Journal of Mathematical Modelling and Algorithms (2005) 4: 129–147
DOI: 10.1007/s10852-004-4072-3

Enhancements to Two Exact Algorithms for Solving the Vertex P-Center Problem

A. AL-KHEDHAIRI and S. SALHI
Management Mathematics Group, The University of Birmingham, Birmingham, B15 2TT, UK

(Received: 13 October 2003; in final form: 7 October 2004)

Abstract. Enhancements to two exact algorithms from the literature to solve the vertex P-center problem are proposed. In the first approach modifications of some steps are introduced to reduce the number of ILP iterations needed to find the optimal solution. In the second approach a simple enhancement which uses tighter initial lower and upper bounds, and a more appropriate binary search method are proposed to reduce the number of subproblems to be solved. These ideas are tested on two well known sets of problems from the literature (i.e., OR-Lib and TSP-Lib problems) with encouraging results.

Mathematics Subject Classifications (2000): 90B10, 90B80, 90C09, 90C47.

Key words: P-center, location, 0–1 programming.

1. Introduction

The P-center problem (also known as the minimax problem) is to locate P facilities and assign clients to them so as to minimise the maximum distance between a client and the facility to which it is assigned. This well known location problem, which was first introduced by Hakimi [7, 8], has several applications including the location of emergency facilities such as ambulance stations and firehouses. There are several possible variations of the basic model. If facility locations are restricted to the nodes of the network, the problem is referred to as a *vertex center problem.* Center problems which allow facilities to be located anywhere on the network are known as *absolute center problems.* Both versions can be either *weighted* or *unweighted.* In the weighted problem, the distances between demand nodes and facilities are multiplied by a weight usually associated with the demand node. As an example, the weight might represent the population or the importance of a node. In the unweighted problem, all demand nodes are treated equally.

To formulate the vertex P-center problem, we define:

I = set of demand nodes, $I = \{1, \ldots, N\}$,
J = set of candidate facility sites, $J = \{1, \ldots M\}$,
d_{ij} = distance between demand node $i \in I$ and candidate site $j \in J$,
P = number of facilities to be located,

$$w_j = \begin{cases} 1 & \text{if a facility is located at candidate site } j \in J, \\ 0 & \text{otherwise,} \end{cases}$$

$$Y_{ij} = \begin{cases} 1 & \text{if demand node } i \in I \text{ is assigned to an open facility at candidate site } j \in J, \\ 0 & \text{otherwise,} \end{cases}$$

$D =$ maximum distance (or time) between a demand node and the nearest facility (D is also referred to as the covering distance or time).

The binary linear programming formulation of the vertex P-center problem is as follows:

$$\text{Minimise} \quad D \tag{1}$$

subject to:

$$\sum_{j \in J} Y_{ij} = 1 \quad \forall i \in I, \tag{2}$$

$$Y_{ij} \leqslant w_j \quad \forall i \in I, j \in J, \tag{3}$$

$$\sum_{j \in J} w_j = P, \tag{4}$$

$$D \geqslant \sum_{j \in J} d_{ij} Y_{ij} \quad \forall i \in I, \tag{5}$$

$$w_j, Y_{ij} \in \{0, 1\} \quad \forall i \in I, j \in J. \tag{6}$$

The objective function (1) minimises the maximum distance between each demand node and its closest open facility. Constraint (2) ensures that each demand node is assigned to exactly one facility, while constraints (3) restrict demand nodes to be assigned to open facilities. Constraint (4) stipulates that P facilities are to be located. Constraints (5) define the maximum distance between any demand node i and the nearest facility at node j. Finally, constraints (6) refer to integrality constraints.

For fixed values of P, the vertex P-center problem can be solved in polynomial time. This can be done by evaluating each of the $O(N^p)$ possible combinations of P facility sites. Evaluating each of these can be done in polynomial time [3] though it may take a considerable amount of CPU time. For variable values of P, the P-center problem is NP-hard [10].

Different authors have used an auxiliary problem (e.g., the Set Covering Problem; SCP) to solve the P-center problem optimally. The objective of the SCP is to find the minimum number of facilities and their locations so that each demand point has to be served by a facility within a specified maximum response time (or distance) which can be referred to as radius. The solution to this problem can be easily found by solving its linear programming relaxation with occasional branch and bound applications. However, for large problems the size of the relaxed version

of the SCP can be reduced using successive row and column reductions (see [3] for more discussion of such reduction rules). This idea is to find the smallest radius such that the optimal solution of the considered auxiliary problem yields a feasible solution to the P-center problem. Initially, Minieka [11] suggested a rudimentary algorithm that relies on solving a finite sequence of Set Covering Problems. The idea is to choose a threshold distance (covering distance) as radius and to check whether all demand points (customers) are covered within this radius using no more than P facilities. Based on Minieka's idea, Daskin [3] developed an algorithm to solve this problem optimally using the bisection method that systematically reduces the gap between the upper and lower bounds of the optimal solution and hence finds the optimal covering distance. Also, Daskin [4] and Elloumi, Labbe and Pochet [5] proposed two efficient and exact algorithms for the vertex P-center problem. Both algorithms solve successive subproblems and rely on carrying out an iterative search over coverage distances rather than sets of facility locations. Daskin has formulated a maximum set covering subproblem and solved it by Lagrangian relaxation, whereas Elloumi et al. used Minieka's subproblems and solved it by a greedy heuristic and the IP formulation of the subproblem. Ilhan and Pinar [9] proposed an interesting exact solution method for the vertex P-center problem. Their method is composed of two phases. The first phase, called the LP-Phase, computes a lower bound to the optimal solution of the problem by solving a series of feasibility problems based on a LP formulation. The second phase, referred to as the IP-Phase, uses also feasibility problems to check whether or not it is possible to serve all customers with no more than P facilities within a given radius.

In this study we introduce some modifications to the algorithms of Ilhan and Pinar [9] and Daskin [3]. As these two algorithms are used as a basis in this work, their respective formulations will be given in the next sections.

The remainder of this paper is organized as follows. In Section 2 we present the algorithm of Ilhan and Pinar [9] and propose our modifications. In Section 3 the algorithm of Daskin [3] is given followed by our enhancement scheme. Our computational results for each modification are provided in Section 4 and Section 5 summarises our findings and points out some research issues.

2. Ilhan and Pinar [9] Algorithm

Ilhan and Pinar developed an interesting 2-phase approach to solve the vertex P-center problem optimally using the following feasibility formulation for a specific covering radius, R. They refer to the problem as IP:

$$\sum_{j \in J} b_{ij} w_j \geqslant 1 \quad \forall i \in I, \tag{7}$$

$$\sum_{j \in J} w_j \leqslant P, \tag{8}$$

$$w_j \in \{0, 1\} \quad \forall j \in J, \tag{9}$$

where I, J and w_j, $j \in J$, are defined as in Section 1, and b_{ij} is calculated as follows:

$$b_{ij} = \begin{cases} 1 & \text{if } d_{ij} \leqslant R, \forall i \in I,\ j \in J, \\ 0 & \text{otherwise.} \end{cases}$$

In the first phase, the LP relaxation for the above problem is solved for a given R. Note that LP feasibility problem which is used in the first phase is similar to the IP feasibility problem, except that the integrality constraints (9) are relaxed and replaced by:

$$0 \leqslant w_j \leqslant 1 \quad \forall j \in J. \tag{10}$$

Observation. Ilhan and Pinar noted a crucial and useful observation that has led to the success of their algorithm. For instance, if the solution is not feasible for the relaxed IP, then obviously there is no need to solve the IP problem. The smallest value of R for which the relaxed IP is feasible is then used as the starting covering radius (say R_0), in the second phase. Starting from this lower bound (R_0), a series of IP feasibility problems with different R values for the radius are solved ($R \geqslant R_0$). As R is integer the optimal solution will be reached in a limited number of iterations.

In other words, if the solution is infeasible for a specific radius R, the value of R needs to be increased until the corresponding (IP) becomes feasible. According to the algorithm (see Step 7 in Figure 1) the value of R is increased to the next minimum distance that is greater than R. This process continues until feasibility is reached and hence the current value of R represents the optimal solution of the vertex P-center problem. The full algorithm is given in Figure 1 where Phase 1 consists of the first 5 steps and Phase 2 represents Steps 6 and 7.

2.1. TWO MODIFICATIONS OF THE ALGORITHM

We propose two changes in the algorithm, the first one is based on Step 5 and the other uses Steps 6 and 7 (in Phase 2 of the algorithm).

Modification of Step 5

In Step 5, the resetting of the radius (R) to the lower bound if the LP formulation is feasible does not, in our views, takes advantage of the usefulness of the observation cited earlier, because we know already that the LP formulation at this value of lower bound is infeasible (from Step 3). In other words, we can guarantee that the IP formulation (Step 7) will be systematically infeasible if its corresponding LP one is found infeasible in Step 3, and hence such an implementation will always require one wasted ILP iteration. To overcome this drawback we reset R to the upper bound if the LP formulation is infeasible (as given in the algorithm) and keep R unchanged otherwise. Our new Step 5 can be written as follows:

–**Phase1** (Solving the LP relaxation)–

Step 1. Set $L = \min\{d_{ij} : \forall i \in I,\ j \in J\}$ and
$U = \max\{d_{ij} : \forall i \in I,\ j \in J\}$.

Step 2. Calculate $R = \left\lfloor \dfrac{(U + L)}{2} \right\rfloor$
If $d_{ij} \leqslant R$ then set $b_{ij} = 1$, else set $b_{ij} = 0$.

Step 3. Solve the LP relaxation of (IP)
If the LP relaxation is not feasible then set $L = R$, else set $U = R$.

Step 4. Calculate $(U - L)$.
If $(U - L) \leqslant 1$ then go to Step 5, else go to Step 2.

Step 5. If the LP relaxation is not feasible then set $R = U$, else set $R = L$.

–**Phase2** (Solving the LP problem)–

Step 6. If $d_{ij} \leqslant R$ then set $b_{ij} = 1$, else set $b_{ij} = 0$.

Step 7. Solve the IP problem.
If the IP problem is feasible then stop (R is the optimal solution), else set $R = \min\{d_{ij} : d_{ij} > R,\ \forall i \in I,\ j \in J\}$ and go to Step 6.

Figure 1. The original algorithm of Ilhan and Pinar [9].

Step 5 (new). If the LP relaxation is not feasible then set $R = U$, else keep the old value of R from Step 2.

Such a scenario happened several times in our computational results as 55 out of the 83 instances were affected. Thus, the number of ILP iterations for these 55 instances is reduced by one iteration. The computational results of this modification are given in Section 4 along with other results.

Modification of Phase 2

(a) Existence of the Value of R

The second enhancement is based on the existence of the value of the radius (R) produced by phase1 in the distance matrix (i.e., there is no client or customer at that distance). It can easily be shown that if R does not exist in the matrix the IP formulation will be infeasible at this coverage radius, R, and hence one additional IP iteration will be performed for no reason. This weakness can be dealt with by checking R before performing Step 6. If R exists in the distance matrix then we continue as before, otherwise we reset R to the next minimum distance greater than the current value of R and continue. Step 6 can be replaced with the following two ministeps.

Step 6.1. Check if R exists in the distance matrix or not;
If R exists then go to Step 6.2, else set $R = \min\{d_{ij} : d_{ij} < R,\ \forall i \in I,\ j \in J\}$ and go to Step 6.2.

Step 6.2. If $d_{ij} \leqslant R$ then set $b_{ij} = 1$, else set $b_{ij} = 0$.

(b) A Jump-Based Scheme

In the second phase of the original algorithm (Step 7), if the IP feasibility problem for the covering radius (R) is not feasible, the new covering radius (R) is set to the minimum of all distances which is greater than the current value of the covering radius:

$$\text{new } R = R' = \min\{d_{ij} : d_{ij} > R,\ \forall i \in I,\ j \in J\}. \tag{11}$$

Our view is that we can extend such an update in the following way. Instead of choosing the next minimum, we select the second next minimum. Note that the calculations of these minima exclude ties. This can be written mathematically as follows:

$$R = R'', \quad \text{where :}$$
$$R'' = \min\{d_{ij} : d_{ij} > R',\ \forall i \in I,\ j \in J\} \quad \text{and}$$
$$R' = \min\{d_{ij} : d_{ij} > R, \forall i \in I,\ j \in J\}.$$

If the solution of the IP is not feasible we repeat this procedure by setting R to the second next minimum distance greater than the current value until we reach feasibility. As there is no guarantee that the solution for the IP using R' is infeasible, an additional computation to check whether there is a feasible solution for R' obviously needs to be carried out.

If the problem is infeasible for R' then we can conclude that the covering radius $R = R''$ is the optimal solution, otherwise we can also guarantee that $R = R'$ is the optimal solution. To apply this jump-based scheme, we modified the second phase of Ilhan and Pinar's [9] algorithm as shown in Figure 2.

Illustration of the Jump-Based Update

Three cases of using this modification are illustrated in the following example as shown in Figure 3. In these three examples, known as A, B and C, we assume that the original algorithm does not stop at the covering radius (R) found by the first phase of the algorithm (i.e., it does not stop at Step 7 in Figure 2) but seeks for feasibility of the IP problem. Note that the top arrows refer to the modified algorithm and the bottom arrows refer to the original algorithm. For convenience, R in Figure 3 refers to the best lower bound found by the first phase of the algorithm.

In Figure 3 (part A), the original algorithm and its modified version first check the feasibility of IP problem for the best lower bound (R) found by the first phase

Step 6.1. Check if R exists in the distance matrix or not;
If R exists then go to Step 6.2, else set $R = \min\{d_{ij} : d_{ij} < R,\ \forall i \in I,\ j \in J\}$ and go to Step 6.2.
Step 6.2. If $d_{ij} \leqslant R$ then set $b_{ij} = 1$, else set $b_{ij} = 0$.
Step 7. Solve the IP problem.
If the IP problem is feasible then stop (R is the optimal solution), else increase the value of R as follows:
$R' = \min\{d_{ij} : d_{ij} > R,\ \forall i \in I,\ j \in J\}$ and
$R'' = \min\{d_{ij} : d_{ij} > R',\ \forall i \in I,\ j \in J\}$.
Set $R = R''$ and go to Step 8.
Step 8. If $d_{ij} \leqslant R$ then set $b_{ij} = 1$ else set $b_{ij} = 0$.
Step 9. Solve the IP problem.
Step 9.1. If the IP problem is infeasible then increase the value of R as follows:
$R' = \min\{d_{ij} : d_{ij} > R,\ \forall i \in I,\ j \in J\}$ and
$R'' = \min\{d_{ij} : d_{ij} > R',\ \forall i \in I,\ j \in J\}$.
Set $R = R''$ and go to Step 8.
Step 9.2. Else (the IP problem is feasible).
Set $R = R'$, and solve the IP problem when $R = R'$:
If the IP problem is feasible then stop ($R = R'$ is the optimal solution), else set $R = R''$ as the optimal solution and stop.

Figure 2. Modified second phase of Ilhan and Pinar algorithm.

which happens to be not feasible (iteration 1). However, the original algorithm jumps to the next minimum distance greater than R which is here R_1 (bottom arrows) and tests the feasibility of IP problem for $R = R_1$, and finds no feasible solution again (iteration 2). This check is performed until $R = R_6$ where a feasible solution for the IP problem is found (iteration 7). The modified algorithm jumps to the second minimum distance greater than R which is here R_2 (top arrows), then to R_4, and R_6 where IP problem is feasible (iterations 3 and 4). Before concluding that R_6 is the optimal solution, an additional check at R_5 (iteration 5) is performed which confirmed infeasibility at this level. In conclusion, the original algorithm performs 7 iterations to reach the optimal solution whereas the modified algorithm performs 5 iterations only.

In part B of Figure 3, the IP problem is not feasible for R (iteration 1). The original algorithm jumps to R_1, R_2, then R_3 (iterations 2, 3 and 4) searching for a feasible covering radius for the IP problem, where a feasible solution is found at R_3, then the algorithm stops. The modified algorithm jumps to the second minimum

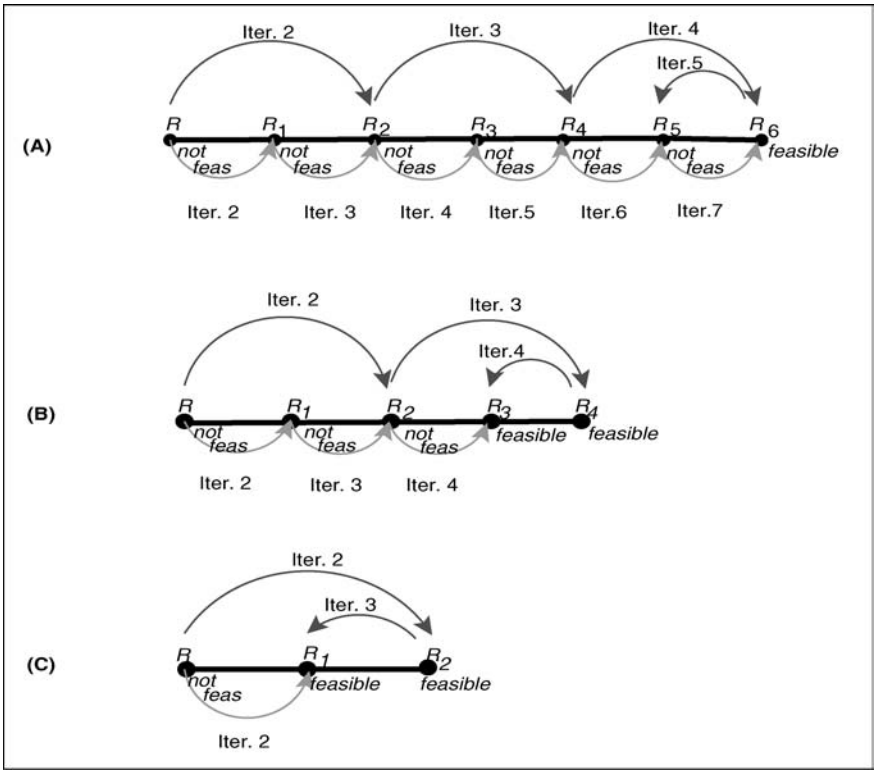

Figure 3. Some illustrative examples of IP iterations of the modified algorithm of Ilhan and Pinar [9].

distance greater than R (because IP problem not feasible at R) which is R_2, then jumps to R_4 which gives a feasible solution to the IP problem. To check whether the IP problem is optimal at R_4 or not, we need to check the feasibility of the IP problem at R_3 (iteration 4). As this problem happens to be feasible, then the optimal solution for the problem is R_3 not R_4. In this example, the original algorithm and its modified version both performed the same number of IP iterations, namely 4 iterations each, to find the optimal solution.

In part C of Figure 3, the original algorithm moves to R_1 where the IP problem is feasible and stops (iteration 2). Our modified algorithm jumps from R to R_2 where IP problem is found to be feasible (iteration 2), then an additional iteration at R_1 is performed to confirm infeasibility and hence guarantees that R_1 is optimal. It is only in this case, when the lower bound yielded by the first phase, for the problem at hand, is extremely tight and requires only one IP iteration, where the modified algorithm carries out one extra iteration.

2.2. ANALYTICAL RESULTS

The saving of the total number of IP iterations (i.e., solving IP problem) performed by the modified algorithm to reach the optimal solution can be summarised as follows:

$$\eta = \begin{cases} \left\lceil \dfrac{k-3}{2} \right\rceil & \text{if } k \geqslant 5, \\ 0 & \text{if } k = 3, 4, \\ 0 & \text{if } k = 1, \\ -1 & \text{if } k = 2. \end{cases} \tag{12}$$

Where η is the number of IP iterations which can be saved when using the modified algorithm, k is the number of IP iterations needed to reach the optimal solution by the original algorithm including $k = 1$ for the initial R value, and $\lceil x \rceil$ denotes the smallest integer greater than or equal to x.

In summary, it can be shown analytically that our modification will require less IP iterations if the original implementation requires 5 or more iterations, and it will need the same number if the original algorithm requires 1, 3 or 4 iterations. It is only for the special case when the original algorithm requires 2 iterations where our modification uses one more iteration.

The computational results for this modification are given in Section 4 along with other results.

3. Daskin [3] Algorithm

Based on Minieka's [11] algorithm, Daskin proposed an improved algorithm to find the optimal solution of the vertex P-center problem. This algorithm can be described as follows:

- Select initial lower and upper bounds on the value of the P-center objective function.
- Solve the Set Covering Problem (SCP) using the average (D) of lower and upper bounds as the coverage distance (rounded down) and let k be the number of facilities found to cover all nodes (or customers). If $k \leqslant P$, reset the value of the upper bound to D, else ($k > P$) reset the lower bound to $D + 1$.
- If the lower and upper bounds are equal then the lower bound (or the upper bound) is the optimal solution to the P-center problem and stop; otherwise solve the SCP with the new coverage distance (D) set to the average of the lower and upper bounds (rounded down), and continue the process.

The main steps of this algorithm are given in Figure 4.

This approach is based on searching over the range of coverage distances for the smallest coverage distance that allows all demand nodes to be covered. This search procedure is usually referred to as the *binary search.*

One way to reduce the number of iterations needed to find the optimal solution of the P-center problem using this algorithm could be done by either introducing

Step 0. Set $L = 0$ (zero) and $U = \max_{i,j}(d_{ij})$.

Step 1. Calculate $D = \left\lfloor \frac{L+U}{2} \right\rfloor$

Step 2. Solve SCP for the coverage distance D, and let k be the number of facilities found.

(i) If SCP is feasible (i.e. $k \leqslant P$) then set $U = D$.

(ii) Else (i.e. SCP is infeasible, $k > P$) set $L = D + 1$.

Step 3. If $L = U$, then the optimal solution is L, and stop.

Else go to Step 1.

Figure 4. The original algorithm of Daskin [3].

tighter initial lower and upper bounds or by designing a more powerful binary search. Our modifications are based on these two observations.

3.1. TIGHTENING OF THE INITIAL BOUNDS

Lower Bound

We use the initial lower bound (L) as the Pth minimum value of the distances, in the distance matrix. In other words, we order the distance values ascendingly, including ties and excluding the diagonal values which equal to zero, until the Pth minimum value which is used as the initial lower bound (L).

Note that there is no need to order all the values in the distance matrix after the Pth minimum.

This setting guarantees that $R^* \geqslant L$ where R^* is the optimal covering radius. In addition, as the ties are included, it is impossible to have $R < L$ using P facilities.

Upper Bound

The initial upper bound (U) is generated by finding the maximum distance value of each row in the distance matrix and then choosing the minimum value among them as the initial upper bound. In other words, $U = \min_i\{\hat{d}_i : \hat{d}_i = \max_j\{d_{ij}\}\}$, where d_{ij} is the distance between demand nodes i and j.

This setting guarantees that $R^* \leqslant U$ as the facility which is located at node i^* which yields U can serve all customers within a radius of U and hence the remaining $(P - 1)$ facilities will only attract some of these customers (as these may be nearer to them) and hence make $R \leqslant U$.

3.2. MODIFICATION BASED ON GOLDEN SECTION METHOD

The binary search (bisection method) which is used in the original algorithm to speed up the process of minimising the gap between the upper and lower bounds is slower to converge relative to other techniques such as the golden section method.

Golden Section is a commonly used numerical technique in finding a good estimate of the minimiser of one-dimensional functions (i.e., minimise functions of one variable), within an interval $[a, b]$. The golden search requires that two interior points, x_1 and x_2, to be used, where $x_1 = a + (\frac{1}{\gamma})^2(b-a)$ and $x_2 = a + (\frac{1}{\gamma})^2(b-a)$ and $\frac{1}{\gamma}$ is the golden ratio which equal to 0.618.

The proposed modification uses x_1 and x_2 (adaptively) as the coverage distance of the SCP with a and b replaced by the initial L and U, respectively. Note that since all distance values are integers, x_1 and x_2 mentioned here are rounded down.

Description of the Modification

The first step is to find an initial range, $[a, b]$, which contains the optimal solution (i.e., initial lower and upper bounds), where the initial values of a and b are the minimum and the maximum distance values in the distance matrix, respectively. This can be done by setting the coverage distance to the value of x_1 and checking the feasibility of the SCP at this coverage distance. If the solution is found feasible then the optimal solution occurs between a and x_1; otherwise we check feasibility at x_2. If the solution is feasible then the optimal solution exists between x_1 and x_2, otherwise the optimal solution occurs between x_2 and b. If the optimal solution occurs between a and x_1 (or x_2 and b) we solve the problem using x_1 (or x_2) until we get an infeasible solution at x_1 (or a feasible one at x_2). Each time we check x_1 or x_2 we record the number of facilities (k) found to cover all nodes (say $P_1 = k$ if the SCP is infeasible and $P_2 = k$ if the SCP is feasible). Note that the initial values of P_1 and P_2 are set to N and 1, respectively. After defining the range of the optimal solution and the corresponding number of facilities needed to cover all nodes for a and b, we use the following updating rule to choose between x_1 or x_2 at the next iteration as the coverage distance for the SCP:

1. Set $Q_1 = P - P_2$ and $Q_2 = P_1 - P$

1.1. If $Q_1 > Q_2$ (i.e., we are near feasible solution), use x_1 as the coverage distance.

1.2. If $Q_1 < Q_2$ (i.e., we are near infeasible solution), use x_2 as the coverage distance.

1.3. If $Q_1 = Q_2$, use x_1 as the coverage distance.

We would like to mention that in our first attempt we used x_1 and x_2 separately as the coverage distance of the SCP, but we found that using them adaptively as described above is more informative besides producing better results. The steps of the proposed scheme are given in Figure 5.

Step 0. (i) Set $L = \min_{i,j}(d_{ij})$, $U = \max_{i,j}(d_{ij})$ and $Count = 0$.
(ii) Set $P_1 = N$ and $P_2 = 1$.

Step 1. (a) Calculate $x_1 = \lfloor L + 0.382 * (U - L) \rfloor$ and set $D = x_1$.
(b) Solve SCP for the coverage distance D, and let k be the number of facilities found,
(i) If SCP is feasible (i.e. $k \leqslant P$) then set $U = D$, $P_2 = k$ and $Count = Count + 1$. Go to Step 1(a).
(ii) Else (i.e. SCP is infeasible, $k > P$) set $L = D$, and $P_1 = k$.
(iii) If $Count \geqslant 1$ go to Step 3, else set $L = \min_{i,j}(d_{ij})$, and go to Step 2.

Step 2. (a) Calculate $x_2 = \lfloor L + 0.618 * (U - L) \rfloor$ and set $D = x_2$.
(b) Solve SCP for the coverage distance D, and let k be the number of facilities found;
(i) If SCP is feasible then set $U = D$, $P_2 = k$, and go to Step 3.
(ii) Else set $L = D$, $P_1 = k$, and go to Step 2(a).

Step 3. Set $Q_1 = P - P_2$ and $Q_2 = P_1 - P$.
– If $Q_1 > Q_2$ then compute x_1 and set $D = x_1$.
– Else if $Q_1 < Q_2$ then compute x_2 and set $D = x_2$.
– Else $Q_1 = Q_2$, compute x_1 and set $D = x_1$.

Step 4. (a) – If $U - L = 2$ and there is no distance values (d_{ij}) between U and L then stop. The optimal solution is U.
– Else set $D = L + 1$ and solve SCP for the coverage distance D;
(i) If SCP is feasible then stop. The optimal solution is D.
(ii) Else the optimal solution is U and stop.
(b) If $U - L \leqslant 1$, then stop. The optimal solution is U.

Step 5. Solve SCP for the coverage distance D, and let k be the number of facilities found;
(i) If SCP is feasible then set $U = D$, $P_2 = k$, and go to Step 3.
(ii) Else set $L = D$, $P_1 = k$, and go to Step 3.

Figure 5. The proposed algorithm using Golden Section.

Observation. As we mentioned before the optimal solution of the problem is reached when $U - L \leqslant 1$. So, it is easy to notice that the optimal solution of the P-centre problem is U if $U - L \geqslant 2$ and there is no distance values between them (i.e. there is no clients at distance greater than L and less than U). Moreover, if there is just one distance value between U and L then we do not need to check at

x_1 or x_2 and it is enough to use this distance value as the coverage distance of the SCP and check it. If feasible (i.e. $k < P$) then this value is the optimal solution of the problem. If infeasible then the optimal solution is U.

4. Computational Results

The original and the modified algorithms of Ilhan and Pinar [9] and Daskin [3] are coded in C, and tested on a Sun Enterprise Workstation 450 running Solaris 2.6. We use Xpress-MP Optimisation software (Modeller Release 12.06, Optimiser Release 12.50) to solve the corresponding LP and IP problems. For comparison, we use the same instances as used in Ilhan and Pinar [9]. These problems were taken from the OR-Lib [1, 2] for the 40 P-median problems, and from the TSPLIB [12] for the Travelling Salesman Problems which we refer to as TSP instances. The same sets of data are used as a platform for testing the algorithm of Daskin [3].

In the OR-Lib test problems, the set of facilities is equal to the set of demand nodes or customers ($M = N$). We obtained the shortest path distances (d_{ij}) using the Floyd–Warshal algorithm [6]. The d_{ij} values are made integer since edges of the given networks of the 40 problem instances are integral. The problem parameters range from instances with $N = 100$ nodes and $P = 5, 10, 20$ and 33 up to instances with $N = 900$ and $P = 5$, 10 and 90.

As in the OR-Lib instances, the set of facilities for the TSP-Lib test problems is made equal to the set of demand nodes or customers (i.e., $M = N$). The coordinates of the nodes are given, and hence the shortest path distances are computed as the Euclidean distances between every pair of nodes. The entries of the distance matrix (d_{ij}) are non-integral, but as the original algorithms and their modified versions apply only to integral data, we rounded down every distance (d_{ij}) as used in the literature. The size of the instances range from 226 to 657 nodes. For each instance, four P values are used namely $P = 5$, 10, 20, and 40.

4.1. ILHAN AND PINAR'S [9] ALGORITHM

Table I reports the results, for the TSP and P-median instances, of the original algorithm and their two simple enhancements. Note that the new results could also be derived from the original paper and using Equation (12). We have recomputed all the results for consistency reasons only. Columns 1 and 7 give file number in case of P-median instances, and file name in case of TSP instances. Columns 2 and 8 refer to the number of vertices in the network (or number of demand points), whereas columns 3 and 9 refer to the optimal number of facilities needed to locate. Columns 4–6 and columns 10–12 represent the number of ILP iterations done in IP part (Phase 2) of Ilhan and Pinar [9] algorithm for TSP and P-median instances respectively. '*Original*' in columns 4 and 10 give the number of ILP iterations performed by the original algorithm (before any enhancement or modifications). '*Enh-1*' in columns 5 and 11 refer to number of ILP iterations carried out by the

Table I. Comparison results of the original algorithm of Ilhan and Pinar and its enhancements on TSP and OR-Lib instances

TSP-Lib instances						OR-Lib instances					
File Name	N	P	# ILP Iterations			File No.	N	P	# ILP Iterations		
			Original	Enh-1	Enh-2				Original	Enh-1	Enh-2
pr226	226	40	2	1	**1**	Pmed1	100	5	7	7	**5**
pr226	226	20	5	5	**4**	Pmed2	100	10	1	1	1
pr226	226	10	2	1	**1**	Pmed3	100	10	2	1	**1**
pr226	226	5	17	17	**10**	Pmed4	100	20	2	1	**1**
pr264	264	40	2?	1?	1?	Pmed5	100	33	1	1	1
pr264	264	20	2	1	**1**	Pmed6	200	5	**2***	2	3*
pr264	264	10	2	1	**1**	Pmed7	200	10	2	1	**1**
pr264	264	5	1	1	1	Pmed8	200	20	1	1	**1**
pr299	299	40	2	1	**1**	Pmed9	200	40	2	1	**1**
pr299	299	20	1	1	1	Pmed10	200	67	2	1	**1**
pr299	299	10	2	1	**1**	Pmed11	300	5	2	1	**1**
pr299	299	5	2	1	**1**	Pmed12	300	10	2	1	**1**
pr439	439	40	2	1	**1**	Pmed13	300	30	2	1	**1**
pr439	439	20	1	1	1	Pmed14	300	60	2	1	**1**
pr439	439	10	2	1	**1**	Pmed15	300	100	2	1	**1**
pr439	439	5	1	1	1	Pmed16	400	5	2	1	**1**
pcb442	442	40	6	5	**4**	Pmed17	400	10	2	1	**1**
pcb442	442	20	1	1	1	Pmed18	400	40	2	1	**1**
pcb442	442	10	2	1	**1**	Pmed19	400	80	2	1	**1**
pcb442	442	5	2	1	**1**	Pmed20	400	133	1	1	1
kroA200	200	40	2	1	**1**	Pmed21	500	5	1	1	1
kroA200	200	20	1	1	1	Pmed22	500	10	2	1	**1**
kroA200	200	10	2	1	**1**	Pmed23	500	50	2	1	**1**
kroA200	200	5	2	1	**1**	Pmed24	500	100	2	1	**1**
kroB200	200	40	2	1	**1**	Pmed25	500	167	2	1	**1**
kroB200	200	20	5	5	**4**	Pmed26	600	5	3	2	3
kroB200	200	10	2	1	**1**	Pmed27	600	10	2	1	**1**
kroB200	200	5	1	1	1	Pmed28	600	60	2	1	**1**
lin318	318	40	6	5	**4**	Pmed29	600	120	1	1	1
lin318	318	20	2	1	**1**	Pmed30	600	200	1	1	1
lin318	318	10	1	1	1	Pmed31	700	5	2	1	**1**
lin318	318	5	1	1	1	Pmed32	700	10	3	2	3
gr202	202	40	1	1	1	Pmed33	700	70	2	1	**1**
gr202	202	20	1	1	1	Pmed34	700	140	2	1	**1**
gr202	202	10	1	1	1	Pmed35	800	5	1	1	1
gr202	202	5	2	1	**1**	Pmed36	800	10	1	1	1
d493	493	40	2	1	**1**	Pmed37	800	80	1	1	1
d493	493	20	3	2	3	Pmed38	900	5	2	1	**1**
d493	493	10	2	1	**1**	Pmed39	900	10	2	1	**1**
d493	493	5	2	1	**1**	Pmed40	900	90	1	1	1
d657	657	40	6	5	**4**	**Total**			76	**49**	**50**
d657	657	20	2	1	**1**	# Best			-	**27**	**26**
d657	657	10	2	1	**1**						
d657	657	5	2	1	**1**						
Total			108	**80**	**69**						
# Best			-	**28**	**30**						

enhanced algorithm using the simple two enhancements of Step 5 and part a of modification of Phase 2 (i.e., *Existence of the value of R*). '*Enh-2*' in columns 6 and 12 report the number of iterations after using all modification schemes mentioned in Section 2.1 (i.e., modification of Step 5 and modifications of Phase 2). '# *Best*' in the last row of the table refers to the number of times the proposed algorithm (i.e., the enhancement scheme) outperformed the original algorithm excluding ties. Note that in the TSP instances, '?' in data file *pr264* ($P = 40$) means the original algorithm, and its modified algorithm could not find the optimal solution for the problem after 12 hours of running time. The number with '?' is recorded as the last iteration performed by the algorithm before stopping.

According to Table I the two enhancements (columns 5 and 11) reduced the number of ILP iterations for most instances by one iteration (bold numbers) comparing with the original algorithm (columns 4 and 10). Thus, the total number of ILP iterations decreased from 76 and 108 to 49 and 80 for the P-median and TSP instances respectively. It can be seen that the modified algorithm performed better than the original one for most instances. For example, in instance *pr226* ($P = 5$) the number of ILP iterations needed to solve the problem reduced to 10 iterations from 17 iterations, a significant saving of 7 ILP iterations. The instances which solved by the original algorithm in one ILP iteration cannot obviously be improved. Just one case the original algorithm performed better by using one less iteration (shown by *). This result coincides, according to Equation (12), with the case when $k = 2$ as mentioned in our analytical result. This is the only special case where the original implementation uses one less iteration.

4.2. DASKIN'S [3] ALGORITHM

Table II reports the results using the original algorithm of Daskin and the enhancement schemes discussed in Section 3. Columns 1–3 and 6–8 are as described earlier in Table I, whereas '*Original*', columns 4 and 11, report the number of IP iterations performed by the original algorithm (before enhancing it). '*Mod-1*' in columns 5 and 12 refer to the number of IP iterations carried out by the enhanced algorithm using the modification based on tighter initial lower and upper bounds. '*Mod-2*' in columns 6 and 13 report the number of IP iterations yielded when using Golden Section. '*Mod-3*' in columns 7 and 14 report the number of IP iterations after combining '*Mod-1*' and '*Mod-2*' (i.e., using tighter initial lower and upper bounds proposed in the simple enhancement instead of the proposed ones in Step 0 of Figure 5). '# *Best*' in the last row of the table refers to the number of times the proposed algorithm (i.e., the enhancement scheme) outperformed the original algorithm excluding ties. Note that the number of iterations in columns 4–7, and 11–14 refers to the number of times that the Set Covering Problem (SCP) is solved.

As can be seen from Table II, the proposed enhancements of the original algorithm reduced the number of IP iterations needed in most instances. Also, it may be possible to say that merging these two proposed modifications (or *Mod-3*)

Table II. Comparison results of the original algorithm of Daskin and its enhancements on TSP and OR-Lib instances

TSP-Lib instances							OR-Lib instances						
			# Iterations							# Iterations			
File Name	N	P	Original	Mod-1	Mod-2	Mod-3	File No.	N	P	Original	Mod-1	Mod-2	Mod-3
pr226	226	40	14	14	13	*10*	Pmed1	100	5	8	7	8	8
pr226	226	20	14	13	12	12	Pmed2	100	10	8	8	8	7
pr226	226	10	14	14	13	14	Pmed3	100	10	9	7	8	7
pr226	226	5	14	14	14	11	Pmed4	100	20	8	7	9	7
pr264	264	40	3?	4?	6?	6?	Pmed5	100	33	8	7	7	6
pr264	264	20	13	13	11	10	Pmed6	200	5	7	7	7	8
pr264	264	10	13	13	12	12	Pmed7	200	10	7	7	6	6
pr264	264	5	13	13	12	14	Pmed8	200	20	7	7	7	5
pr299	299	40	13	11	12	11	Pmed9	200	40	8	7	7	6
pr299	299	20	13	12	12	11	Pmed10	200	67	7	7	6	6
pr299	299	10	13	12	11	11	Pmed11	300	5	7	6	5	7
pr299	299	5	13	12	12	11	Pmed12	300	10	7	6	7	6
pr439	439	40	14	13	12	12	Pmed13	300	30	7	6	7	5
pr439	439	20	14	13	12	11	Pmed14	300	60	8	6	6	7
pr439	439	10	14	13	14	12	Pmed15	300	100	8	6	7	*5*
pr439	439	5	14	13	12	12	Pmed16	400	5	7	6	6	6
pcb442	442	40	12	12	13	11	Pmed17	400	10	7	6	6	6
pcb442	442	20	12	11	12	10	Pmed18	400	40	8	6	7	6
pcb442	442	10	12	11	11	11	Pmed19	400	80	7	6	6	6
pcb442	442	5	13	11	13	10	Pmed20	400	133	7	6	7	6
kroA200	200	40	12	11	11	11	Pmed21	500	5	7	6	5	7
kroA200	200	20	12	11	12	10	Pmed22	500	10	7	6	7	6
kroA200	200	10	12	11	11	10	Pmed23	500	50	7	5	7	5
kroA200	200	5	12	11	12	11	Pmed24	500	100	7	6	6	5
kroB200	200	40	12	11	10	9	Pmed25	500	167	7	6	6	*4*
kroB200	200	20	12	12	11	9	Pmed26	600	5	7	6	5	7
kroB200	200	10	12	11	11	9	Pmed27	600	10	6	6	5	5
kroB200	200	5	12	11	13	11	Pmed28	600	60	7	6	6	*4*
lin318	318	40	13	11	11	10	Pmed29	600	120	7	5	6	6
lin318	318	20	13	11	11	11	Pmed30	600	200	6	5	6	5
lin318	318	10	12	12	11	11	Pmed31	700	5	6	5	6	6
lin318	318	5	12	11	12	12	Pmed32	700	10	7	7	6	5
gr202	202	40	7	6	5	5	Pmed33	700	70	6	5	6	5
gr202	202	20	6	6	6	6	Pmed34	700	140	7	5	6	5
gr202	202	10	7	6	7	5	Pmed35	800	5	7	5	5	5
gr202	202	5	7	6	6	5	Pmed36	800	10	7	6	6	5
d493	493	40	12	11	10	10	Pmed37	800	80	6	5	6	5
d493	493	20	12	11	10	9	Pmed38	900	5	7	6	5	6
d493	493	10	12	11	11	11	Pmed39	900	10	7	6	6	5
d493	493	5	12	11	12	11	Pmed40	900	90	6	5	6	5
d657	657	40	12	11	11	10	Total			284	**243**	**254**	**232**
d657	657	20	12	11	11	10	# Best			-	**33**	**25**	**34**
d657	657	10	12	12	11	10							
d657	657	5	12	12	13	11							
Total			521	**486**	**482**	**443**							
# Best			-	**31**	**30**	**39**							

performs better than using them separately. So, the average number of IP iterations reduced to 5.80 and 10.30 from 7.10 and 12.12 for the OR-Lib and TSP instances, respectively. Moreover, 34 out of 40 OR-Lib instances and 39 out of 43 TSP instances are solved in fewer iterations than the original algorithm. For instance, 3 OR-Lib instances required 3 iterations less than the original algorithm (italic

Table III. Computing time for the exact methods of Ilhan and Pinar, and Daskin against our best implementation found for both data sets, TSP and OR-Lib instances

CPU time for TSP-Lib instances							CPU time for OR-Lib instances						
			Ilhan and Pinar		Daskin					Ilhan and Pinar		Daskin	
File Name	N	P	Orig-inal	Enh-2	Orig-inal	Mod-3	File No.	N	P	Orig-inal	Enh-2	Orig-inal	Mod-3
pr226	226	40	7.55	7.39	6.63	5.68	Pmed1	100	5	5.24	4.05	2.99	2.09
pr226	226	20	9.8	9.55	6.93	7.28	Pmed2	100	10	1.86	1.52	2.82	1.45
pr226	226	10	8.82	8.85	7.36	9.29	Pmed3	100	10	2.27	1.81	2.48	1.35
pr226	226	5	24.87	19.92	8.89	8.08	Pmed4	100	20	1.56	1.01	2.2	0.92
pr264	264	40	?	?	?	?	Pmed5	100	33	1.29	1.49	2.07	0.73
pr264	264	20	11.37	11.19	9.8	9.5	Pmed6	200	5	11.66	13.53	7.41	9.01
pr264	264	10	11.89	12	10.43	11.73	Pmed7	200	10	6.85	5.09	5.24	4.31
pr264	264	5	13.22	14.62	12.35	16.27	Pmed8	200	20	5.1	5.31	4.99	3.34
pr299	299	40	13.53	12.95	12.31	11.7	Pmed9	200	40	4.56	3.46	4.46	2.66
pr299	299	20	13.78	14.29	15.07	13.1	Pmed10	200	67	3.9	2.76	3.58	2.57
pr299	299	10	36.63	23.95	21.86	18.62	Pmed11	300	5	22.32	11.67	15.73	16.25
pr299	299	5	19.08	18.24	20.55	18.74	Pmed12	300	10	21.84	12.03	15.93	12.25
pr439	439	40	35.04	32.52	39.66	33.03	Pmed13	300	30	281.4	14.43	13.82	8.23
pr439	439	20	34.45	35.11	37.14	29.21	Pmed14	300	60	11.01	6.61	14.17	6.81
pr439	439	10	42.12	39.96	41.36	36.16	Pmed15	300	100	8.91	4.43	18.05	4.4
pr439	439	5	53.43	53.08	53.13	51.74	Pmed16	400	5	42.7	30.01	33.06	28.1
pcb442	442	40	6030	2714	805.4	1302	Pmed17	400	10	137.5	30.88	27.75	27.06
pcb442	442	20	38.67	37.62	170.6	151.2	Pmed18	400	40	20.51	12.49	23.13	13.17
pcb442	442	10	40.84	34.3	51.87	128.8	Pmed19	400	80	15.72	9.89	29.46	10.16
pcb442	442	5	44.79	40.31	55.23	41.37	Pmed20	400	133	13.35	13.53	24.04	9.3
kroA200	200	40	5.55	5.19	5.06	5.02	Pmed21	500	5	56.05	56.4	57.75	56.01
kroA200	200	20	5.76	5.51	5.8	5.38	Pmed22	500	10	547.1	495.2	64.59	60.78
kroA200	200	10	8.64	6.57	7.57	6.83	Pmed23	500	50	70.98	28.52	42.83	16.45
kroA200	200	5	9.05	8.22	8.13	8.28	Pmed24	500	100	24.62	14.64	36.09	12.59
kroB200	200	40	5.6	5.08	4.76	4.29	Pmed25	500	167	22.32	13.06	39.41	10.28
kroB200	200	20	9.03	8.27	5.64	4.75	Pmed26	600	5	397.9	401.6	106.4	104.2
kroB200	200	10	7.73	6.69	7.17	5.84	Pmed27	600	10	103.9	78.24	84.39	65.23
kroB200	200	5	7.68	7.63	7.34	7.53	Pmed28	600	60	48.82	39.51	51.25	19.31
lin318	318	40	558.13	219.4	13.6	11.67	Pmed29	600	120	30.97	32	40.29	23.61
lin318	318	20	14.6	13.54	15.53	14.49	Pmed30	600	200	33	34.72	123.4	17.22
lin318	318	10	17.83	17.46	19.29	18.52	Pmed31	700	5	335.9	303	131.1	122.6
lin318	318	5	18.85	18.68	19.67	22.06	Pmed32	700	10	1432	1447	166.9	116.8
gr202	202	40	5.41	5.15	5.1	3.81	Pmed33	700	70	97.48	94.07	181.2	33.11
gr202	202	20	5.87	5.65	5.01	4.63	Pmed34	700	140	52.9	50.23	159.2	29.66
gr202	202	10	6.61	6.79	6.33	4.89	Pmed35	800	5	183.3	183.9	204.6	123.3
gr202	202	5	9.99	8.59	8.13	5.83	Pmed36	800	10	3634	3602	196.2	110.5
d493	493	40	333.4	45.38	73.03	79.97	Pmed37	800	80	105.4	105.8	280.4	49.02
d493	493	20	1410	1406	85.15	75.54	Pmed38	900	5	313.2	251	395.7	273.1
d493	493	10	85.71	60.13	82.69	82.25	Pmed39	900	10	6024	5817	278.8	208.7
d493	493	5	77.95	72.24	77.83	75.41	Pmed40	900	90	241.7	240.8	629.8	462.9
d657	657	40	8937	3126	1207	351.7	Total			14374	**13475**	3524	**2079**
d657	657	20	1047.5	301.15	215.4	255.8							
d657	657	10	2586.3	751.01	192.1	156.6							
d657	657	5	117.87	100.56	158.6	154.7							
Total			21783	**9352**	3622	**3269**							

numbers), and 13 instances are solved using 2 iterations less. Also, for the TSP instances a reduction of 4 iterations is obtained in one instance (italic number), and reductions of 3 and 2 iterations in 9 and 19 instances, are recorded respectively. It can be observed that our proposed scheme required one additional iteration than the original algorithm in one instance from each data set only.

Table III reports the CPU time (in seconds) for both exact methods and the two best implementations, namely *Enh-2* in case of Ilhan and Pinar algorithm, and *Mod-3* in case of Daskin algorithm, respectively. Note that in the TSP instances, '?'

in data file *pr264* ($P = 40$) means the original algorithm, and our best implementation could not find the optimal solution for the problem after 12 hours of running time.

5. Conclusion

In this paper we show how simple and easy to implement enhancements can improve results of some existing algorithms. In this work, we introduced modifications of two exact algorithms for solving the vertex P-center problem. For Ilhan and Pinar's [9] algorithm, we proposed two enhancements to reduce the number of ILP iterations carried out by the second part (IP phase) of the algorithm which is the most time consuming part. The obtained results showed that this enhanced algorithm performs better than the original algorithm if the number of IP iterations needed to get the optimal solution is more than 4 iterations and will be no worse (same iterations) if the required number of iterations is 1, 3 or 4. However, it will need one extra IP iteration only in the case where 2 iterations are needed to find the optimal solution. In addition, the enhancement always reduces the number of ILP iterations by 1 if the resulting LP problem happens to be feasible for phase 2. For Daskin's [3] algorithm, we introduced into the original algorithm a simple enhancement tighten both the initial lower and upper bounds. Moreover, the Golden Section method, as our binary search, proved to be more efficient than the previously used bisection method. The combination of these two modifications yielded a reasonable reduction in the number of IP iterations.

As for future work the authors are investigating the use of a 'good' heuristic to produce tighter upper bound, which can then be used as a measure to define a corresponding tighter lower bound. Such upper and lower bounds will obviously help in speeding up the process of finding the optimal solution even further. In the Ilhan and Pinar's algorithm, our jump-based scheme could also be extended to consider not only a jump of two for all iterations but of m steps where the value of m could be adaptively adjusted at each iteration. Such an adaptive learning, which in itself is a challenging research issue, could reduce the number of IP iterations significantly, and hence these ideas are worth being extended to more complex covering-type problems.

Acknowledgement

The authors are grateful to both referees for their constructive comments. We would also like to thank the Saudi-Government for the sponsorship of the first author.

References

1. Beasley, J. E.: A note on solving large P-median problems, *European J. Oper. Res.* **21** (1985), 270–273.
2. Beasley, J. E.: OR library: Distributing test problems by electronic mail, *J. Oper. Res. Soc.* **41**(11) (1990), 1069–1072.

3. Daskin, M. S.: *Network and Discrete Locations: Models, Algorithms, and Applications*, Wiley, New York, 1995, pp. 154–191.
4. Daskin, M.: A new approach to solve the vertex P-center problem to optimality: Algorithm and computational results, *Comm. Oper. Res. Soc. Japan* **45**(9) (2000), 428–436.
5. Elloumi, S., Labbé, M. and Pochet, Y.: New formulation and resolution method for the P-center problem, http://www.optimization-online.org/DB.HTML/2001/10/394.html, 2001.
6. Floyd, R.W.: Algorithm 97, Shortest Path, *Comm. Assoc. Comput. Mach.* **5** (1962), 345.
7. Hakimi, S. L.: Optimum locations of switching centers and the absolute centers and medians of a graph, *Oper. Res.* **12** (1964), 450–459.
8. Hakimi, S. L.: Optimum distribution of switching centers in a communications network and some related graph theoretic problems, *Oper. Res.* **13** (1965), 462–475.
9. Ilhan, T. and Pinar, M.C.: An efficient exact algorithm for the vertex P-center problem, http://www.optimization-online.org/DB.HTML/2001/09/376.html, 2001.
10. Kariv, O. and Hakimi, S.: An algorithmic approach to network location problems. Part I: The P-centers, *SIAM J. Appl. Math.* **37** (1979), 513–538.
11. Minieka, E.: The m-center problem, *SIAM Rev.* **12** (1970), 138–139.
12. Reinelt, G.: TSPLIB – A traveling salesman problem library, *ORSA J. Comput.* **3** (1991), 376–384.