NEW FORMULATION AND RESOLUTION METHOD FOR THE p-CENTER PROBLEM

Sourour ELLOUMI[1], Martine LABBÉ[2] and Yves POCHET[3]

December 2001

Abstract

The p-Center problem consists in locating 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 made of N rows and M columns. We also show that, when the distances satisfy triangle inequalities, LB^* is at least equal to half of the optimal value. Finally, we use LB^* as a starting point in an exact solution method and report extensive computational results on test problems from the literature. For Euclidean instances, our method outperforms the runtime of other recent exact methods by an order of magnitude. Moreover, it is the first one to solve large instances of size up to $N = M = 1817$.

Keywords : Min-Max objective, Facility Location, p-Center, Mathematical Programming

[1]CEDRIC-CNAM,292 Rue Saint-Martin, F-75141 Paris cedex 03, France.

[2]Service d'Optimisation, Institut de Statistique et de Recherche Opérationnelle, Université Libre de Bruxelles, Boulevard du Triomphe CP 210/01, B-1050 Bruxelles, Belgium.

[3]CORE and IAG, Université catholique de Louvain, Voie du Roman Pays, 34 B-1348 Louvain-La-Neuve, Belgium.

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.

1 Introduction

The p-Center problem consists in locating at most p facilities and assigning n clients, each to its closest open facility, in order to minimize the radius, i.e. the maximum distance between a client and its closest facility. Many applications of this problem arise in the public sector as for example locating fire stations or ambulance depots, see Daskin [4].

Let N be the number of clients called $C_1, C_2, ..., C_N$; M be the number of potential sites for facilities $F_1, F_2, ..., F_M$; and d_{ij} be the distance from C_i to F_j. The problem is to find a subset S of $\{1, ..., M\}$ such that:

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

This problem includes the case where a nonnegative wheight w_i is associated with each client C_i and where the radius of a solution S is $max_{i=1,...,N}(w_i min_{j \in S} d_{ij})$. To consider such weights in an unweighted instance, it suffices to modify the distances by setting $d'_{ij} = w_i d_{ij}$.

Of course, 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 locations exist. In Megiddo et al. [14], the p-Center problem is proved to be polynomial if the d_{ij}'s represent distances in a tree network but in general, the p-Center problem is NP-hard, see Kariv and Hakimi [9].

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

A strong relation exists between the p-Center problem and the *set-covering* problem in the general case, and with the *dominating set* 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 [6]. In the general case, this problem is a minimal cardinality set covering problem [17]. Daskin [5] considers a different auxilliary subproblem, consisting in maximizing the number of covered clients by no more than p facilities, within a given radius. He calls it the *maximal covering* problem. Ilhan and Pinar [8] consider yet a different subproblem which 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. Hochbaum and Shmoys [6] consider the symmetric p-Center problem and use a greedy heuristic to build a dominating set in the square of the initial graph. The global algorithm leads to a solution to the p-Center problem with worst case ratio 2. The authors also prove that this 2-approximation algorithm is optimal i.e., unless $P = NP$, it is impossible to find a δ-approximation polynomial time algorithm with $\delta < 2$.

In this paper we suggest a new formulation for the general p-Center problem that takes into account its relationship with the set covering problem. In the next section we first recall the classical formulation (PC). Then we present the 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 the LP relaxation of (PC). Furthermore we show, with a few experimental results, how formulation $(PC-SC)$ takes advantage from the knowledge of tight lower and upper bounds. In Section 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 in solving a series of linear set-covering problems. UB^* is simply the best solution to the p-Center problem obtained while computing LB^*. In Section 4 we show that our lower bound LB^* can be compared to the lower bound included in Hochbaum and Shmoys' algorithm [6], yet the later algorithm was devoted to a heuristic point of view. This comparison allows us to conclude that LB^* cannot be less than half of the optimal radius. In Section 5 we describe our exact solution algorithm and we report experimental results. All along this paper, our computational results concern instances coming either from OR-Lib [2] or from TSPLIB [16]. Further, we used a PC with 384 MB of RAM and a 400 MHz PentiumII processor, C++, and CPLEX 7.0 [3]. We conclude in Section 6 with some possible extensions of this approach to solve 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 [4]). For any feasible solution (x,y,z): $y_j=1$ iff facility F_j is open, $x_{ij}=1$ iff client C_i is assigned to facility F_j, and z is an upper bound on the radius of the feasible solution. In an optimal solution, $z = max_{i=1,...,N} \sum_{j=1}^{M} d_{ij}x_{ij}$ is the optimal radius.

Formulation (PC)

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

subject to

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

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

$$x_{ij} \leq y_j \quad i=1,...,N, j=1,...,M \tag{4}$$

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

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

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.

Observation 1
If a lower bound LB and an upper bound UB are known on z then we can reduce the problem size. Indeed,

$$\begin{array}{c} d_{ij} > UB \text{ implies } x_{ij} = 0 \quad i = 1, ..., N, j = 1, ..., M, \text{ and} \\ d_{ij} < LB \text{ allows to set } d_{ij} = LB \quad i = 1, ..., N, j = 1, ..., M. \end{array}$$

□

Our new formulation of the p-Center problem is based on its well-known relation with the set-covering problem. Let D_{min} (resp. D_{max}) be the smallest (resp. largest) value in the distance matrix, and let $D_{min} = D^0 < D^1 < D^2 < ... < D^K = D_{max}$ be the sorted different values in that matrix. We now present a new formulation $(PC - SC)$ using variables $y_j, j = 1, ..., M$, and $z^k, k = 1, ..., K$. In this formulation, for any feasible solution (y, z), $y_j = 1$ iff facility F_j is open and $z^k = 0$ if it is possible to locate p facilities and cover all the clients within the radius D^{k-1}. Note that $z^k = 0$ implies $z^{k+1} = ... = z^K = 0$ and the objective function is strictly lower than D^k.

Formulation $(PC - SC)$

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

subject to

$$\sum_{j=1}^{M} y_j \geq 1 \tag{8}$$

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

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

$$z^k \in \{0, 1\} \qquad k = 1, ..., K \tag{11}$$

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

Constraint (8) discards solutions with no open facility and 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, $z^k = 0$, iff 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 $(PC-SC)$ iff the optimal value of the set-covering problem $SC_{D^{k-1}}$ is less or equal to p.

Problem (SC_α)

$$\min \quad \sum_{j=1}^{M} y_j \tag{13}$$

subject to

$$\sum_{j:d_{ij}\leq\alpha} y_j \geq 1 \quad i = 1, ..., N \tag{14}$$

$$y_j \in \{0,1\} \quad j = 1, ..., M \tag{15}$$

We define $v(SC_\alpha)$ as the optimal objective function value of SC_α, where $v(SC_\alpha) = \infty$ when SC_α contains no feasible solution. It is easy to check that $v(SC_{D^0}) \geq v(SC_{D^1}) ... \geq v(SC_{D^K})$. Now, we make more precise the relation between the optimal solutions of problems SC_{D^k} and $(PC-SC)$. Let k^* be the smallest k in $0, ..., K$ 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 $\tilde{z}^1 = ... = \tilde{z}^{k^*} = 1$ and $\tilde{z}^{k^*+1} = ... = \tilde{z}^K = 0$. Therefore, solving $(PC-SC)$ can be done 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 by looking for a facility F_{j_0} such that $d_{ij_0} = min_{j:\tilde{y}_j=1} d_{ij}$.

Observation 2
The knowledge of bounds on the optimal value can also be useful when considering $(PC-SC)$. If a lower bound LB and an upper bound UB are known then

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

This also reduces the size of problem $(PC-SC)$ since fixing a z^k variable either to 0 or 1 amounts to reduce the value of K and thus the number of Constraints (10).

□

To compare the two formulations, we analyse their LP-relaxations.

Theorem 1 *Let (LPC) and $(LPC-SC)$ be the two linear programs obtained by relaxing the integrality constraints in problems (PC) and $(PC-SC)$. Then $v(LPC-SC) \geq v(LPC)$.*

Proof
Let $(\tilde{y}, \tilde{z})$ be an optimal solution to $(LPC - SC)$. We build a feasible solution $(\hat{x}, \hat{y}, \hat{z})$ to (LPC) whose value $\hat{z}$ is not larger than $v(LPC - SC)$.

- $\hat{y} = \tilde{y}$
- Here, we define the values of the allocation variables $\hat{x}_{ij}, j = 1, ..., M$, for a given client $C_i, i = 1, ..., N$. Consider the following partition of $\{1, ..., M\}$. Let $J_i^0, ..., J_i^K$ be the set of facilities at distance $D^0, ..., D^K$, respectively, from client C_i. That is, $J_i^k = \{j \in \{1, ..., M\} : d_{ij} = D^k\}$, for $k = 0, ..., K$. We also 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 (16)-(17) is consistent.

$$\sum_{j \in J_i^0} x_{ij} = 1, \tag{16}$$

$$0 \leq x_{ij} \leq \tilde{y}_j \qquad j \in J_i^0. \tag{17}$$

     We can now define the allocation variables as:
     - $\hat{x}_{ij}, j \in J_i^0$ is a feasible solution to (16)-(17) and
     - $\hat{x}_{ij} = 0$ for all $j \in J_i^k, k = 1, ..., K$.

  2. Client C_i satisfies $\sum_{j \in J_i^0} \tilde{y}_j < 1$. Define $\alpha_i = max\{\alpha \geq 0 : \sum_{j:d_{ij} \leq D^\alpha} \tilde{y}_j < 1\}$. Of course, α_i always exist in this case 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 (18)-(19) is consistent.

$$\sum_{j \in J_i^{\alpha_i+1}} x_{ij} = 1 - \sum_{j:d_{ij} \leq D^{\alpha_i}} \tilde{y}_j \tag{18}$$

$$0 \leq x_{ij} \leq \tilde{y}_j \qquad j \in J_i^{\alpha_i+1} \tag{19}$$

     We can now define the allocation variables as:
     - $\hat{x}_{ij} = \tilde{y}_j$ for all $j \in J_i^k, k = 0, ..., \alpha_i$,
     - $\hat{x}_{ij}, j \in J_i^{\alpha_i+1}$, is a feasible solution to (18)-(19), and
     - $\hat{x}_{ij} = 0$ for all $j \in J_i^k, \alpha_i + 1 < k \leq K$.

  It is straightforward to see that $\hat{x}$ and $\hat{y}$ satisfy (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 solution $(\hat{x}, \hat{y}, \hat{z})$ to (LPC), to $v(LPC - SC) = D^0 + \sum_{k=1}^{K} (D^k - D^{k-1})\tilde{z}^k$, the optimal solution value to $(LPC - SC)$. For this, we compare $v(LPC - SC)$ to $\sum_{j=1}^{M} d_{ij}\hat{x}_{ij}$ for each i.

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

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

  $= \sum_{j \in J_i^0} d_{ij} \hat{x}_{ij}$

  $= \sum_{j=1}^{M} d_{ij} \hat{x}_{ij}$.

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

  $v(LPC - SC) \geq D^0 + \sum_{k=1}^{\alpha_i+1} (D^k - D^{k-1}) \tilde{z}^k$

  $\geq D^0 + \sum_{k=1}^{\alpha_i+1} (D^k - D^{k-1})(1 - \sum_{j:d_{ij}<D^k} \tilde{y}_j)$ by Constraints (10)

  $= D^0 \sum_{j:d_{ij}=D^0} \tilde{y}_j + ... + D^{\alpha_i} \sum_{j:d_{ij}=D^{\alpha_i}} \tilde{y}_j + D^{\alpha_i+1}(1 - \sum_{j:d_{ij}<D^{\alpha_i+1}} \tilde{y}_j)$

  $= D^0 \sum_{j \in J_i^0} \tilde{y}_j + ... + D^{\alpha_i} \sum_{j \in J_i^{\alpha_i}} \tilde{y}_j + D^{\alpha_i+1}(1 - \sum_{j:d_{ij} \leq D^{\alpha_i}} \tilde{y}_j)$

  $= D^0 \sum_{j \in J_i^0} \hat{x}_{ij} + ... + D^{\alpha_i} \sum_{j \in J_i^{\alpha_i}} \hat{x}_{ij} + D^{\alpha_i+1} \sum_{j \in J_i^{\alpha_i+1}} \hat{x}_{ij}$ as $\hat{x}_{ij}, j \in J_i^{\alpha_i+1}$ satisfy (18)

  $= \sum_{j=1}^{M} d_{ij} \hat{x}_{ij}$.

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) on (LPC) may be strict. 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 Section 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 $(LPC - SC)$.

Of course, the two MILP (PC) and $(PC - SC)$ are polynomial in size. But they have different structures. 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 gives the number of different values in the distance matrix and is therefore lower than $N \times M$. In the worst case where K is close to $N \times M$, the two problems have almost the same number of variables but problem $(PC - SC)$ has about N times more constraints.

As suggested by Observation 1 and Observation 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 define a simple lower bound LB_0 and a simple upper bound UB_0. Then we define slightly more sophisticated bounds LB_1 and UB_1. Finally, we give experimental results showing the impact of tighter bounds on the behaviour of each of the formulations (PC) and $(PC - SC)$.

The bounds LB_0 and UB_0 come from the observation that the optimal radius is a decreasing function of p. The lower bound LB_0 is the trivial solution value of the M-Center problem. Suppose all the facilities are opened and each client is allocated to the closest facility, then the radius is $LB_0 = Max_{i=1,N}(Min_{j=1,M} d_{ij})$.

Now, for any facility F_j, let $\gamma_j = Max_{i=1,N}(Min_{h \neq j} d_{ih})$. This expression can be viewed as a minimal "cost" of closing facility F_j. 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 $\gamma_{j_1} \leq \gamma_{j_2} \leq ... \leq \gamma_{j_M}$ 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., $UB_0 = Min_{j=1,M}(Max_{i=1,N} d_{ij})$. The second upper bound UB_1 is obtained by the *Greedy* heuristic described by Mladenovic et al. in [15]. Once the 1-Center problem is solved, a second facility that minimizes the resulting radius is opened. The same iteration can be performed until p facilities are opened.

						(PC) is used				$(PC - SC)$ is used			
Instance	$N = M$	p	Opt	LB_0	UB_0	LP	#nodes	Gap	cpu	LP	#nodes	Gap	cpu
Pmed1	100	5	127	0	186	91	28	-	3600	108	38	-	3600
Pmed2	100	10	98	0	178	64	51	-	3600	85	82	20%	3600
Pmed3	100	10	93	0	205	63	107	-	3600	82	50	11%	3600
Pmed4	100	20	74	0	204	42	160	-	3600	61	96	22%	3600
Pmed5	100	33	48	0	169	20	2598	10%	3600	35	214	0%	2500

Table 1: Solving (PC) and $(PC - SC)$ with $LB = LB_0$ and $UB = UB_0$
- : no integer feasible solution was found after 3600s of CPU time

						(PC) is used				$(PC - SC)$ is used			
Instance	$N = M$	p	Opt	LB_1	UB_1	LP	#nodes	Gap	cpu	LP	#nodes	Gap	cpu
Pmed1	100	5	127	59	138	98	1733	22%	3600	108	1065	0%	3200
Pmed2	100	10	98	56	117	77	3184	0%	3500	86	66	0%	340
Pmed3	100	10	93	55	146	74	1114	32%	3600	84	105	0%	1100
Pmed4	100	20	74	41	111	55	8081	0%	2100	65	360	0%	1300
Pmed5	100	33	48	23	92	31	1748	0%	450	38	144	0%	260

Table 2: Solving (PC) and $(PC - SC)$ with the tighter bounds $LB = LB_1$ and $UB = UB_1$

Tables 1 and 2 show experimental results for five instances from OR-Lib [2] of size $N = M = 100$. The 40 instances coming from OR-Lib are usually used to for the p-median problem but have also been used for the p-Center problem, in order to compare heuristic solution methods in [15] and to test an exact solution method in [5]. We call these instances p-median instances. The MIPs are solved by use of MIPCplex from the CPLEX 7.0 commercial software [3]. The first three columns of each table give the characteristics

of the instance, the fourth columns give its optimal value. The following 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 to 10 (resp. 11 to 14) indicate the results obtained by executing the same MIP solver on formulation (PC) (resp. $(PC-SC)$) together with a preprocessing phase based on Observation 1 (resp. Observation 2). Columns LP contain the rounded up value of the linear relaxation of the formulation, Columns *#nodes* contain the number of *Branch & Bound* nodes. Columns *Gap* contain the relative gap at the end of the *Branch & Bound*, i.e. either 0% if the problem is solved to optimality within 1 hour of CPU time or the gap (in the sense of MIPCplex) between the best found lower bound and integer solution value after one hour. Columns cpu indicate 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 formulation (PC). They also indicate how each formulation takes advantage from 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 the 40 p-median instances from OR-Lib [2].

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 $j = 1, ..., M$. Let LB^* be the optimal value of $(PC-SC-R)$.

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

Proof
Let (LSC_α) be the LP-relaxation of problem (SC_α). For problem $(PC-SC-R)$, $z^k = 0$ in an optimal solution iff the optimal solution value to problem $(LSC_{D^{k-1}})$ is lower than p. Further the relation $z^{k+1} \geq z^k$ still holds for problem $(PC-SC-R)$. Hence, in an optimal solution $(\tilde{y}, \tilde{z})$ to $(PC-SC-R)$, $\tilde{z}$ has the form $\tilde{z}^1 = ... = \tilde{z}^{h^*} = 1$, $\tilde{z}^{h^*+1} = ... = \tilde{z}^K = 0$ where h^* is the smallest $h \in \{0, ..., K-1\}$ such that $v(LSC_{D^h}) \leq p$. This is equivalent to $LB^* = v(PC-SC-R) = D^{h^*}$ and the computation of LB^* can be done by solving at most $K \leq N \times M$ linear set-covering problems (LSC_α).

□

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

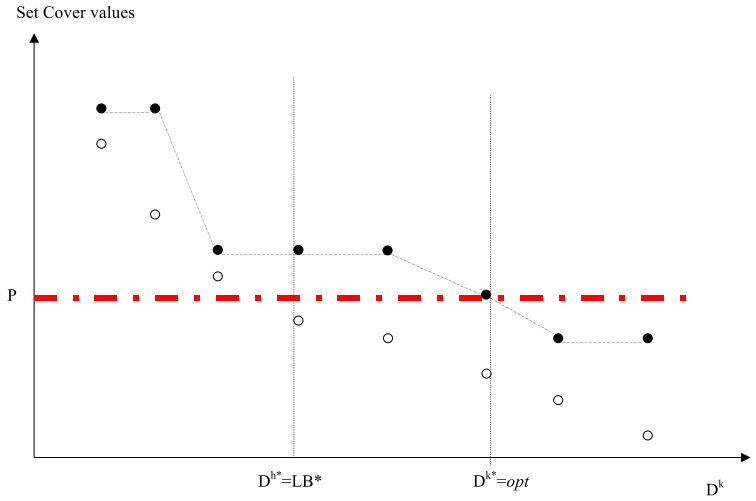

Figure 1: Illustration of h^* and k^*. $\bullet : v(SC_{D^k})$, $\circ : v(LSC_{D^k})$.

$(PC - SC - R)$ and $(PC - SC)$. Figure 1 illustrates the definitions and relation between h^* and k^*.

Using an LP solver to compute $v(LSC_\alpha)$, one can easily compute LB^* by searching h^* in the interval $[0, K]$ or in a smaller interval if a better upper bound and/or a lower bound on the optimal radius are known. The search can be done by dichotomy, reducing the maximal number of (LSC_α) to be solved to $log_2(K + 1)$. Our algorithm *Bsearch* takes as input a p-Center instance and the value of the solution computed by the greedy algorithm, UB_1. 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 to get LB^* is to compare $v(LSC_{D^h})$ 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, $h^* \in \{head + 1, ..., tail\}$.

Algorithm *Bsearch*
Input: N, M, P, UB_1, distances $D^0, ..., D^K = UB_1$, and distances $d_{ij}, i = 1, ..., N, j = 1, ..., M$.
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 $rc_j, j = 1, ..., M$;

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 [1]. Let n_i be the number of facilities which 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 which minimizes the number of uncovered clients. Once all clients are covered, we try to close redundant facilities, i.e. facilities which can be closed without uncovering any client. Step 4. and Step 9. are performed in order 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 which 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 in general. Step 7. is performed if the optimal solution of the LP-relaxed set-covering problem 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 to 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. Of course, since $0 \leq rc_j \leq 1$, this procedure 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_α) with $\alpha \leq D^h$ but in our 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 correctenss of algorithm *Bsearch*, one can observe that the following properties are satisfied every time steps 1 to 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 p-median instances of OR-Lib [2]. Columns 1 to 3 give the characteristics of the instance, 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 4 instances *Pmed1, Pmed6, Pmed26, and Pmed32.* For 29 over 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 to compute 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. Althougth 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 real variables, and 13901 constraints. MIPCplex needed 40 minutes and 178 *Branch & Bound* nodes (cutting planes were inhibited) to compute LB^* i.e. algorithm *Bsearch* is about 6000 times faster for that instance.

4 Worst case perfomance of LB^* and relation to an early heuristic

Hochbaum and Shmoys [6] considered the symmetric p-Center problem and proposed a 2-approximation algorithm based on its relation with both the dominating set problem and the strong stable set problem. Recall that in the symmetric case, the facilities have to be placed among the set of clients. For a given distance D^k, let $G_{D^k} = (V, E)$ be the graph where V is the set of clients and an edge exists between two vertices if the distance between them is not larger than D^k. There exists a solution with a radius at most D^k if a dominating set S of cardinality at most p exist for graph G_{D^k}. The problem of finding the smallest cardinality dominating set is dual to the problem of finding the largest strong stable set, in the usual linear programming sense of duality. A strong stable set S in graph G_{D^k} is a stable set with the additional restriction that every vertex not in S can be adjacent to at most one vertex in S. The heuristic of Hochbaum and Shmoys is based on a binary search for the smallest distance $D^{\tilde{k}}$ such that the solution $\tilde{S}$ of their greedy algorithm satisfies $|\tilde{S}| \leq p$. The greedy algorithm finds a set $S \subset V$ which is both a strong stable set in graph G_{D^k} and a dominating set in $G^2_{D^k}$, the square of G_{D^k}. From the triangle inequalities, they deduce that $\tilde{S}$ is also a dominating set in $G_{2D^{\tilde{k}}}$ and hence $\tilde{S}$ is a feasible solution for the p-Center with a radius not larger than $G_{2D^{\tilde{k}}}$. The optimality of this 2-approximation algorithm was proved by Hsu and Nemhauser [7]. Indeed, they

Instance	N=M	P	Opt	LB*	UB*	Time (s)
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

Table 3: Results of algorithm *Bsearch* for the 40 p-median instances.

showed that unless P=NP, no better polynomial δ-approximation algorithm can exist for the symmetric p-Center problem. Moreover, the heuristic has been implemented first by Martinich [13] and tested for random instances having up to 75 clients. More recently, Mladenovic et al. [15] also evaluated this heuristic for the 40 p-median instances from OR-Lib [2] and for 15 instances derived from a 1060 clients TSP instance from TSPLIB [16]. Their experimentations show that the solution of the heuristic of Hochbaum and Shmoys has an average deviation of about 50% from the best known solution, computed by a Variable Neighborhood Search or by a Tabu Search method.

A by-product of Hochbaum and Shmoys's algorithm is the fact that $D^{\tilde{k}}$ is a lower bound of the optimal radius. But, to our knowledge, the quality of this lower bound has never been estimated. In the remainder of this section, we show that the algorithm to compute the lower bound $D^{\tilde{k}}$ can be extended to the general p-Center problem and prove that the lower bound LB^* is at least as tight as $D^{\tilde{k}}$.
In the general p-Center problem and for a given distance D^k, let $\Delta_{D^k} = (V, E)$ be the neighborhood graph where V is the set of clients and an edge connects two clients C_i and C_r if there exist a facility F_j within the distance D^k from both C_i and C_r, or, in other words, C_i and C_r can be covered by the same facility within the radius D^k. It is easy to check that the LP derived from the problem of finding a maximal cardinality stable set on graph Δ_{D^k}, say (LSS_{D^k}), is the LP dual of problem (LSC_{D^k}) obtained by LP-relaxation of the set covering problem (SC_{D^k}). Hence, another way to compute the lower bound LB^* consists in solving a series of problems (LSS_{D^k}) rather than problems (LSC_{D^k}). Of course, the cardinality of any stable set in graph Δ_{D^k} is a lower bound on the optimal value of (LSS_{D^k}). Therefore, one can consider a heuristic for the maximal stable set problem and find the smallest distance $D^{\tilde{k}}$ such that this heuristic returns a stable set with less than p client nodes. This method for computing the lower bound $D^{\tilde{k}}$ is just the extention of Hochbaum and Shmoys's idea and by the weak duality theorem, $D^{\tilde{k}} \leq D^{h^*} = LB^*$. An immediate consequence of this is the following proposition.

Proposition 2 *If distances $d_{ij}, i = 1, ..., N, j = 1, ..., M$ satisfy triangle inequalities then the optimal value of the p-Center problem is not more than $2LB^*$.*

Proof When the triangle inequalities are satisfied, $D^{\tilde{k}} \geq \frac{1}{2}D^{k^*}$ follows from the 2-approximation result in [6], and we have shown above that $D^{\tilde{k}} \leq LB^*$.

□

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 a MIP solver, in the same way as to obtain Tables 1 and 2. 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 $LB^* = D^{h^*} \leq D^{h^*+1} \leq ... \leq D^{K'} = UB^*$. Algorithm *BsearchEx* is similar to our algorithm *Bsearch* of Section 3, except in the following steps:

- The Init. step is replaced by

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

- Step 6. is suppressed

- 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 of the new algorithm, $k^* \in \{head + 1, ..., tail\}$. At Step 8.ex, an optimal solution rather than a heuritic one is computed for problem (SC_{D^h}). In fact, we do not need to solve this problem to optimality but only 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 [3] and change some of its default parameters:

- *mip tolerances uppercutoff* to $(p + 0.001)$, i.e., integer solutions with a value larger than p are discarded,

- *mip limits solutions* to 1, i.e., the resolution stops as soon as an integer solution (with a value at most p) is found,

- *timelimit* to 3600, 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 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 that p is 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.

We applied our algorithm *BsearchEx* to the 11 instances in Table 3 where $LB^* < UB^*$. For the totality of the 11 instances, *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*.

Table 4 gives results for the 11 p-median instances for which $LB^* < UB^*$ (see Table 3) and for some TSP instances taken from TSPLIB [16]. Those instances have already been used by the authors of [15]. For those instances, usually devoted to the *Travelling Salesman Problem*, if N is the number of cities, we set $M = N$ and consider different values of p, typically 10, 20, 30, etc.... Hence, all the instances of Table 4 represent symmetric p-Center problems. The first 3 columns characterize the instance. Column 4 gives either the optimal value or the best found solution value. Columns 5 to 8 report the results

of Algorithm *Bsearch*. Column $\#LSC_\alpha$ contains the number of problems (LSC_α) which have been solved and *cpu1* is the cpu time devoted to Algorithm *Bsearch*. Column 9 gives $\#SC_\alpha$, the number of problems (SC_α) which have been solved in Algorithm *BsearchEx*. The last column, *cpu2* is the total cpu time devoted for solving the instance, i.e. to read the instance data, perform the greedy algorithm, sort the distances, and perform Algorithms *Bsearch* and *BsearchEx*.

Daskin [5] gives experimental results of his exact solution method for the p-Median instances. 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 the 40 p-Median instances is 9.1 seconds with a maximum of 59.1 seconds while Daskins' results show 91.5 seconds with a maximum of 1402.5 seconds. Daskin's results are obtained using a PC with a RAM of 128MB and a PentiumIII processor of 650Mhz.

Another comparison can be done with the work by Mladenovic, Labbé, and Hansen [15] whose objective were 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 heuristic. This deviation is of 2.37% for the 40 p-Median instances (average CPU time of 133.6 seconds for the *VNS* heuristic on a SUN Spark 10) and of 0.02% (average CPU time of 300 seconds) for the 15 u1060.tsp instances from TSPLIB.

We also wanted to check how does the quality of our lower bound LB^* varies if the distances do not satisfy the triangle inequalities. For this, we constructed a first problem *Euc100* where $N = M = 100$, coordinates of the points are randomly generated in [0,100], and Euclidean distances computed between the different points. Based on this problem, three instances are considered with different values for p. The second problem *Rand100* has similar properties but now, distances are randomly generated in [0,100], they are still symmetric and $d_{ii} = 0, i = 1, ..., N$. Table 5 reports the results obtained for these instances. It appears that lower bound LB^* is much tighter for the Euclidean instances. The relative gap $\frac{Opt-LB^*}{Opt}$ does not exeed 5% for any of the Euclidean instances we tested while it is equal to 28% for problem *Rand100* with $p = 5$.

6 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 and because it takes better advantage from the knowledge of bounds on the optimal value. Moreover, it guided us to the lower bound LB^* which is the optimal value of the problem obtained by the continous 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 *capacitated* p-Center (see [11] among others) and the *fault tolerant* p-Center (see [10] among others). 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.

Instance	$N = M$	p	**Opt**	LB^*	UB^*	$\#LSC_\alpha$	*cpu1*	$\#SC_\alpha$	*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	1060	10	2273	2273	2273	6	27	0	53
u1060	1060	20	1581	1556	1768	9	63	6	2778
u1060	1060	30	1208	1205	1275	8	50	4	298
u1060	1060	40	1021	1013	1079	9	35	4	366
u1060	1060	50	905	895	963	6	21	4	383
u1060	1060	60	781	765	807	8	21	4	233
u1060	1060	70	711	707	761	6	17	3	135
u1060	1060	80	652	652	711	6	18	2	60
u1060	1060	90	608	604	636	6	19	3	38
u1060	1060	100	570	570	570	5	18	0	29
u1060	1060	110	539	539	539	5	18	0	30
u1060	1060	120	510	510	538	7	29	1	44
u1060	1060	130	500	495	510	6	28	1	44
u1060	1060	140	452	452	500	5	28	3	46
u1060	1060	150	447	430	447	6	34	3	50
rl1323	1323	10	3077	3062	3329	11	106	8	1380
rl1323	1323	20	2016	2008	2152	11	115	5	480
rl1323	1323	30	1632	1611	1797	11	99	7	900
rl1323	1323	40	1352	1334	1521	10	76	7	3000
rl1323	1323	50	1187	1165	1300	9	61	7	8580
rl1323	1323	60	1063	1047	1194	11	55	8	9120
rl1323	1323	70	972	959	1040	10	42	7	1740
rl1323	1323	80	895	889	948	10	37	5	420
rl1323	1323	90	832	830	857	10	30	4	120
rl1323	1323	100	787	777	803	9	26	5	120
u1817	1817	10	458	455	467	8	611	4	2700
u1817	1817	20	310?	306	342	9	660	5	4920
u1817	1817	30	250?	240	287	8	355	6	16500
u1817	1817	40	210?	205	234	6	247	4	6420
u1817	1817	50	187?	180	205	8	242	4	9840
u1817	1817	60	163	163	183	6	177	4	1260
u1817	1817	70	148	148	152	6	166	1	420
u1817	1817	80	137	137	148	5	150	3	1140
u1817	1817	90	130?	127	148	6	161	3	7202
u1817	1817	100	127	127	130	6	159	1	300
u1817	1817	110	109	108	127	5	119	4	420
u1817	1817	120	108	108	108	6	131	0	120
u1817	1817	130	108?	105	109	4	121	1	3720
u1817	1817	140	105?	102	108	5	121	2	4020
u1817	1817	150	94?	92	108	5	144	3	5640

Table 4: Results of our exact solution method for instances from OR-Lib and TSPLIB. ? : Opt is the value of the best found solution for that instance.

Instance	$N = M$	p	**Opt**	LB^*	UB^*	$\#LSC_\alpha$	*cpu1*	$\#SC_\alpha$	*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

Table 5: Results of our exact solution method for random instances.

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 & Bound* algorithm can lead to a better global performance.

Our computational results show that LB^* is a very tight lower bound for Euclidean instances. For non-Euclidean 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 usefull topic for further research. Indeed, the addition of strong valid inequalitites to the new formulation and/or to the related set-covering problems might yield a tighter lower bound.

While finishing the writing of this paper, we were aware of a very recent work by Ilhan and Pinar [8]. An exact solution method for the p-Center problem is proposed. Their solution method is composed of two phases : the first phase called *LP-Phase* turns 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_α) 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 on their results for instances from [2], one can however notice that our bound LB^* is often slightly better than their *LP-Phase* bound. We believe this is due to computational precision.

References

[1] E. Balas, A. Ho. Set covering algorithms using cutting planes, heuristics and subgradient optimization: a computational study. *Mathematical Programming Study 12, 37-60*, 1980.

[2] J.E. Beasley. OR-Library: Distributing test problems by electronic mail. *Journal of the Operational Research Society 41(11), 1069-1072*, 1990.

[3] *Cplex* ILOG CPLEX Division, 889 Alder Avenue, Suite 200, Incline Village, NV 89451

[4] M. Daskin. Network and discrete location. Wiley, New York, 1995.

[5] M. Daskin. A New Approach to Solving the Vertex P-Center Problem to Optimality: Algorithm and Computational Results. *Communications of the Operations Research Society of Japan 45:9, 428-436*, 2000.

[6] D. Hochbaum, D. Shmoys. A best possible heuristic for the k-center problem. *Math. Oprns. Res. 10, 180-184*, 1985.

[7] W.L. Hsu, G.L. Newhauser. Easy and hard bottleneck location problems. *Discrete Appl. I, 209-216*, 1979.

[8] T. Ilhan , M.C. Pinar An Efficient Exact Algorithm for the Vertex p-Center Problem *http://www.optimization-online.org/DB_HTML/2001/09/376.html*, 2001

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

[10] S. Khuller, R. Pless and Y. Sussmann. Fault Tolerant K-Center Problems. *Theoretical Computer Science, 242, 237-245*, 2000.

[11] S. Khuller and Y. Sussmann. The Capacitated K-Center Problem. *SIAM Journal on Discrete Mathematics, Vol 13(3), 403-418*, 2000.

[12] V. Marianov, C. ReVelle. Siting Emergency Services. *in Facility Location: A Survey of ARTICLES Applications and Methods, Drezner, Z., ed. Springer Series in Operations Research, 199-222*, 1995.

[13] J. Martinich. A vertex closing approach to the p-Center problem. *Naval Res. Log. 35, 185-201*, 1988.

[14] N. Megiddo, A. Tamir, E. Zemel, and R. Chandrasekaran. An $O(nlog^2n)$ algorithm for the k^{th} longest path in a tree with applications to location problems. *SIAM Journal on Computing, 10(2), 328-337*, 1981.

[15] N. Mladenovic, M. Labbé, P. Hansen. Tabu search and variable Neighborhood search in solving the p-Center problem. *ULB-ISRO preprint 2000/20*, (available at http://smg.ulb.ac.be/).

[16] G. Reinelt. TSPLIB-a traveling salesman problem library. *ORSA Journal on Computing, 3, 376-384*, 1991.

[17] C. Revelle C. Toregas, R. Swain and L. Bergman, The location of emergency service facilities. *Operations Research 19, 1363-73*, 1971.