Computers & Operations Research 40 (2013) 2991–2999

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

Double bound method for solving the *p*-center location problem

Hatice Calik*, Barbaros C. Tansel

Bilkent University, Department of Industrial Engineering, 06800 Bilkent, Ankara, Turkey

ARTICLE INFO

Available online 19 July 2013

Keywords:
p-Center location
Multi-center location
Covering location
Minimax location
Set covering

ABSTRACT

We give a review of existing methods for solving the absolute and vertex restricted *p*-center problems on networks and propose a new integer programming formulation, a tightened version of this formulation and a new method based on successive restrictions of the new formulation. A specialization of the new method with two-element restrictions obtains the optimal *p*-center solution by solving a series of simple structured integer programs in recognition form. This specialization is called the double bound method. A relaxation of the proposed formulation gives the tightest known lower bound in the literature (obtained earlier by Elloumi et al., [1]). A polynomial time algorithm is presented to compute this bound. New lower and upper bounds are proposed. Problems from the OR-Library [2] and TSPLIB [3] are solved by the proposed algorithms with up to 3038 nodes. Previous computational results were restricted to networks with at most 1817 nodes.

1. Introduction

The *p*-center problem is a relatively well known facility location problem that involves locating p identical facilities on a network to minimize the maximum distance between demand nodes and their closest facilities. The focus is on the minimization of the worst case possible time spent on the way in providing service. This sort of objective is more meaningful than total cost objectives for problems with a time sensitive service structure. A majority of the applications arise in emergency service locations such as determining optimal locations of ambulances, fire stations and police stations where the human life is at stake. There is also an increased interest in *p*-center location and related location covering problems in the contexts of terror fighting, natural disasters and human-caused disasters.

Our primary interest in the *p*-center problem is from a modeling and algorithmic perspective. We give a review of existing approaches, propose a new formulation and a new method based on this formulation. We obtain a new integer programming model with larger linear programming (LP) bounds by tightening one set of constraints in our model. Additionally, a semi-relaxation of our proposed model gives the tightest lower bound obtained earlier by [1]. We give a polynomial time algorithm to compute the lower bound by solving a finite number of linear programming problems of polynomial size. The predominant approach in the literature is to perform a bisection or binary search over an interval between lower and upper bounds on the optimal value of the *p*-center problem and solves a sequence of set covering problems for each selected value of cover radius. We differ from the set covering approach in that we use restrictions of the proposed formulation to converge to an optimal solution. While the restriction approach is general enough to allow many variations as dependent on how one chooses restrictions during the process, we focus on a particularly simple restriction which we refer to as the double-bound method. We evaluate the computational merits of the proposed formulations and certain variants of the double bound method using test problems from the OR-Library and TSPLIB. We are able to solve some large problems that are reported unsolved in the previous literature. We provide additional larger sized test problems that have not been attempted previously. With the proposed methodology, we are able to solve problems with up to 3038 nodes.

Let $G=(V,E)$ be an embedded network with vertex set $V=\{v_1,\ldots,v_n\}$ and edge set E. The following definitions are like in [4]. An edge with end vertices v_i and v_j in an embedded network is the image $[v_i,v_j]\equiv T_{ij}([0,1])$ of the unit interval $[0,1]$ in some space S (e.g. the plane) under a continuous one-to-one mapping $T_{ij}:[0,1]\to S$ with $T_{ij}(0)=v_i, T_{ij}(1)=v_j$. For any point $x\in[v_i,v_j]$, denote by $[v_i,x]$ and $[x,v_j]$ the sub-edges defined by point x. Note that $[v_i,x]\cap[x,v_j]=\{x\}$ while $[v_i,x]\cup[x,v_j]=[v_i,v_j]$. Let $l_{ij}>0$ be a length assigned to each edge $[v_i,v_j]\in E$. For $x\in[v_i,v_j]$, the lengths of sub-edges $[v_i,x]$ and $[x,v_j]$ are defined as λl_{ij} and $(1-\lambda)l_{ij}$, respectively, where λ is the unique real number $\lambda\in[0,1]$ such that $x=T_{ij}(\lambda)$. We take G as a point set defined by the union of its embedded edges. For any two points $x,y\in G$, define $d(x,y)$ to be the length of a shortest path between points x and y. For any positive integer p, let $S_p(G)$ be the set of all p-element subsets of G. For $X\in S_p(G)$ and $v_i\in V$, denote the elements of X by $x_1,\ldots,x_p$ and define

* Corresponding author. Tel.: +90 3122903269.
E-mail addresses: calik@bilkent.edu.tr, hatice.calik@gmail.com (H. Calik).

0305-0548/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cor.2013.07.011

the closest distance between the point set X and vertex v_i by $D(X,v_i)=\min\{d(x_1,v_i),\ldots,d(x_p,v_i)\}$. For $X\in S_p(G)$, define $f(X)=\max\{D(X,v_i): v_i\in V\}$. The absolute p-center problem is to find a point set $X^*\in S_p(G)$ such that $r_p(G)=f(X^*)\le f(X), \forall X\in S_p(G)$. We refer to X^* in the foregoing definition as an absolute p-center and the optimal value $r_p(G)$ as the p-radius of G. For any finite subset F of G with $|F|\ge p$, define $S_p(F)$ to be the set of all p-element subsets of F. The F-restricted p-center problem is to find a point set $X^*\in S_p(F)$ such that $f(X^*)\le f(X), \forall X\in S_p(F)$. If $F=V$, the resulting problem is referred to as the vertex restricted problem. Denote by p-C, p-VC and p-FC the absolute, vertex restricted and F-restricted p-center problems, respectively, and denote their optimal values (p-radii) by $r_p(G)$, $r_p(V)$ and $r_p(F)$, respectively. In the sequel, the term "p-center problem" refers to any of the aforementioned variants.

In the weighted version of the p-center problem, there is a positive weight w_i associated with each vertex $v_i\in V$ and the definition of $f(X)$ is revised as $f(X)=\max\{w_iD(X,v_i): v_i\in V\}$. All of the definitions in the previous paragraph apply just the same with this definition of $f(\cdot)$.

Next, we give an r-cover problem which is closely related to the p-center problem. Let $N=\{1,\ldots,n\}$. For a fixed nonnegative real number r, define C(r) to be the following optimization problem: $\min\{|X|: X\subset G \text{ and } D(X,v_i)\le r, i\in N\}$. This problem seeks to place the minimum number of centers (facilities) on a network G such that each vertex has at least one center within a distance of r. In the weighted case, the distance constraints $D(X,v_i)\le r$, $i\in N$, in the definition of C(r) are replaced by the weighted distance constraints $w_iD(X,v_i)\le r$, $i\in N$. For the vertex and F-restricted cases, the condition $X\subset G$ is replaced by $X\subset V$ and $X\subset F$, respectively, and the resulting problems are referred to as VC(r) or FC(r), respectively.

In the next section we give a detailed review of the related works on the p-center problem. In Section 3, we present our new IP formulation and prove that this formulation solves the p-center problem optimally. Additionally, we give a tightened version of our formulation and the experimental results obtained from the computational comparison of our formulations with the previous formulations. In Section 4, we make a comparison between the LP relaxations of our formulations and the previous formulations. We present a lower bound that we obtain from our IP formulation by relaxing the binary restriction on one set of variables. We prove that this bound is equivalent to the tightest known lower bound in the literature and provide a polynomial time algorithm to obtain this bound. In addition to the relaxation bounds, we provide new lower and upper bounds which can be obtained very quickly. In Section 5, we first give the underlying idea of our method, and then we give the general structure of our double bound algorithm, which is a specialization of our solution method. We introduce six variations of the double bound algorithm. Among the six variations, we select the one that requires least amount of time on the average and give the computational results obtained from this algorithm in Section 4. In Section 6, we give the experimental results obtained from our algorithm. We solve problems from OR-Library and TSPLIB in these experiments. Finally, we conclude in Section 7.

2. Related work

The absolute 1-center problem is initially defined by Hakimi [5] and solved using graphical methods by taking advantage of the piecewise linearity of the function $f(x)$ on any edge. Piecewise linearity for the absolute 1-center problem has important consequences for $p>1$ as it leads to the existence of a finite point set $P\subset G$ such that there exists an absolute p-center X^* in $S_p(P\cup V)$. This is initially observed by Minieka [6] and extended to the weighted case by Kariv and Hakimi [7]. This property is generalized later by Hooker et al. [8] to a more general setting. Points in P are referred to as intersection points. A point x in G qualifies as an intersection point if there exist two distinct vertices v_r and v_s such that x is the unique point in its edge for which $d(v_r,x)=d(x,v_s)$. For the weighted case, a point x in G qualifies as an intersection point if there exist two distinct vertices v_r and v_s such that $w_rd(v_r,x)=w_sd(x,v_s)$ and there exists a positive real number ε such that $\max\{w_rd(v_r,x'),w_sd(v_s,x')\}>w_rd(v_r,x)$ for all points x' for which $0<d(x,x')<\varepsilon$. There are at most $O(n^2)$ intersection points on any given edge and $O(n^2|E|)$ points on the entire network. Because the absolute p-center problem reduces to a search over a finite subset $P\cup V$ of G, we focus on the F-restricted problem where F is any finite subset of G with $F=P\cup V$ for p-C and $F=V$ for p-VC. Let f_j, $j\in M\equiv\{1,\ldots,m\}$, be an enumeration of the points in F, with $m=|P\cup V|$ for p-C and $m=n$ for p-VC.

The p-center problem is NP-hard for general networks [7] even if the network G is planar with unit vertex weights, unit edge lengths and with maximum vertex degree 3. Low order polynomial time algorithms are available for tree networks [7,9–13]. A review of the early theory and algorithms for network location problems, including the p-center problem, is given in [14,15] (see also [16,17]). Various theoretical and algorithmic aspects of the p-center problem for general and tree networks are discussed in [18]. A concise review of the computational state of the art including approximation methods is given in [1].

There are two IP formulations of the p-center problem in the literature proposed by Daskin [17] and Elloumi et al. [1].

Daskin [17]'s formulation is for the vertex restricted case. Define a binary variable y_j with $y_j=1$ if a center is placed at vertex v_j and 0 otherwise. Define binary variables x_{ij} to be 1 if v_i assigns to a center placed at v_j and 0 otherwise. The formulation of Daskin [17], referred to as P1 in the sequel, is as follows:

$$(\text{P1}): \quad \min \quad z \tag{1}$$

$$\text{s.t.} \quad \sum_{j\in N} d_{ij}x_{ij}\le z \quad \forall i\in N \tag{2}$$

$$\sum_{j\in N} x_{ij}=1 \quad \forall i\in N \tag{3}$$

$$x_{ij}\le y_j \quad \forall i,j\in N \tag{4}$$

$$\sum_{j\in N} y_j\le p \tag{5}$$

$$y_j\in\{0,1\} \quad \forall j\in N \tag{6}$$

$$x_{ij}\in\{0,1\} \quad \forall i,j\in N. \tag{7}$$

Constraints (3) assign each vertex to exactly one center and (1) and (2) ensure that the objective value is no less than the maximum vertex-to-center distance. Constraints (4) ensure that no vertex assigns to v_j unless there is a center at v_j. Constraint (5) restricts the number of centers to p. Constraints (6) and (7) are the binary restrictions. We may extend Daskin [17]'s formulation to p-FC by replacing "$j\in N$" with "$j\in M$".

The second IP formulation is due to Elloumi et al. [1]. Their formulation is for p-FC and is similar to a canonical representation of the simple plant location problems given earlier by Cornuéjols et al. [19]. Let $\rho_1<\rho_2<\cdots<\rho_K$ be an ordering of the distinct distance values of the n by m matrix of distances $d_{ij}\equiv d(v_i,f_j)$. Define y_j to be the same as in P1 and the additional binary variables u^k, $k=2,\ldots,K$, with $u^k=0$ only if all vertices can be covered within a radius value of ρ_{k-1} and $u^k=1$ otherwise. Denote by P2 the formulation of Elloumi et al. [1] given below:

$$(\text{P2}): \quad \min \quad \rho_1+\sum_{k=2}^{K}(\rho_k-\rho_{k-1})u^k \tag{8}$$

$$\text{s.t.} \quad \sum_{j\in M} y_j \geq 1 \tag{9}$$

$$\sum_{j\in M} y_j \leq p \tag{10}$$

$$u^k + \sum_{j:d_{ij}<\rho_k} y_j \geq 1 \quad \forall i\in N, k=2,\ldots,K \tag{11}$$

$$y_j \in \{0,1\} \quad \forall j\in M \tag{12}$$

$$u^k \in \{0,1\}, \quad k=2,\ldots,K. \tag{13}$$

Constraint (9) is required to eliminate null solutions (with no center). Constraint (10) is the same as constraint (5). Constraints (11) and the objective function (8) ensure that all vertices are covered by their closest centers.

Existing solution methods are either based on solving a sequence of set covering problems or enumerating p-element subsets of F. For fixed $r>0$, the set covering problem is the problem of minimizing $\sum_{j=1}^{m} y_j$ subject to $Ay\geq 1, y\in\{0,1\}^m$ where $A=[a_{ij}]$ is an n by m matrix of zeros and ones with $a_{ij}=1$ if $d(v_i,f_j)\leq r$ and $a_{ij}=0$ if $d(v_i,f_j)>r$. Denote the set covering problem by SC(r). Note that SC(r) is the traditional way of solving the r-cover problem C(r).

The first set-covering based approach is proposed by Minieka [6] for p-C. Minieka [6] presented a systematic method to update the set covering matrix to converge to an optimal solution in a finite number of steps. His updating method takes advantage of the fact that the distances d_{ij} ($i\in N, j\in M$) are the only possible values for $r_p(F)$. This leads to an updating of A that corresponds to successive reductions of r over the list of distinct distance values. Christofides and Viola [20] solved the weighted problem by first generating regions in the network reachable by at least one vertex within a radius of r and solving a set covering problem that selects the fewest number of points from these regions. This approach is the same as the set covering approach but does not make use of the finite dominating set $P\cup V$. Toregas et al. [21] solved p-VC by solving a linear programming relaxation of the associated set covering problem and adding a cut in case of fractional solutions. Garfinkel et al. [22] solved a sequence of set covering problems but they first reduced the search space by finding a heuristic solution X and eliminating from F all those points whose relative radii are greater than $f(X)$. Daskin [17] gives the first IP formulation (P1 above) of p-VC but prefers to use a set covering based bisection search over an interval defined by pre-computed lower and upper bounds on $r_p(V)$. Daskin [23] improves this algorithm by solving, via Lagrangian relaxation, a maximal covering location problem in which the total number of vertices that are covered within r is maximized while the number of open facilities is restricted to p. Ilhan and Pinar [24] propose a two phase extension of Daskin [17]'s algorithm for the vertex restricted problem. In the first phase, several LP relaxations of the set covering problem are solved as feasibility problems with the addition of the constraint $\sum_{j\in M} y_j \leq p$ to find an appropriate lower bound on $r_p(V)$. In the second phase, several set covering problems with the additional constraint $\sum_{j\in M} y_j \leq p$ are solved by systematically changing the objective value starting from this lower bound. Al-khedhairi and Salhi [25] propose some modifications to the algorithms of Daskin [17] and Ilhan and Pinar [24]. Elloumi et al. [1] propose a different IP formulation (P2 above) for p-FC. They give a lower bound which is tighter than the LP relaxations of P1 and P2 and a polynomial time method to compute it via a sequence of LPs. They solve p-FC by performing a binary search over the ordered list of distinct values of distances that are between their proposed lower and upper bounds. A set covering problem is solved for each selected distance value between the bounds. They solve problems from the OR-Library and TSPLIB with up to 1817 nodes using binary search. To our knowledge, this is the largest network size solved previously.

We note here that the term "bisection search" refers to successively halving a real interval and discarding either the lower or the upper half in each step until its size is smaller than a predetermined positive real number whereas the term "binary search" refers to performing essentially the same operation on a finite list of numbers using a median element of the list.

3. Proposed formulation (P3)

We now propose a new formulation of p-FC. Define $R=\{\rho_1, \rho_2, \ldots, \rho_K\}$ with $\rho_1<\rho_2<\cdots<\rho_K$ as defined earlier. Exactly one of these values determines the value of $r_p(F)$. Associate a binary variable z_k with ρ_k, $k\in T\equiv\{1,\ldots,K\}$ with $z_k=1$ if $r_p(F)=\rho_k$ and 0 if not. For $i\in N, j\in M$ and $k\in T$, define the parameters $a_{ijk}=1$ if $d_{ij}\leq\rho_k$ and 0 otherwise. We use the variables y_j as before; that is, $y_j=1$ if a center is placed at site f_j and 0 otherwise. The proposed formulation, referred to as P3, is as follows:

$$(\text{P3}): \quad \min \quad \sum_{k\in T} \rho_k z_k \tag{14}$$

$$\text{s.t.} \quad \sum_{j\in M} a_{ijk} y_j \geq z_k \quad \forall i\in N, \forall k\in T \tag{15}$$

$$\sum_{j\in M} y_j \leq p \tag{16}$$

$$\sum_{k\in T} z_k = 1 \tag{17}$$

$$y_j \in \{0,1\} \quad \forall j\in M \tag{18}$$

$$z_k \in \{0,1\} \quad \forall k\in T. \tag{19}$$

Constraint (17) ensures that exactly one of the variables z_k is selected as 1 and the objective function (14) determines the value of $r_p(F)$ as the corresponding value ρ_k. Constraints (15) ensure that each vertex is covered within the selected radius by at least one center. Constraint (16) restricts the number of centers to at most p. Constraints (18) and (19) are binary restrictions.

For any feasible solution (y, z) of P3, we can obtain a feasible solution (y, u) for P2, which provides exactly the same objective value, by setting

$$u^k = \sum_{q=k}^{K} z_q, \quad k=2,\ldots,K. \tag{20}$$

The reverse can also be achieved by using

$$\begin{aligned} &z_k = u^k - u^{k+1}, \quad k=2,\ldots,K-1,\\ &z_K = u^K,\\ &z_1 = 1-u^2. \end{aligned} \tag{21}$$

By using this relationship, we can obtain a tighter constraint and replace it with (15). When we consider all distinct distance values in the increasing order, (11) implies $u^k + \sum_{j:d_{ij}\leq\rho_{k-1}} y_j \geq 1, \forall i\in N$, $k=2,\ldots,K$. Replacing u^k with $\sum_{q=k}^{K} z_q$ and right hand side with $\sum_{q=1}^{K} z_q$ we obtain $\sum_{j:d_{ij}\leq\rho_k} y_j \geq \sum_{q=1}^{k} z_q, \forall i\in N, k=1,\ldots,K-1$. For $k=K$, $\sum_{j:d_{ij}\leq\rho_k} y_j = \sum_{j\in M} y_j \geq 1$. Then we can replace (15) with

$$\sum_{j\in M} a_{ijk} y_j \geq \sum_{q=1}^{k} z_q, \quad \forall i\in N, \forall k\in T. \tag{22}$$

The new formulation, referred to as (P4), with the tightened constraints is basically as follows:

$$(\text{P4}): \quad \min \quad (14)$$
$$\text{s.t.} \quad (16)\text{–}(19), \text{ and } (22).$$

P3 and P4 have $m+K$ binary variables and $nK+2$ constraints. P2 has $m+K-1$ binary variables and $n(K-1)+2$ constraints. On the other hand, P1 (as adapted to p-FC) has one real variable, $m+mn$ binary variables and $2n+mn+1$ constraints. Since K is at most mn, P2, P3 and P4 have $O(mn)$ binary variables and $O(mn^2)$ constraints, which is $O(n)$ times more than the number of constraints of P1.

While P1 is a more compact formulation of p-FC than P2, P3 and P4, the computational performances of P2, P3 and P4 are far better than that of P1 when all four formulations are directly solved by a commercial solver. Table 1 gives a comparison of solution times and optimal or best-found objective values for P1, P2, P3 and P4 for the vertex restricted case using 40 p-median instances taken from the OR-Library. These instances are solved for each of the formulations using IBM ILOG CPLEX 12.4 [26] concert technology with MIPEmphasis option of CPLEX set equal to 1. The possible values of $r_p(V)$ in R that are needed in P2, P3 and P4 are restricted to a subset $R' \equiv R \cap [LB2, UB2]$ where $LB2$ and $UB2$ are proposed lower and upper bounds on $r_p(F)$ (see Section 4). In this table, columns 5, 6, 7 and 8 give the solution times in seconds taken by the solver for the IP models P4, P3, P2 and P1, respectively. While solving the three IP models, we put a time restriction of 3 h. P2, P3 and P4 can solve each problem optimally within the 3 h limit while P1 cannot solve 17 of these problems optimally. Of the 17 unsolved instances, 5 of them are instances for which no feasible integer solution can be found by P1 within the 3 h limit. These are indicated by NFS (No Feasible Solution) in the middle column under P1 in the table. For the 12 instances that are solved sub-optimally by P1, the last column under P1 gives the percent gap reported by CPLEX. When we compare P2, P3 and P4 with each other, we see that the largest time required is 772.68 s for P4 (instance pmed39) while it is 1232.92 s for P3 (instance pmed39) and 1428.94 s for P2 (instance pmed8). The average time required to solve 40 instances with P4 is 82.25 s while it is 99.80 s for P3 and 155.25 s for P2. The differences in computational times are more significant in some instances. For example, P3 solves pmed8 in 10.31 s and P4 solves the same instance in 12.74 s while P2 solves it in 1428.94 s, which is more than 138 times that of P3 and 112 times that of P4. Among 40 instances, for 7 of them P2 achieves the smallest solving time while P3 is the successor for 22 of them and P4 is the successor for 11 of them. We may conclude from these observations that the proposed model P3 and its tightened version P4 perform noticeably better than P2 on the 40 p-median instances taken from OR-Library and that P2, P3 and P4 perform significantly better than P1. While direct solution times are reasonable for P2, P3 and P4, the double bound algorithms that we give in Section 5 lead to much shorter solution times than direct solution times available in this table.

Table 1
Solving times (s) of IP models.

Instance	n	p	Opt	P4 time	P3 time	P2 time	P1		
							Time	Obj	Gap%
pmed1	100	5	127	11.33	4.38	52.78	188.81	127	
pmed2	100	10	98	7.02	2.00	12.94	73.34	98	
pmed3	100	10	93	5.26	1.90	7.72	37.89	93	
pmed4	100	20	74	1.30	0.21	8.35	0.90	74	
pmed5	100	33	48	1.08	0.26	1.45	0.39	48	
pmed6	200	5	84	11.50	15.81	33.50	2797.12	84	
pmed7	200	10	64	26.89	16.67	96.79	4476.78	64	
pmed8	200	20	55	12.74	10.31	1428.94	776.21	55	
pmed9	200	40	37	4.71	0.68	3.57	6.40	37	
pmed10	200	67	20	0.62	0.13	0.34	1.40	20	
pmed11	300	5	59	9.09	15.08	14.01	537.31	59	
pmed12	300	10	51	34.74	38.42	28.93	7115.81	51	
pmed13	300	30	36	16.05	19.83	42.15	3188.15	36	
pmed14	300	60	26	5.60	2.19	4.92	281.05	26	
pmed15	300	100	18	1.33	0.29	0.81	3.82	18	
pmed16	400	5	47	5.76	4.18	33.91	10 800.00	47	6.78
pmed17	400	10	39	45.21	50.32	38.33	10 800.00	41	14.63
pmed18	400	40	28	71.64	48.98	55.39	10 539.49	28	
pmed19	400	80	18	4.91	2.24	5.43	33.34	18	
pmed20	400	133	13	1.33	0.28	0.75	3.79	13	
pmed21	500	5	40	12.36	17.95	35.65	10 800.00	43	16.28
pmed22	500	10	38	239.84	373.73	1223.58	10 800.00	41	18.37
pmed23	500	50	22	185.59	54.86	129.87	10 800.00	24	16.67
pmed24	500	100	15	10.33	3.46	4.78	439.70	15	
pmed25	500	167	11	1.61	0.26	0.73	17.13	11	
pmed26	600	5	38	32.64	72.78	57.96	10 800.00	39	10.26
pmed27	600	10	32	95.41	295.62	201.31	10 800.00	40	25.00
pmed28	600	60	18	180.21	75.00	46.47	10 800.00	20	19.72
pmed29	600	120	13	19.19	7.24	12.28	1976.99	13	
pmed30	600	200	9	1.25	0.31	0.70	5.29	9	
pmed31	700	5	30	19.75	15.70	34.13	10 800.00	NFS	
pmed32	700	10	29	351.59	407.49	949.57	10 800.00	38	32.89
pmed33	700	70	15	177.61	180.43	213.66	10 800.00	17	17.46
pmed34	700	140	11	22.98	6.91	7.59	3159.30	11	
pmed35	800	5	30	18.38	14.53	14.25	10 800.00	NFS	
pmed36	800	10	27	358.30	414.04	227.85	10 800.00	NFS	
pmed37	800	80	15	186.11	268.12	154.61	10 800.00	25	47.64
pmed38	900	5	29	17.64	20.63	19.53	10 800.00	NFS	
pmed39	900	10	23	772.68	1232.82	837.51	10 800.00	NFS	
pmed40	900	90	13	308.41	295.98	167.11	10 800.00	20	40.00
Average				82.25	99.80	155.25	5481.51		

4. Relaxation and heuristic bounds

Before solving P3, if we know that there exists a set $S \subset R$ such that the optimal value of the p-center problem is different from any $\rho_j \in S$, we can effectively use this information: we remove these values from R and drop associated z_j variables from the model, thus, decrease the size of the problem to be solved. For example, if we have a lower bound LB on the optimal objective value, then we may remove any $\rho_j < LB$ from R; similarly, if we have an upper bound UB, then we may remove any $\rho_j > UB$ and solve the model with the restricted R and obtain an optimal solution. In this section, we first analyze the LP relaxation bounds of the four models discussed in this paper. Then, we propose a tighter bound obtained from a partial relaxation of our formulation P3. In addition to the lower bounds that we obtain from relaxations, we propose new lower and upper bounds that can be obtained in a matter of time.

4.1. LP relaxations

Let LP1, LP2, LP3 and LP4 denote the LP relaxations of P1, P2, P3 and P4, respectively and $val(LP1), val(LP2), val(LP3)$ and $val(LP4)$ denote their optimal values. Elloumi et al. [1] showed that the LP bound of P2 is as good as the LP bound of P1. From (20) and (21) we know that there is a one-to-one correspondence between the feasible solutions of P2 and P4. Obviously, this is valid for also the LP relaxations of P2 and P4, that is, for any feasible solution (y, u) of P2, there is a corresponding feasible solution (y, z) of P4 with the same objective value and vice versa. Therefore, the LP bounds of P2 and P4 are equivalent and they are as good as the LP bound of P1. Since any feasible solution to the LP4 is also feasible for the LP3, $val(LP3) \le val(LP4)$. However, the reverse might not be true and we are able to find problems that support otherwise. On the other hand, the LP bounds of P1 and P3 are not comparable.

4.2. Semi-relaxations

Lower bounds based on LP relaxations of the set covering problem are generated for various values of r used in a bisection search by the algorithm of Ilhan and Pinar [24]. Elloumi et al. [1] propose a lower bound LB* which is tighter than the LP relaxation bounds of P1 and P2. LB* is obtained from P2 by relaxing the integrality restrictions on the variables y_j, $j \in M$, while retaining all other constraints of P2. LB* requires solving a mixed integer program, but Elloumi et al. [1] additionally give a method to compute LB* that requires solving a polynomial number of linear programming problems of polynomial size.

We propose now a relaxation bound based on P3 and prove that this bound is equal to the tightest known bound (the bound LB* of Elloumi et al. [1]). Let RP2, RP3 and RP4 be the relaxations that retain the objective function and all constraints of P2, P3 and P4, respectively, except that the constraints $y_j \in \{0,1\}, j \in M$, are replaced with the constraints $y_j \ge 0$, $j \in M$. LB* is exactly defined as the optimal value of RP2. Let $val(RP3)$ and $val(RP4)$ be the optimal values of RP3 and RP4, respectively. The equivalence of LB* and $val(RP4)$ is obvious and can be proven with arguments similar to those in Section 3. Moreover, one can directly see that $val(RP3) \le val(RP4)$ since any (y, z) that satisfies (22) satisfies (15) as well. To prove that $val(RP4) \le val(RP3)$, let us consider an optimal solution $(\bar{y}, \bar{z})$ for RP3 with $val(RP3) = \rho_{k'}$. Then $\bar{z}_{k'} = 1$ and $\bar{z}_k = 0$ for $k \ne k'$. In this case, $\sum_{q=1}^{k} \bar{z}_q = 0$ for $k < k'$ and $\sum_{q=1}^{k} \bar{z}_q = 1$ for $k \ge k'$. Since $\sum_{j \in M} a_{ijk} \bar{y}_j \ge \sum_{j \in M} a_{ijk'} \bar{y}_j \ge 1, \forall i \in N, k > k'$, $(\bar{y}, \bar{z})$ is feasible for RP4 and this implies that $val(RP4) \le val(RP3)$. Thus, we can conclude that $val(RP3) = val(RP4) = \text{LB}^*$.

A direct computation of the proposed lower bound requires solving a mixed integer linear program, RP3, with K binary variables. We now give an alternative method which works in polynomial time. For fixed $h \in T$, add the single constraint $z_h = 1$ to problems P3 and RP3 and call the resulting problems P_h and RP_h, respectively. Let $val(P_h)$ and $val(RP_h)$ be the optimal values of P_h and RP_h, respectively. In case of infeasibility, $val(\cdot)$ is taken to be ∞.

Proposition 1. $val(RP3) = \min_{h \in T} val(RP_h)$.

proof. We have $val(RP3) \le val(RP_h), \forall h \in T$, since RP_h is a restriction of RP3. Now, we show that equality is achieved by some $h \in T$. Let $val(RP3)$ be ρ_a for some $a \in T$. Then there is an optimal solution (y', z') to RP3 such that $z_{k'} = 1$ for $k = a$ and $z_{k'} = 0$ for $k \in T \setminus \{a\}$. This implies that (y', z') is a feasible solution to RP_a and its objective value is ρ_a. Hence, $val(RP_a) \le \rho_a$ due to the feasibility of (y', z') to RP_a. We also have $\rho_a = val(RP3) \le val(RP_a)$ from the first line in the proof. Hence, $val(RP3) = val(RP_a) = \min_{h \in T} val(RP_h)$. □

A closer examination of RP_h shows that it is a linear program in recognition form. To justify this, consider first the problem P_h. Since P_h has all constraints of P3 plus the new constraint $z_h = 1$, constraint (17) and $z_h = 1$ imply that $z_k = 0, \forall k \in T \setminus \{h\}$ in every feasible solution. With $z_h = 1$ and $z_k = 0$ for $k \in T \setminus \{h\}$, constraints (17) and (19) become redundant and can be dropped. Substituting the values of the z-variables in (14) results in a constant objective value of ρ_h. Substituting $z_k = 0$ for $k \in T \setminus \{h\}$ in constraints (15) makes all constraints in (15) redundant except those corresponding to $i \in N$ and $k = h$. It follows that P_h is the following integer program in recognition form: Find $y \in \{0,1\}^m$, if it exists, such that

$$\sum_{j \in M} a_{ijh} y_j \ge 1 \quad \forall i \in N, \tag{23}$$

$$\sum_{j \in M} y_j \le p. \tag{24}$$

RP_h is obtained from P_h by relaxing the binary restriction on y and replacing it with $y_j \ge 0, j \in M$. Hence, RP_h is the following LP in recognition form: Find $y \in \mathbb{R}^m$, if it exists, such that $y \ge 0$ and y satisfies (23) and (24).

To compute $val(RP3)$, it suffices to compute $\min_{h \in T} val(RP_h)$. This is achieved in polynomial time by solving $O(\log_2 K)$ linear programs RP_h for each ρ_h selected from R during a binary search. The binary search is as follows. Let R be the ordered list of $\rho_1 < \rho_2 < \cdots < \rho_K$ as defined before.

Algorithm BINARY. .

Initial: Set $\min = 1, \max = K$ and $LB = \infty$
Main:
(1) Set $mid = \lfloor (\min + \max)/2 \rfloor$.
(2) Solve RP_{mid}.
(3) If RP_{mid} is infeasible, set $\min = mid$; else, set $\max = mid$ and $LB = \rho_{mid}$.
(4) If $\max - \min \ge 1$, go to (1); else, go to Termination.
Termination: Stop with the lower bound $val(RP3) = LB$.

We solve RP2, RP3 and RP4 on 40 p-median instances from the OR-Library with R restricted to $R \cap [LB2, UB2]$. We observe that in 36 of these instances, the bound $val(RP3)$ is equal to the optimal value of P3, referred to as $val(P3)$, while 4 of them have lower bounds with deviations of at most 4.72% from $val(P3)$.

4.3. Attaining quick lower and upper bounds

We propose two upper bounds UB1 and UB2 and two lower bounds LB1 and LB2 to restrict R. We obtain UB1 from the following 2-approximation algorithm for the p-center problem with $p \ge 2$. The algorithm constructs a set $X \subset V$ of centers with $|X| = p$ and allocates each vertex to its closest center. In order to construct X, the two most distant vertices in the network are initially added to the set. While X has less than p elements, the

vertex that is most distant to X is added to the set. After allocating each vertex to the closest center in X, we obtain a feasible solution to the p-center problem. The objective value of this solution is no more than two times the optimal solution value [27]. We refer to this objective value as UB1.

We obtain UB2 by making an improvement on the above algorithm. Let $\{x_1,\ldots,x_p\}$ be the set of centers selected. We divide the network into p clusters $I_1,\ldots,I_p$ where I_j is the set of vertices allocated to center x_i (break ties arbitrarily). Let $T_i=\max\{d(x_i,v_j): j\in I_i\}$ for $i\in\{1,\ldots,p\}$. Then the objective value of the current solution is $T_{max}=\max_{i=1,\ldots,p}T_i$. In order to improve the obtained solution, we solve a 1-center problem in each of these clusters. We start with cluster I_i, where $T_i=T_{max}$ and obtain a new radius value $\overline{T}_i\le T_i$. If $\overline{T}_i=T_i$, we stop the algorithm. If $\overline{T}_i<T_i$, we set $T_{max}=\overline{T}_i$. Then we solve a 1-center problem for each remaining cluster I_k if $T_k>T_{max}$ and set $T_{max}=\overline{T}_k$ if $\overline{T}_k>T_{max}$. Obviously, we will have a solution which is no worse than the initial one after this improvement procedure. Thus, we can conclude that our algorithm is a 2-approximation algorithm. We refer to this solution value as UB2.

LB1 is obtained as follows: Suppose we sort positive distance values in non-decreasing order as $\beta_1\le\beta_2\le\cdots\le\beta_{n\times(n-1)}$ (ties are allowed in the sequence) and let $X=\{v_1,v_2,\ldots,v_p\}\subset V$ be an optimal p-center. Then the remaining $n-p$ vertices need to be served by these centers. Even if we assume that each vertex is served by its closest center, the maximum of the closest distance values cannot be smaller than β_{n-p} since $val(P3)=\max_{i\notin X}\min_{v_j\in X} d_{iv_j}\ge\beta_{n-p}$. We refer to this lower bound as LB1.

LB2 is obtained from UB1: Since UB1 is less than or equal to two times the optimal value, (UB1)/2 provides a lower bound on the optimal value. We improve this lower bound one more step and select the smallest distance value in R, which is greater than or equal to (UB1)/2, as a lower bound (LB2) for the p-center problem.

We compute the LB1, LB2, UB1 and UB2 values for the 40 p-median instances. When we compare the values and calculation times of UB1 and UB2, we see that in 23 of these instances, UB2 value is smaller than UB1 value. This means that the improvement stage is helpful in obtaining a solution with better objective in these instances. In the other instances, the improvement stage is not able to find a solution with a smaller objective value; thus, UB2 value equals UB1 value. When the values of LB1 and LB2 are compared, we observe that LB2 value is larger than LB1 value for each instance. Each of these lower and upper bounds is obtained in at most 0.06 s, which is much faster than the calculation times of any relaxation bound discussed. When we compare LB2 with $val(LP4)$ and $val(RP3)$, we see that LB2 and $val(LP4)$ values are quite close to each other and $val(RP3)$ is greater than both of them for each instance. Since we can obtain LB2 values very quickly, we decide to use LB2 and UB2 values to restrict the set R when we make experiments with the proposed model P3 and double bound algorithms.

5. Double bound algorithms

Let S be any nonempty subset of $T=\{1,\ldots,K\}$ and define $P(S)$ to be the problem which is exactly the same as P3 except that all variables z_k, $k\in T\setminus S$, are dropped from P3. This amounts to replacing the index set T in (14)–(19) with the index set S. Let $val(P(S))$ be the optimal value of $P(S)$ with $P(S)=\infty$ if $P(S)$ is infeasible.

Proposition 2. *Suppose $|S|>2$. Let a and b be the smallest and largest indices in S, respectively.*

(a) *If $val(P(S))=\rho_a$, then $r_p(F)\in\{\rho_1,\ldots,\rho_a\}$.*
(b) *If $val(P(S))=\rho_k$ for some k with $a<k\le b$, then $r_p(F)\in\{\rho_{k'+1},\ldots,\rho_k\}$ where k' is the largest index in S which is smaller than k.*
(c) *If $val(P(S))=\infty$, then $r_p(F)\in\{\rho_{b+1},\ldots,\rho_K\}$.*

Proof. Since S is a subset of T, $P(S)$ is a restriction of P3. Hence, $val(P3)\le val(P(S))$. We have $r_p(F)=val(P3)$ from Proposition 1 implying that $r_p(F)\le\rho_a$ if (a) holds. This proves (a). To prove (b), observe that $val(P(S))=\rho_k$ and $k>a$ imply that P_k is feasible while $P_{k'}$ is infeasible. Accordingly, $\rho_{k'}<r_p(F)\le\rho_k$ which gives $r_p(F)\in\{\rho_{k'+1},\ldots,\rho_k\}$. To prove (c), observe that $val(P(S))=\infty$ implies that P_b is infeasible. Hence, $r_p(F)\in\{\rho_{b+1},\ldots,\rho_K\}$. □

This proposition allows devising a search strategy based on the restrictions of P3. The main idea is this: select a nonempty subset S of T and solve $P(S)$. Depending on which of (a), (b) or (c) occurs in Proposition 2, delete from T the set of indices k such that ρ_k cannot equal $r_p(F)$. Select a new subset S of T after deletions and repeat the procedure. Termination occurs when T reduces to a singleton. The computational success of such a procedure depends on how efficiently we solve each subproblem, $P(S)$, as well as how many times we have to solve a new problem. We give below a specialized way of doing this with sets S containing two elements. The method is called the double bound method. Six variations are discussed.

The double bound algorithm solves $P(S)$ for $S=\{a,b\}$ where $a,b\in T$ with $a<b$. Let $T_1=\{1,\ldots,a\}, T_2=\{a+1,\ldots,b\}$ and $T_3=\{b+1,\ldots,K\}$. We may solve $P(S)$ directly or solve P_a and P_b separately. The former problem involves $m+2$ binary variables while each of the latter problems involves m binary variables. After solving $P(S)$, we know which one of the subsets T_1, T_2 and T_3 contains $r_p(F)$. Since we do not need to consider two of these subsets anymore, we repeat the procedure with the subset that contains the optimal value until we have a single element subset on hand. We propose three ways to choose a and b in Table 2. For each choice, we give two types of algorithms, one type solving $P(S)$ directly (referred to as DB algorithms) and the other solving P_a and P_b separately to obtain a solution to $P(S)$ (referred to as DBR algorithms). The initial steps and the terminations of the algorithms are the same. The algorithms work with the ordered list $R=\{\rho_1,\ldots,\rho_K\}$. We provide a general scheme of the double bound algorithm in Fig. 1.

In the worst case, DB1, DBR1, DB2 and DBR2 algorithms terminate in $O(\log_2 T)$ iterations and DB3, DBR3 algorithms terminate in $O(\log_3 T)$ iterations. Suppose we are at the beginning of some iteration of our DB2 algorithm and we have an a value and a max value. Then we solve $P(S)$ for $S=\{a,\max-1\}$. If $r_p(F)$ equals $\rho_{\max}$ of this iteration, then $P(S)$ becomes infeasible and we terminate the algorithm at the end of this iteration. This quick termination is also available in DBR2 algorithm. In DB1 algorithm, we solve $P(S)$ for $S=\{a,\max\}$. If $r_p(F)$ equals $\rho_{\max}$ of the current iteration, we cannot know this until we solve $P(S)$ for $S=\{\max-1,\ \max\}$ and this takes $O(\log_2(\max-a))$ iterations. This is also valid for DBR1 algorithm. In the same case, DB3 and DBR3 algorithms terminate after $O(\log_3(\max-b))$ iterations.

6. Computational experiments

In our computational experiments, we follow the general trends in the literature and take $F=V$ with unit weights for

Table 2
Selection of radius values.

(a,b)	Together	Separately
$a=\lfloor(\max+\min)/2\rfloor$ $b=\max$	DB1	DBR1
$a=\lfloor(\max+\min)/2\rfloor$ $b=\max-1$	DB2	DBR2
$a=\min+\lfloor(\max-\min)/3\rfloor$ $b=\min+2\lfloor(\max-\min)/3\rfloor$	DB3	DBR3

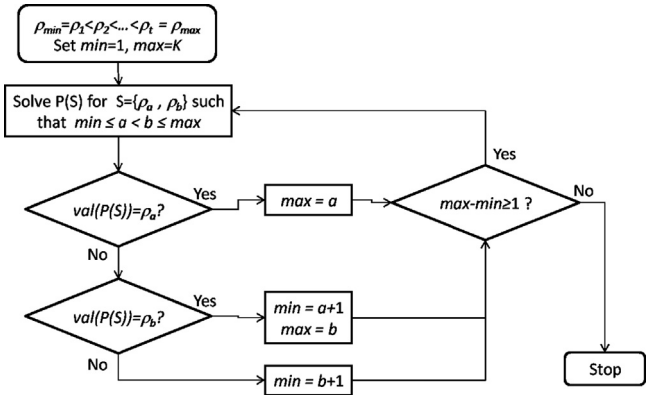

Fig. 1. General scheme of the double bound algorithm.

vertices. The input data used for the computations consists of the p-median data from OR-Library for 40 instances with n varying between 100 and 900 and p varying between 5 and $(n/3)$ and some instances from TSPLIB. The original data in OR-Library consists of a listing of edges and their lengths. By using the all-pair shortest path algorithm due to Floyd [28] on this data, we obtain the distance matrix $D=[d_{ij}]$. In TSPLIB instances, the coordinates of vertices are provided. We calculate the Euclidean distance for each vertex pair and round it to the nearest integer. We solve problems from u1817 with $n=1817$, from d15112 with $n=2500$ and from pcb3038 with $n=3038$. During our computations we use JCreator LE 4.50 [29] and CPLEX 12.4 concert technology. In all mathematical models that we solve, we set MIPEmphasis option of CPLEX to 1. For the algorithms DBR1, DBR2 and DBR3, we additionally set IntSolLim to 1.

In all tables in this section, the first three columns give the characteristics of the instances and the column labeled "Opt" gives the optimal p-radii. The reported solution times in tables are in seconds.

The experiments that we conducted on the problems with large number of nodes revealed that the DB algorithms require significantly more time to arrive at optimal solutions than DBR algorithms and among the DBR algorithms, DBR2 algorithm performed slightly better than the other algorithms. For this reason, we give the results only for DBR2 algorithm in this paper. One can see the detailed results of the other algorithms in [30].

We give the solution times of DBR2 algorithm on 40 p-median instances in Table 3. In these experiments, we continue to restrict R to $R\cap[LB2, UB2]$ and give LB2 and UB2 values in the fifth and sixth columns, respectively. The last column presents the time required by DBR2 algorithm. The first observation from this table is that the DBR2 performs much better than solving P3 or P4 directly. The solution times of DBR2 algorithm are much better for all instances than direct solution times. The average solution time of DBR2 algorithm is also much better than solving P3 or P4 directly.

Table 4 gives the solution times for DBR2 algorithm on 20 instances from TSP-Library with $n=1817$. We again restrict R by using LB2 and UB2 values in these experiments and these values are given in the fifth and sixth columns, respectively. The solution time of the algorithm is given in the last column. For the bold instances in this table, the optimal values are provided for the first time in the literature. Elloumi et al. [1] conduct experiments on the first 15 instances of this table, but they are not able to solve optimally the bold instances. The algorithm spends an excessive amount of time for solving the problem with $p=90$. On the average, DBR2 spends 563.74 s to solve these instances.

We solve larger problems with $n\in\{1817, 2500, 3008\}$ from the TSPLIB by using the lower bound $val(RP3)$. We compute this bound by solving $O(\log_2 K)$ LP problems RP_h using the algorithm BINARY

Table 3
Times (s) required to solve P3 with DBR2 algorithm on 40 p-median instances.

Instance	n	p	Opt	LB2	UB2	Time
pmed1	100	5	127	101	191	0.54
pmed2	100	10	98	83	155	0.23
pmed3	100	10	93	73	143	0.21
pmed4	100	20	74	56	92	0.13
pmed5	100	33	48	38	75	0.12
pmed6	200	5	84	69	107	0.32
pmed7	200	10	64	51	102	0.18
pmed8	200	20	55	42	83	0.19
pmed9	200	40	37	27	53	0.20
pmed10	200	67	20	16	31	0.10
pmed11	300	5	59	49	69	0.54
pmed12	300	10	51	46	76	0.32
pmed13	300	30	36	30	55	0.30
pmed14	300	60	26	20	39	0.15
pmed15	300	100	18	12	24	0.07
pmed16	400	5	47	43	56	0.55
pmed17	400	10	39	35	57	0.33
pmed18	400	40	28	23	46	0.17
pmed19	400	80	18	15	27	0.11
pmed20	400	133	13	9	18	0.05
pmed21	500	5	40	36	49	0.95
pmed22	500	10	38	32	54	1.08
pmed23	500	50	22	18	35	0.31
pmed24	500	100	15	12	23	0.14
pmed25	500	167	11	8	15	0.07
pmed26	600	5	38	33	52	1.61
pmed27	600	10	32	29	46	0.94
pmed28	600	60	18	15	29	0.29
pmed29	600	120	13	11	21	0.18
pmed30	600	200	9	7	13	0.08
pmed31	700	5	30	27	36	2.03
pmed32	700	10	29	25	38	2.35
pmed33	700	70	15	13	25	0.33
pmed34	700	140	11	9	17	0.15
pmed35	800	5	30	27	34	1.69
pmed36	800	10	27	25	36	2.71
pmed37	800	80	15	13	26	0.40
pmed38	900	5	29	26	32	2.55
pmed39	900	10	23	20	30	2.59
pmed40	900	90	13	11	21	0.61
Average						0.65

with R restricted to $R\cap[LB2, UB2]$. The bounds $val(RP3)$ are generally computed in a matter of seconds. The minimum, average and maximum computing times are 0.59, 33.16 and 166 s, respectively. We also note that the lower bound $val(RP3)$ is equal to the optimal value of P3 for 8 instances of the TSPLIB problems with $n\in\{1817, 2500, 3008\}$ while the gap between them is at most 0.053 for the remaining instances.

Table 5 gives the relevant statistics for solving P3 via the algorithm DBR2 by restricting R to $R\cap[val(RP3), UB2]$. For the 33 instances reported in Table 5, a time limit of 3 h is imposed for each problem $P_h\in\{P_{mid}, P_{\max-1}\}$ attempted in main steps of the algorithm DBR2. At the end of 3 h, either a solution is found for P_h or the infeasibility of P_h is detected or the feasibility/infeasibility status of P_h remains unknown. If the status of P_h remains unknown, we solve the problems P_{h+1}, P_{h+2}, etc. with successively increasing radius values until a feasible solution is found within 3 h for some problem P_{h+k}. This gives us an upper bound ρ_{h+k} on $r_p(F)$ that is the best (smallest) bound that can be confirmed within the time limit. Similarly, we solve problems P_{h-1}, P_{h-2}, etc. with successively decreasing radius values until we detect infeasibility of some $P_{h-k'}$ within the time limit. This gives us a lower bound $\rho_{h-k'+1}$ on $r_p(F)$ that is the largest possible that can be confirmed within the time limit. Thus, $r_p(F)\in\{\rho_{h-k'+1}, \ldots, \rho_{h+k}\}$. The values $\rho_{h-k'+1}$ and ρ_{h+k} are reported as the "Best LB" and "Best UB" in the table in columns 5 and 6, respectively. Column 7 gives the gap between the "Best UB"

Table 4
Results of algorithm DBR2 for solving P3 for TSPLIB instances with $n=1817$.

Instance	n	p	Opt	LB2	UB2	Time (s)
u1817	1817	10	458	301	585	22.11
u1817	1817	20	**309**	191	357	278.49
u1817	1817	30	**241**	166	331	344.59
u1817	1817	40	**209**	155	307	1221.86
u1817	1817	50	**185**	127	229	330.09
u1817	1817	60	163	105	209	17.52
u1817	1817	70	148	99	194	6.04
u1817	1817	80	137	93	185	35.12
u1817	1817	90	**129**	90	180	7519.04
u1817	1817	100	127	86	170	10.22
u1817	1817	110	110	81	161	5.32
u1817	1817	120	107	74	148	3.99
u1817	1817	130	**105**	71	137	335.03
u1817	1817	140	**102**	68	129	4.62
u1817	1817	150	**92**	64	127	6.61
u1817	1817	200	**80**	56	105	2.56
u1817	1817	250	**76**	46	92	2.17
u1817	1817	300	**63**	46	80	2.01
Average						563.74

and “Best LB” values while column 4 gives the optimal values for P3. If the optimal solution is found for a problem within 3 h, the values in columns 4–6 are equal. For such problems, the gaps in column 7 are zero. Column 8 gives the solution times obtained by the algorithm DBR2. For all of the instances with $n=1817$, we obtain the optimal solution. The most time consuming instance among them is the one with $p=40$. This instance is solved in 1225 s. For the problems with $n=2500$, we obtain the optimal solution for all instances except the one with $p=100$. The gap between the best upper bound and the best lower bound we obtain for this instance is 0.008. The optimal solutions for the six instances with $n=3038$ are obtained. For the remaining instances the gap between the best bounds is at most 0.027. For any of the reported network sizes, all instances with $p=5$ and 10 are solved in less than 3 min and all instances with $p=400$ and 500 are solved in less than 12 s.

Before solving the individual problem P_k for some $k\in T$ in the main steps of DBR2 algorithm, it is possible to eliminate some of the variables and constraints by applying the reduction rules which are well known for the set covering problem [31]. These rules not only detect any infeasibility immediately, but also reveal the optimal value of some decision variables and the dominance relation between the constraints. We utilized these reduction rules in our algorithm to solve the instances that require large amount of time (more than 1 h) and provided the results in Table 6. One of the instances in this table is taken from Table 4 and the others are taken from Table 5. Table 6 consists of the same columns as in Table 5 except that it has an additional column “PP Time” which represents the total time consumed for the reduction procedure. Among the 13 instances in this table, for 11 of them reduction makes an improvement in terms of either the best LB, the best UB or in the total time and those improved parameters are depicted in bold. We observe that the maximum gap between the best LB and the best UB decreases from 0.027 to 0.022 and we are able to obtain the optimal solution for the instance with $n=3038$ and $p=30$. For some instances the total time consumption decreases significantly. For instance, the total time consumed decreases around 85% for the first instance and 84% for the second instance in Table 6. The average decrease in total time is 61% for the instances with decrease in time and 38% for all instances. When we subtract the reduction time from the total time, we can observe how much reduction helps CPLEX to solve the problems. The average of this improvement is 60% for the improved instances and 49% for all instances. These experiments reveal that the reduction rules are quite helpful in solving both individual problems P_k for $k\in T$ and P3 via DBR2 algorithm.

Table 5
Results of algorithm DBR2 for solving problem P3 for TSPLIB instances with $n\in\{1817, 2500, 3038\}$.

Instance	n	p	$val(P3)$	Best LB	Best UB	Gap	Time (s)
u1817	1817	5	715	715	715	0	14.10
u1817	1817	10	458	458	458	0	20.31
u1817	1817	20	309	309	309	0	257.68
u1817	1817	30	241	241	241	0	204.77
u1817	1817	40	209	209	209	0	1225.00
u1817	1817	50	185	185	185	0	314.46
u1817	1817	100	127	127	127	0	7.43
u1817	1817	200	80	80	80	0	2.57
u1817	1817	300	63	63	63	0	1.19
u1817	1817	400	51	51	51	0	0.84
u1817	1817	500	51	51	51	0	0.23
d15112	2500	5	5856	5856	5856	0	95.91
d15112	2500	10	3705	3705	3705	0	84.85
d15112	2500	20	2573	2573	2573	0	288.12
d15112	2500	30	2029	2029	2029	0	5293.59
d15112	2500	40	1723	1723	1723	0	21 899.83
d15112	2500	50	1524	1524	1524	0	5782.76
d15112	2500	100		1049	1057	0.008	94 245.98
d15112	2500	200	723	723	723	0	28 371.08
d15112	2500	300	571	571	571	0	38.39
d15112	2500	400	481	481	481	0	4.99
d15112	2500	500	424	424	424	0	4.84
pcb3038	3038	5	1064	1064	1064	0	116.74
pcb3038	3038	10	729	729	729	0	176.28
pcb3038	3038	20	493	493	493	0	22 740.76
pcb3038	3038	30		391	397	0.015	76 923.67
pcb3038	3038	40		331	337	0.018	72 364.56
pcb3038	3038	50		292	300	0.027	91 029.77
pcb3038	3038	100		207	209	0.010	27 292.39
pcb3038	3038	200		138	141	0.022	33 929.91
pcb3038	3038	300	115	115	115	0	6185.96
pcb3038	3038	400	97	97	97	0	11.48
pcb3038	3038	500	85	85	85	0	5.23

Table 6
Results with utilization of the reduction rules.

Instance	n	p	val ($P3$)	Best LB	Best UB	Gap	PP time (s)	Total time (s)
u1817	1817	90	129	129	129	0	578.59	**1225.76**
d15112	2500	30	2029	2029	2029	0	108.25	**865.63**
d15112	2500	40	1723	1723	1723	0	132.37	**6831.56**
d15112	2500	50	1524	1524	1524	0	158.75	**1645.55**
d15112	2500	100		**1050**	1059	0.009	416.46	104 013.04
d15112	2500	200	723	723	723	0	482.84	**5293.26**
pcb3038	3038	20	493	493	493	0	3102.33	**6800.68**
pcb3038	3038	30	**393**	**393**	**393**	**0**	4063.66	**52 285.79**
pcb3038	3038	40		**332**	337	**0.015**	4920.76	75 702.72
pcb3038	3038	50		**293**	**299**	**0.020**	6199.33	**80 313.04**
pcb3038	3038	100		207	**208**	**0.005**	4483.28	**15 717.50**
pcb3038	3038	200		138	141	0.022	6060.09	42 621.69
pcb3038	3038	300	115	115	115	0	3701.42	6569.00

7. Conclusion

In this paper, we proposed a new IP formulation for the absolute and vertex restricted p-center problems. By tightening one set of constraints in our model, we obtained a modified model with much larger LP bounds. When compared with the two models from the literature, both of our models performed better in terms of the time requirement. The LP bounds of our tightened

model are equivalent to the LP bounds of the model proposed by [1] and they are the strongest LP bounds among four models. We obtained a stronger lower bound from our models by relaxing one set of binary variables and this lower bound is equivalent to the best known lower bound in the literature. We also provided a polynomial time algorithm to obtain this lower bound. In addition to the relaxation bounds, we proposed new lower and upper bounds which can be computed very quickly and we used them effectively in our models and algorithms. We proposed a new method that solves successive restrictions of our model and we computationally tested a specialization of this algorithm, referred to as double bound algorithm, using benchmark problems from OR-Library and TSPLIB. We were able to solve problems with $n=2500$ and 3038 from the TSPLIB using this algorithm while the largest problem solved in the literature had 1817 nodes. We solved the problems that require large amount of time by integrating the reduction rules to our algorithm. We observed significant improvements in utilization of the reduction rules.

Acknowledgments

This research is supported by Grant 111M520 of Program 1001 of TÜBİTAK, The Scientific and Technological Research Council of Turkey. The authors thank two anonymous referees for their valuable comments. They also thank Bahar Y. Kara and Oya E. Karasan for their helpful reading and comments.

References

[1] Elloumi S, Labbé M, Pochet Y. A new formulation and resolution method for the *p*-center problem. INFORMS Journal on Computing 2004;16(1):84–94.
[2] Beasley, JE. OR-LIBRARY. June 2012. URL ⟨http://people.brunel.ac.uk/~mastjjb/jeb/info.html⟩.
[3] Reinelt G. TSPLIB—a traveling salesman problem library. ORSA Journal on Computing 1991;3(4):376–84.
[4] Dearing P, Francis R, Lowe T. Convex location problems on tree networks. Operations Research 1976;24(4):628–42.
[5] Hakimi SL. Optimum locations of switching centers and the absolute centers and medians of a graph. Operations Research 1964;12(3):450–9.
[6] Minieka E. The *m*-center problem. SIAM Review 1970;12(1):138–9.
[7] Kariv O, Hakimi SL. An algorithmic approach to network location problems. Part I: the *p*-centers. SIAM Journal on Applied Mathematics 1979;37(3):513–38.
[8] Hooker J, Garfinkel R, Chen C. Finite dominating sets for network location problems. Operations Research 1991;39(1):100–18.
[9] Megiddo N, Tamir A, Zemel E, Chandrasekaran R. An $O(n \log^2 n)$ algorithm for the *k*-th longest path in a tree with applications to location problems. SIAM Journal on Computing 1981;10(2):328–37.
[10] Tansel B, Francis R, Lowe T, Chen M. Duality and distance constraints for the nonlinear *p*-center problem and covering problem on a tree network. Operations Research 1982;30(4):725–44.
[11] Megiddo N, Tamir A. New results on the complexity of *p*-centre problems. SIAM Journal on Computing 1983;12(4):751–8.
[12] Jaeger M, Kariv O. Algorithms for finding *p*-centers on a weighted tree (for relatively small *p*). Networks 1985;15(3):381–9.
[13] Shaw DX. A unified limited column generation approach for facility location problems on trees. Annals of Operations Research 1999;87:363–82.
[14] Tansel BC, Francis RL, Lowe TJ. State of the art—location on networks: a survey. Part I: the *p*-center and *p*-median problems. Management Science 1983;29(4):482–97.
[15] Tansel BC, Francis RL, Lowe TJ. State of the art—location on networks: a survey. Part II: exploiting tree network structure. Management Science 1983;29(4):498–511.
[16] Handler GY, Mirchandani PB. Location on networkstheory and algorithms. Cambridge, MA: MIT Press; 1979.
[17] Daskin MS. Network and discrete location: models, algorithms, and applications. New York: Wiley; 1995.
[18] Tansel BC. Discrete center problems. In: Eiselt HA, Marianov V, editors. Foundations of location analysis. New York: Springer; 2011. p. 79–106 [Chapter 5].
[19] Cornuéjols G, Nemhauser GL, Wolsey LA. A canonical representation of simple plant location problems and its applications. SIAM Journal on Algebraic Discrete Methods 1980;1(3):261–72.
[20] Christofides N, Viola P. The optimum location of multi-centres on a graph. Operational Research Quarterly 1971:145–54.
[21] Toregas C, Swain R, ReVelle C, Bergman L. The location of emergency service facilities. Operations Research 1971;19(6):1363–73.
[22] Garfinkel R, Neebe A, Rao M. The *m*-center problem: minimax facility location. Management Science 1977;23(10):1133–42.
[23] Daskin MS. A new approach to solving the vertex *p*-center problem to optimality: algorithm and computational results. Communications of the Operations Research Society of Japan 2000;45(9):428–36.
[24] Ilhan T, Pinar M. An efficient exact algorithm for the vertex *p*-center problem. 2001. URL ⟨http://www.ie.bilkent.edu.tr/~mustafap/pubs/⟩.
[25] khedhairi Al-A, Salhi S. Enhancements to two exact algorithms for solving the vertex *p*-center problem. Journal of Mathematical Modelling and Algorithms 2005;4(2):129–47.
[26] IBM. IBM ILOG CPLEX Optimization Studio. 2013 URL ⟨www.ibm.com/software/products/us/en/ibmilogcpleoptistud/⟩.
[27] González TF. Clustering to minimize the maximum intercluster distance. Theoretical Computer Science 1985;38:293–306.
[28] Floyd RW. Algorithm 97: shortest path. Communications of ACM 1962;5(6):345–6.
[29] XINOX Software. JCreator—Java IDE. 2012. URL ⟨http://www.jcreator.com/⟩.
[30] Calik H. New formulations and exact solution methods for the capacitated and uncapacitated *p*-center location problem. PhD dissertation. Ankara, Turkey: Department of Industrial Engineering, Bilkent University; to appear.
[31] Francis RL, Leon F, McGinnis J, White JA. Facility layout and location: an analytical approach. Upper Saddle River, NJ: Prentice Hall; 1992.