO preprint 2000/20*. http://smg.ulb.ac.be/.

Reinelt, G. 1991. TSPLIB-a traveling salesman problem library. *ORSA J. Comput.* **3** 376–384.

ReVelle, C., C. Toregas, R. Swain, L. Bergman. 1971. The location of emergency service facilities. *Oper. Res.* **19** 1363–1373.