Ann Oper Res (2009) 167: 307–325
DOI 10.1007/s10479-008-0339-9

Optimal location with equitable loads

Oded Berman · Zvi Drezner · Arie Tamir · George O. Wesolowsky

Published online: 8 April 2008

Abstract The problem considered in this paper is to find p locations for p facilities such that the weights attracted to each facility will be as close as possible to one another. We model this problem as minimizing the maximum among all the total weights attracted to the various facilities. We propose solution procedures for the problem on a network, and for the special cases of the problem on a tree or on a path. The complexity of the problem is analyzed, $O(n)$ algorithms and an $O(pn^3)$ dynamic programming algorithm are proposed for the problem on a path respectively for $p = 2$ and $p > 2$ facilities. Heuristic algorithms (two types of a steepest descent approach and tabu search) are proposed for its solution. Extensive computational results are presented.

Keywords Equitable loads · Facility location · Network

1 Introduction

In this paper we consider the following problem. A network with weights (demands) generated at its nodes is given. We wish to locate p facilities on the nodes of the network. Each

O. Berman
Joseph L. Rotman School of Management, University of Toronto, 105 St. George Street, Toronto, ON M5S 3E6, Canada

Z. Drezner (✉)
Steven G. Mihaylo College of Business and Economics, California State University-Fullerton, Fullerton, CA 92834, USA
e-mail: zdrezner@fullerton.edu
url: http://business.fullerton.edu/isds/zdrezner

A. Tamir
Department of Statistics and Operations Research, School of Mathematical Sciences, Tel Aviv University, Ramat Aviv, Tel Aviv 69978, Israel

G.O. Wesolowsky
Faculty of Business, McMaster University, Hamilton, ON L8S 4M4, Canada

demand point selects its closest facility (in case of a tie, we either split the demand between the tying facilities, or apply a selection rule which assigns the whole demand to one facility). Our objective is to have "equitable" load at the facilities. We formulate the problem as minimizing the maximum weight assigned to each facility. Other objectives, such as the variance or the range of total weights assigned to the facilities, can also be used as the objective function of such a model. Note that the *average* load of the facilities is constant because the total load is fixed. Our objective can be viewed as minimizing the extra load above the average. If, for example, the maximum load is equal to the average load, then all loads must be equal to the average load and both the range of loads and the variance of loads is zero, thus minimal. The problem of locating p facilities on the unit square so as to minimize the maximal demand faced by each facility subject to closest assignments and coverage constraints is discussed in Baron et al. (2007) and Suzuki and Drezner (2008). The problem in the Euclidean plane was investigated in Drezner and Drezner (2006).

This problem is closely related to the capacitated p-median (van Roy 1986) or p-center problems (Mirchandani and Francis 1990; Drezner 1995). Each facility has a given load capacity (usually all facilities are identical thus having the same capacity). Capacitated problems seek the location of p facilities that minimize the objective function subject to the constraints that the loads at each facility do not exceed their capacities. In our problem there is no objective function of a p-median or p-center type. Our objective is to find the smallest possible capacity that will have a feasible solution to the capacitated p-median or p-center problem. In fact, the objective function of the underlying problem is irrelevant to our model. For example, if one uses the capacitated p-median problem formulation, then the objective function should be "0", but we need to find the smallest capacity that yields a feasible solution to such a problem. Therefore, the capacity is a variable, and our objective function is to minimize the capacity.

The problem is also related to the voting districting problem (Garfinkel and Nemhauser 1970; Hess et al. 1965; Meholtra et al. 1998). Districts need to be defined such that the number of voters in each district will be about the same for all districts. In the voting districting problem the districts need to "somehow" be connected and forming simple shapes. In our problem the "center" of each district is selected and the distribution of voters to a district is determined by their proximity to the selected center. Our solution (in Euclidean space) will guarantee convex, connected, well-shaped regions determined by the Voronoi diagram of the facilities.

There are many applications for this problem. The construction of pre-fabricated facilities such as health centers in an underdeveloped country, where there would be standard staff and equipment. A lowest possible capacity for such facilities would ensure that all facilities could cope with the demand. Carving market territories to be assigned to marketing representatives (Kalcsics et al. 2002). The objective is to assign to each territory equitable market potential. The objective of minimizing the maximum market potential among all territories serves this objective. Another example is the problem of designing machines with similar capacities in a production facility. The throughput of the system depends on the machine with the maximum load. Therefore, the objective is to minimize the maximum load among the machines. Designing p identical M/M/1 servers with a common given service rate for each of them (Berman and Larson 1985; Berman et al. 1987; Wang et al. 2002), such as ATMs, towing services, etc. This is modeled as the stochastic p-median problem. Note that the objective of minimizing the maximum of the waiting times at the facilities is equivalent to minimizing the maximum arrival rate at the facilities (which is proportional to the number of customers getting service at the facility) when service times at each facility are the same. Elaboration on this problem is detailed in Baron et al. (2008) and Berman and Drezner (2007).

In the next section we present and formulate two versions of the problem. The versions differ in the case of demand points that are equally close to more than one facility. In one version we do not allow splitting of nodes in the case of a tie whereas in the second version we allow splitting. In Sect. 3 we analyze the complexity of the problem and propose $O(n)$ algorithms and an $O(pn^3)$ dynamic programming algorithm for the solution of the problem on a path respectively for $p = 2$ and $p > 2$ facilities. We also present algorithms to find the optimal solution of the problem for two facilities on a tree. In Sect. 4 we present meta-heuristics for a general number of facilities, assuming that demand nodes can be split evenly between the facilities that are equally close to them.

2 Problem definition

Let $G = (V, E)$ be an undirected, connected graph with node set $V = \{v_1, \dots, v_n\}$ and edge set E. Each edge (link) has a positive length, and each node $v_i \in V$ is associated with a nonnegative demand weight w_i. For any pair of nodes, v_i, v_j, let $d_{ij} = d(v_i, v_j)$ denote the length of a shortest path in G connecting v_i and v_j. For any node v_i and $V' \subseteq V$, define $d(v_i, V') = d(V', v_i) = \min_{v_j \in V'}\{d(v_j, v_i)\}$.

Let $S \subseteq V$ be a subset of p nodes. For each node v_i let $S^i = \{v_j \in S : d(v_j, v_i) = d(v_i, S)\}$. A feasible S-assignment π is an assignment where each node v_i is assigned to some node in S^i. If v_i is assigned to $v_j \in S^i$, we have $\pi(i) = j$. The total load (weight) assigned to v_j is defined by

$$L_S^\pi(j) = \sum_{i:\pi(i)=j} w_i.$$

The maximum load of π is then defined by

$$L_S^\pi = \max_{v_j \in S}\{L_S^\pi(j)\}.$$

A feasible S-assignment π^* is called an optimal S-assignment if it has the smallest maximum load among all feasible S-assignments. The value $L_S^{\pi^*}$ is called the maximum load of S, and will be denoted by L_S^*.

The discrete equitable location problem, P_1, is to find $S \subseteq V$, $|S| = p$, minimizing the maximum load L_S^*.

In the continuous version of the model, the demand w_i of a node v_i can be split among the nodes in S^i. Any such (continuous) distribution δ of the demands $\{w_1, \dots, w_n\}$ is called a feasible S-distribution. We let δ_j^i denote the part of the demand w_i allocated to node v_j. δ_j will denote the total demand allocated to node v_j.

The maximum load of δ is then defined by

$$L_S^\delta = \max_{v_j \in S}\{\delta_j\}.$$

A feasible S-distribution δ^* is called an optimal S-distribution if it has the smallest maximum load among all feasible S-distributions. The value $L_S^{\delta^*}$ is called the maximum load of S, and will be denoted by L_S^*.

The continuous equitable location problem, P_2, is to find $S \subseteq V$, $|S| = p$, minimizing the maximum load L_S^*.

The above problems can be formulated as integer programs. Consider first the discrete model. Let us define two sets of binary decision variables x_j and y_{ij} as follows:

$x_j = 1$ if there is a facility located at node v_j, and 0 otherwise,

$y_{ij} = 1$ if node v_i is assigned to facility located at node v_j, and 0 otherwise.

The problem denoted by **(P₁)** is:

$$\textbf{(P}_1\textbf{)} \quad \min\{z\}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} y_{ij} w_i \le z \quad j = 1, \dots, n, \tag{1}$$

$$\sum_{j=1}^{n} x_j = p, \tag{2}$$

$$y_{ij} \le x_j \quad i, j, = 1, \dots, n, \tag{3}$$

$$\sum_{j=1}^{n} y_{ij} = 1 \quad i = 1, \dots, n, \tag{4}$$

$$\sum_{k=1}^{n} y_{ik} d_{ik} + (F - d_{ij}) x_j \le F \quad i, j = 1, \dots, n, \tag{5}$$

$$x_j, y_{ij} = 0, 1 \quad i, j = 1, \dots, n, \tag{6}$$

where F is a very large number ($F \ge \max_{i,j}\{d_{ij}\}$).

The formulation of the continuous problem, denoted by **(P₂)** is identical to the above except for constraints (6) where for **(P₂)**, $y_{ij} \ge 0$.

The objective function and constraints (1) ensure the minimax criterion. Constraint (2) limits the number of facilities to p. Constraints (3) ensure that node v_j cannot serve node v_i unless there is a facility located at node v_j. Constraints (4) state that each node is assigned to one facility. We prove that constraints (5) guarantee (for both P_1 and P_2) that each node is assigned to a closest facility.

Let $J = \{j | x_j = 1\}$. For $j \notin J$ (5) is always true because $x_j = 0$. By (3) $y_{ij} = 0$ for $j \notin J$. Therefore, the sum on the left hand side of (5) can be written as $\sum_{j \in J} y_{ik} d_{ik}$ and for $j \in J$ (5) can be written as

$$\sum_{k \in J} y_{ik} d_{ik} \le d_{ij} \tag{7}$$

For P_1: if $y_{ik} = 1$ for $d_{ik} > \min_t\{d_{it}\}$, constraint (7) will be violated for $d_{ij} = \min_t\{d_{it}\}$. Therefore, y_{ik} can be equal to 1 only for d_{ik} being the minimum distance. The objective function (1) takes care of ties because in case of a tie, it is preferred to assign the demand to a facility with a lower load.

For P_2: By (4), $\sum_{k \in J} y_{ik} d_{ik} \ge \min_t\{d_{it}\}$. If there exist $y_{ik} > 0$ such that $d_{ik} > \min_t\{d_{it}\}$, then $\sum_{k \in J} y_{ik} d_{ik} > \min_t\{d_{it}\}$ and constraint (7) will be violated for $d_{ij} = \min_t\{d_{it}\}$. Therefore, $y_{ik} > 0$ only for d_{ik} being the minimum distance.

2.1 Complexity analysis

2.1.1 Complexity results for problem P_1

If all the distances between pairs of nodes are distinct, then for each subset S of p nodes there is only one feasible assignment. Therefore, by considering the $O(n^p/p!)$ subsets of p nodes, problem P_1 can be solved by complete enumeration in $O(n^{p+1}/(p-1)!)$ time on a general network. In particular, in this case P_1 is polynomially solvable for any fixed value of p.

If the distances between pairs of nodes are not distinct, for each subset S we need to apply some algorithm to compute its maximum load. The latter problem is a special case of the minimum makespan scheduling problem of n jobs on p unrelated parallel machines (Horowitz and Sahni 1976). Specifically, each demand point is identified as a job, and each facility in S is identified as a machine. The processing time of job i on machine j is w_i for $j \in J$, and ∞ otherwise. This special scheduling model is called the restricted assignment model (Azar et al. 2004).

Lenstra et al. (1990) gave a 2-approximation algorithm for this problem even when p is variable. A stronger result which produces a feasible solution which is at most 2 times the optimal solution for all l_q norms ($q \geq 1$) simultaneously, is presented in Azar et al. (2004). (In the l_q-norm model, the goal is to minimize $((\delta_1)^q + \cdots + (\delta_p)^q)^{1/q}$, the l_q norm of the vector $(\delta_1, \ldots, \delta_p)$ whose components are the loads assigned to the p-facilities. In our case $q = \infty$.) Azar et al. also presented an FPTAS for the case where p is fixed. They produced an ϵ-approximation in $O(pn(pn/\epsilon)^p)$ time.

An exact pseudopolynomial algorithm with $O(\min[nW, p^n])$ complexity, ($W = \sum_{i=1}^{n} w_i$), was presented in Horowitz and Sahni (1976). (For this algorithm we assume that w_i is integer for $i = 1, \ldots, n$.) Thus, by considering the $O(n^p/p!)$ subsets of p nodes, problem P_1 can be solved by complete enumeration in $O(n^{p+1}W/(p-1)!)$ time on a general network.

We also note that unlike the nondegenerate case of distinct distances, in this case problem P_1 is (weakly) NP-hard even when $p = 2$, and G is a star tree.

Consider a star tree with one center node v_0 and n leaf nodes $\{v_1, \ldots, v_n\}$. Each leaf node is connected to v_0 with an edge of unit length. Set $w_0 = 0$. It is easy to see that when $p = 2$, determining whether the optimal solution value of P_1 is bounded by $W/2 = \sum_{i=1}^{n} w_i/2$, is equivalent to the partition problem (Garey and Johnson 1979).

We also observe that when p is variable (i.e., part of the input), problem P_1 is strongly NP-hard even for a star tree. The reduction is from the 3-partition problem, where $n = 3p$ (Garey and Johnson 1979). In the 3-partition problem we are given a set A of $3p$ elements, a positive integer bound B, and a positive integer weight $w(a)$ for each $a \in A$ such that $B/4 < w(a) < B/2$, and such that $\sum_{a\in A} w(a) = pB$. The question is whether A can be partitioned into p disjoint sets $\{A_1, \ldots, A_p\}$ such that for $i = 1, \ldots, p$, $\sum_{a\in A_i} w(a) = B$. (Note that each A_i must therefore contain exactly 3 elements of A.)

We reduce the 3-partition problem to a star tree as follows. The star tree has $3p$ leaf nodes, corresponding to the elements in A. The weight of a leaf is the weight of the respective element of A. The weight of the center node of the star is set to 0, and the length of each one of the $3p$ edges connecting the leaf nodes to the center is one unit. Again, as above, it is easy to see that problem P_1, defined on the star tree is equivalent to the given 3-partition problem.

2.1.2 Complexity results for problem P_2

The complexity for a variable p is unknown even on a general graph. However, we will show that for a general graph, P_2 is solvable in $O(n^{p+2}\log(n/p)/(p-1)!)$ time. We show how to compute an optimal distribution δ^* for a given subset S of p nodes, and the respective maximum load in $O(pn^2\log(n/p))$ time.

The optimal maximum load is the solution to the following transportation flow problem.

Suppose without loss of generality that $S = \{v_1, v_2, \ldots, v_p\}$. We construct the following directed bipartite graph $G' = (U^1 \cup S, E')$ with $U^1 = \{u_1, \ldots, u_n\}$. U^1 is identified as the set of n sources, where the supply at u_i is w_i. S is the set of sinks. There is a directed edge connecting u_i with v_j in E' if and only if the node v_j is in the set consisting of all the nodes of S in G that are closest to $v_i \in V$.

For each edge $(u_i, v_j) \in E'$, let δ^i_j denote the flow on the edge. Let δ_j denote the total flow into v_j, and let δ^i denote the outflow from u_i.

The flow problem is to find a flow vector δ, minimizing $\lambda = \max\{\delta_1, \ldots, \delta_p\}$, subject to the constraints $\delta^i = w_i$, for $i = 1, \ldots, n$. The optimal value L^*_S is clearly the smallest value of λ for which the above transportation flow problem with a capacity bound of λ at each sink has a feasible solution. We use the algorithm in Gallo et al. (1989) to solve this problem. First, we augment the above directed bipartite graph G' with a super source, say u_0, and connect it with a directed edge to each source u_i. The capacity of the augmented edge (u_0, u_i) will be set to w_i. Similarly, we augment a super sink, say v_0, and connect each sink v_j to v_0 with a directed edge of capacity λ. (The capacities of all other edges are equal to ∞.) The augmented graph is denoted by $G'(\lambda)$. Let $F(\lambda)$ be the maximum flow between u_0 and v_0 in $G'(\lambda)$. It was shown in Gallo et al. (1989) that the function $F(\lambda)$ is monotone, concave and piecewise linear with at most n breakpoints. Moreover, since $G'(\lambda)$ has $O(n)$ nodes and $O(pn)$ edges, a complete description of the list of the breakpoints can be obtained in $O(pn^2 \log(n/p))$ time. We note that L^*_S is the smallest value of λ solving the equation $F(\lambda) = W$. Hence, L^*_S is the largest breakpoint of the function $F(\lambda)$, and it can be computed in $O(pn^2 \log(n/p)))$ time. Thus, we conclude that by complete enumeration problem (P_2) can be solved in $O(n^{p+2} \log(n/p)/(p-1)!)$ time.

3 Problems P_1 and P_2 on a path and on a tree

In this section we consider the special case where $p = 2$ and the underlying network is either a tree or a path. We need the following definitions and properties.

Given a tree network $T = (V, E)$, a node v_M is a (weighted) median if $\sum_{j=1}^n w_j d(v_M, v_j) \le \sum_{j=1}^n w_j d(v_k, v_j)$, for any $k = 1, \ldots, n$. We let $V^M \subseteq V$ denote the set of all medians of T. It is well known that V^M induces a subtree of T, and it can be obtained in $O(n)$ time (Goldman 1971; Kariv and Hakimi 1979). Moreover, a node v_k is a median if and only if it is a (weighted) centroid, i.e., the total weight of any connected component obtained by the removal of v_k, is at most $W/2$.

3.1 Problems P_1 and P_2 on a path for $p = 2$

Assume that the path is defined by its edge set $E = \{(v_1, v_2), \ldots, (v_{n-1}, v_n)\}$. With this notation v_1 and v_n are the two leaves of the path, and for $i = 1, \ldots, n-1$, v_i is adjacent to v_{i+1}.

A node v_j on a path is a median if $\sum_{k=1}^j w_k \ge W/2$, and $\sum_{k=j}^n w_k \ge W/2$. V^M, the set of all medians of P is a subpath. If V^M contains more than one node its two endpoints, say nodes v_s and v_t have positive weights. Moreover, $\sum_{k=1}^s w_k = \sum_{k=1}^t w_k = W/2$. Assuming that $s < t$, the latter equation implies that $w_k = 0$ for all $s < k < t$. We note that the median set can be obtained in $O(n)$ time by solving the 1-median problem using Goldman's algorithm (1971).

The optimal values of both problems P_1 and P_2 on a path for $p = 2$ are clearly bounded below by $W/2$. Therefore, when the median set contains more than one node, an optimal solution to both problems is attained by setting facilities at nodes v_s and v_t. The optimal solution value is then $W/2$.

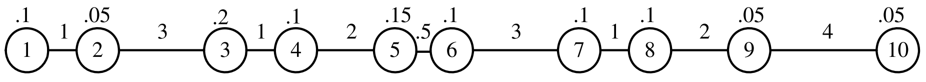

Fig. 1 A 10-nodes path

Next suppose that the unique median is node v_M. By definition we have $w_M > 0$, $\sum_{k=1}^{M} w_k \geq W/2$, $\sum_{k=M}^{n} w_k \geq W/2$, $\sum_{k=1}^{M-1} w_k < W/2$ and $\sum_{k=M+1}^{n} w_k < W/2$.

Due to symmetry, suppose without loss of generality that $\sum_{k=1}^{M} w_k \leq \sum_{k=M}^{n} w_k$.

We claim that an optimal solution to P_1 is attained by setting facilities at nodes v_M and v_{M+1}. The objective function value of this solution is $\max(\sum_{k=1}^{M} w_k, \sum_{k=M+1}^{n} w_k) = \sum_{k=1}^{M} w_k$.

To prove the claim, note that for any pair of facilities v_s and v_t, with $s < t$, there is a node v_q, such that $s \leq q \leq t$, and the respective objective function value is given by $\max(\sum_{k=1}^{q} w_k, \sum_{k=q+1}^{n} w_k)$. But, for any q, we have either $\{1, \dots, M\} \subseteq \{1, \dots, q\}$ or $\{M, \dots, n\} \subseteq \{q+1, \dots, n\}$. Hence,

$$\max\left(\sum_{k=1}^{q} w_k, \sum_{k=q+1}^{n} w_k\right) \geq \min\left(\sum_{k=1}^{M} w_k, \sum_{k=M}^{n} w_k\right) = \sum_{k=1}^{M} w_k.$$

Since the median node v_M can be found in $O(n)$ time using Goldman's Algorithm (1971), problem P_1 can be solved in $O(n)$ time on a path when $p = 2$.

Let us consider the 10-node path depicted in Fig. 1 where the numbers next to the nodes are weights and the number next to the links are their lengths. By applying Goldman's algorithm (1971) it is easy to verify that node 5 is the unique median. Since $\sum_{k=1}^{5} w_k = 0.6 > \sum_{k=5}^{10} = 0.55$, the optimal solution is to locate the two facilities at nodes 4 and 5 with an optimal objective function value of 0.55.

Next consider problem P_2. Suppose first that there are no pair of nodes v_a and v_b, such that $a < M < b$ and $d(v_a, v_M) = d(v_M, v_b)$. In this case the demand w_M will not be split among two nodes. Moreover, it is easy to see that in this case for any pair of facilities v_s and v_t, the respective objective function value is greater than or equal to $\min(\sum_{k=1}^{M} w_k, \sum_{k=M}^{n} w_k) = \sum_{k=1}^{M} w_k$. Again, it follows that in this case an optimal solution is attained by setting facilities at v_M and v_{M+1}.

Suppose now that there is a pair of nodes v_a and v_b, such that $a < M < b$, and $d(v_a, v_M) = d(v_M, v_b)$. It is easy to verify that the following is an optimal solution to problem P_2 with objective value equal to $W/2$: Set facilities at v_a and v_b. For each $j = 1, \dots, M-1, M+1, \dots, n$, arbitrarily assign the demand w_j to a closest facility. Split the demand w_M to $w_M^a \geq 0$ and $w_M^b \geq 0$, such that $w_M^a + w_M^b = w_M$, and $w_M^a + \sum_{k=1}^{M-1} w_k = w_M^b + \sum_{k=M+1}^{n} w_k = W/2$.

We show that even in this case the effort needed to solve problem P_2 is also $O(n)$. It is sufficient to show that in $O(n)$ time we can check the existence of nodes v_a and v_b as above. Indeed, in $O(n)$ time we compute the two sorted sequences of distances $\{d(v_1, v_M), d(v_2, v_M), \dots, d(v_{M-1}, v_M)\}$ and $\{d(v_n, v_M), d(v_{n-1}, v_M), \dots, d(v_{M+1}, v_M)\}$. Next, in $O(n)$ time we merge the two sequences in nonincreasing order. The existence of a pair v_a, v_b can then be checked in $O(n)$ time by scanning the merged list. The above is summarized in the next lemma.

Lemma 1 *When $p = 2$, both problems P_1 and P_2 defined on a path network can be solved in $O(n)$ time.*

We note that the lengths of the edges (distances) are irrelevant to finding an optimal solution to problem P_1 when the network is a path and $p = 2$ (the same is true also when solving the 1-median problem on a tree).

As an example consider again the path in Fig. 1 with one change: $d_{56} = 1$ instead of 0.5. Since $d_{15} = d_{95} = 7$, we locate the two facilities at nodes 1 and 9 and split the demand from node 5 by allocating 1/3 of the demand to the facility at node 1 and 2/3 of the demand to the facility at node 9.

3.2 An algorithm for problem P_1 on a path for $p \geq 2$

To solve problem P_1 when $p > 2$ we use dynamic programming.

Algorithm 1 For each $i = 1, \ldots, n$, we let P^i denote the subpath connecting v_i with v_n. Next, for each triplet (i, j, q), with $i \leq j$, and $q \leq p$, define $F(i, j, q)$ to be the optimal solution value of problem P_1, defined on P^i, provided that a total of q facilities are selected in P^i, and the leftmost facility is established at v_j. From this definition the optimal solution value of (P_1) is given by $\min_{1 \leq j \leq n-p+1} F(1, j, p)$.

It is then clear that $F(i, j, 1) = \sum_{k=i}^{n} w_i$, for all i, j, with $i \leq j$. To express the general recursive equations we need the following definition.

For any pair of indices $i < j$, define the index $t(i, j)$ to be the largest index t such that $d_{it} \leq d_{tj}$.

Consider the case where $q \geq 2$. We have the following recursive equations.

$$F(i, j, q) = \min_{j<k\leq n-q+2}\left[\min\left(A(j,k), \max\left[\sum_{m=i}^{t(j,k)} w_m, F(t(j,k)+1, k, q-1)\right]\right)\right],$$

where

$$A(j,k) = \max\left[\sum_{m=i}^{t(j,k)-1} w_m, F(t(j,k), k, q-1)\right],$$

if $d_{j,t(j,k)}) = d_{k,t(j,k)})$, and $A(j, k) = \infty$, otherwise.

We note that for any triplet (i, j, q) the effort to compute $F(i, j, q)$, using terms that have already been computed, is $O(n)$. Therefore, the total effort to solve problem P_1 is $O(pn^3)$. (The latter bound dominates the effort needed to compute all the indices $t(i, j)$, $1 \leq i \leq j \leq n$.)

Applying the algorithm to the 10-node example of Fig. 1 with $p = 3$, the optimal solution is to locate the three facilities at nodes 2, 5, and 7 (or 8) where nodes 1, 2, and 3 are assigned to the facility at node 2; nodes 4,5, and 6 are assigned to the facility at node 5; and nodes 7, 8, 9, and 10 are assigned to the facility at node 7 or 8. To illustrate the algorithm consider $F(1, 2, 3)$ which is the optimal objective function for three facilities given that the leftmost node is 2:

$$F(1,2,3) = \min_{2<k\leq 9}\left\{\min\left(A(2,k), \max\left(\sum_{m=1}^{t(2,k)} w_m, F(t(2,k)+1, k, 2)\right)\right)\right\}.$$

It can be verified that

$$F(1,2,3) = \min\left(A(2,5), \max\left(\sum_{m=1}^{t(2,5)} w_m, F(t(2,5)+1, 5, 2)\right)\right).$$

Since $t(2,5)=3$, node 3 is the rightmost node to node 2 that can be assigned to node 2,

$$F(1,2,3)=\min\left(A(2,5),\max\left(\sum_{m=1}^{3} w_m, F(4,5,2)\right)\right)$$

and $A(2,5)=\max\{\sum_{m=1}^{2} w_m, F(3,5,2)\}$. $A(2,5)$ represents the optimal solution taking into account the possibility for a tie in node 3 and that only node 2 and the rest of the nodes left to node 2 (node 1) are assigned to the facility at node 2 and the rest of the nodes are assigned optimally to two facilities the leftmost of them is at node 5. It is easy to verify that $A(2,5)=\max\{0.15,0.5\}=0.5$ where the other facility is located at node 6. In $F(1,2,3)$ we take into account that node 3 is assigned to the facility at node 2. It can be verified that $F(4,5,2)=0.35$ with the optimal solution of the other facility is node 7. Since $\min\{A(2,5),\max(\sum_{m=1}^{3} w_m, F(4,5,2))\}=\min\{0.5,\max(0.35,0.35)\}=0.35$, we obtain the optimal solution.

We note that in the cases where the number of facilities p is relatively large, i.e., $p=\Omega(\log n)$, there is a more efficient procedure to solve P_1. Consider the following algorithm:

Algorithm 2 For each $i=1,\dots,n$, Define $W_i=\sum_{k=1}^{i} w_k$. Set $W_0=0$. We observe that the optimal solution value to problem P_1 on a path is of the form W_j-W_{i-1}, for some pair of indices $1\le i\le j\le n$. Hence, the optimal value is an element in the set $W^*=\{W_j-W_{i-1}: 1\le i\le j\le n\}$, which is of $O(n^2)$ cardinality.

Next, for each $W\in W^*$, let $p(W)$ be the smallest number of facilities needed to ensure that no facility will serve customers of total weight exceeding W. (A customer has to be served by a closest facility.) It is clear that if $W'\le W$, then $p(W')\ge p(W)$. Therefore, the optimal value is the smallest value of $W\in W^*$ such that $p(W)\le p$, and we can apply a binary search on W^* to find the optimal value. Specifically, the optimal value can be obtained by calculating $O(\log n)$ values of $p(W)$. (Note that by applying the search procedures in Frederickson and Johnson 1984, and the references cited therein, we can search over W^* without explicitly generating all its $O(n^2)$ elements.)

We next show how to compute $p(W)$ in $O(n^3)$ time for a given value of W. This will lead to an $O(n^3\log n)$ algorithm to solve problem P_1 on a path.

For a pair (i,j), $i\le j$, let $p(i,j,W)$ denote the smallest number of facilities needed to serve the customers $[v_i,\dots,v_n]$, given that the leftmost facility is at v_j, and that no facility serves a total weight exceeding W. Without loss of generality suppose that $W\ge w_i$, for $i=1,\dots,n$.

As above, for any pair of indices $i<j$, define the index $t(i,j)$ to be the largest index t such that $d(v_i,v_t)\le d(v_t,v_j)$.

We clearly have $p(i,n,W)=1$, if $W_n-W_{i-1}\le W$ and $p(i,n,W)=\infty$, otherwise. Next, suppose that $i\le j<n$, and consider the term $p(i,j,W)$. Assume that $W_j-W_{i-1}\le W$, otherwise $p(i,j,W)=\infty$. For each $k>j$, define $B(j,k)=1+p(t(j,k)+1,k,W)$, if $W_{t(j,k)}-W_{i-1}\le W$, $B(j,k)=1+p(t(j,k),k,W)$, if $d(v_{t(j,k)},v_j)=d(v_{t(j,k)},v_k)$, $W_{t(j,k)}-W_{i-1}>W$, and $W_{t(j,k)-1}-W_{i-1}\le W$, and $B(j,k)=\infty$, otherwise.

We then have the following recursive equations.

$$p(i,j,W)=\min_{j<k\le n} B(j,k).$$

The effort to compute $p(i,j,W)$ is $O(n)$. The total time to compute $p(W)=\min_{j=1,\dots,n} p(1,j,W)$ is $O(n^3)$. Hence, we conclude that the total effort to solve P_1 is $O(n^3\log n)$.

3.3 An algorithm for problem P_2 on a path for $p \geq 2$

We first identify a set of polynomial cardinality W^*, containing the optimal solution value to problem P_2 on a path. For this model, in case of ties, w_j, the weight of node v_j, can be split between two closest facilities, which are at equal distance from v_j. If the optimal solution value is determined by a subset of customers whose weights are unsplit, then, the optimal value is given by $W_j - W_{i-1}$ for some subsequence of nodes, $[v_i, \dots, v_j]$, which are served by the same facility. If this is not the case, the optimal value is attained as follows. For some integer $q \leq p$, the demands of $[v_i, \dots, v_j]$ are equally split between q facilities selected from $\{v_i, \dots, v_j\}$. In particular, each one of the facilities will be allocated the optimal value, which is then equal to $(W_j - W_{i-1})/q$.

Next we determine the possible ways to split a demand w_t at optimality. Suppose that for some $i \leq t \leq j$, the customers $v_i, \dots, v_{t-1}$ are served by the first m facilities selected in $\{v_i, \dots, v_t\}$, and the demand w_t of v_t is split between the m-th facility and the $(m+1)$st facility into $w_t - y$ and y, respectively. y can then be determined by the equation

$$W_t - y - W_{i-1} = m(W_j - W_{i-1})/q.$$

Thus, for each pair (i, j), $i \leq j \leq n$, and each pair (m, q), $m \leq q \leq p$, y is uniquely determined. Altogether, we can assume without loss of generality that there are only $O(p^2 n^2)$ known splitting values that y can take on at v_t.

The above observations lead to a polynomial discretization of the (continuous) problem P_2. We can now modify the dynamic programming algorithm from the previous subsection to solve problem P_2.

For each triplet (i, j, q) and real y, with $i \leq j$, and $q \leq p$, define $F(i, j, q, y)$ to be the optimal solution value of problem P_2, defined on $P^i = [v_i, \dots, v_n]$, provided that a total of q facilities are selected in P^i, the leftmost facility is established at v_j, and the demands at nodes $v_i, \dots, v_n$ are $y, w_{i+1}, \dots, w_n$, respectively. From this definition the optimal solution value of P_2 is given by $\min_{1 \leq j \leq n-p+1} F(1, j, p, w_1)$.

From the above discussion we can limit the set of values that y can take on in $F(i, j, q, y)$ to a known subset of cardinality $O(p^2 n^2)$. We can then modify and mimic the recursive equations of the Algorithm 1 in the previous section. For the sake of brevity we skip the details. We only note that the overall complexity of this modified algorithm to solve P_2 is $O(p^5 n^7)$.

3.4 Problems P_1 and P_2 on a tree for $p = 2$

We have already observed in Sect. 2 that problem P_1 is (weakly) NP-hard on a tree even for $p = 2$, and problem P_2 has a polynomial time algorithm for any fixed value of p. In this section we refine these results for the case where the network is a tree and $p = 2$.

Suppose first that the tree network $T = (V, E)$ has a median set V^M containing at least two nodes. In this case, there are two medians, say v_s and v_t which are adjacent. Moreover, from the property that a median node is also a centroid, it follows that each one of the two subtrees obtained by removing the edge connecting v_s and v_t has a total weight equal to $W/2$ (Kariv and Hakimi 1979). Therefore, both problems, P_1 and P_2 are optimized by setting facilities at v_s and v_t.

Next suppose that the tree network $T = (V, E)$ has a unique median v_M. If we remove v_M from T, it is split into connected subtrees, say $\{T_1, T_2, \dots, T_m\}$, none of them containing a total demand exceeding $W/2$. (Note that unlike the case of a path, we may have $w_M = 0$.) For $k = 1, \dots, m$, let $W(T_k) = \sum_{v_j \in T_k} w_j$. Assume without loss of generality that $W(T_1) \geq W(T_k)$, $k = 2, \dots, m$.

Lemma 2 *Suppose that there is no pair of distinct nodes v_a and v_b such that $d(v_a, v_M) = d(v_b, v_M)$. Then an optimal solution to both problems P_1 and P_2 is obtained by setting facilities at v_M and at the node closest to v_M in T_1.*

Proof Let v_k denote the node closest to v_M in T_1. By the property of v_M it follows that if facilities are set at v_M and v_k, the objective value for both P_1 and P_2 is $\max(W - W(T_1), W(T_1)) = W - W(T_1)$. Consider a solution where facilities are established at nodes v_s and v_t, such that $d(v_s, v_M) < d(v_t, v_M)$. In particular, v_t does not coincide with v_M and the demand w_M is allocated to v_s.

If $v_t \in T_1$ the demands of all the nodes which are outside T_1 are allocated to v_s. Therefore, the objective value of such a solution is greater than or equal to $W - W(T_1)$. Suppose that $v_t \in T_k$ for some $k > 1$. Then the demands of all the nodes which are outside T_k are allocated to v_s. Therefore, the objective value of such a solution is greater than or equal to $W - W(T_k)$. By definition $W - W(T_k) \geq W - W(T_1)$. This completes the proof. □

Lemma 3 *Let v_a and v_b be a pair of distinct nodes such that $d(v_a, v_M) = d(v_b, v_M)$. Then an optimal solution to problem P_2 is obtained by setting facilities at v_a ans v_b. The optimal objective value is then $W/2$.*

Proof Suppose first that v_a and v_b are both in T_k for some $k = 1, \dots, m$. Define $W_a(T_k) = \sum_{v_j \in T_k : d(v_j, v_a) \leq d(v_j, v_b)} w_j$, and $W_b(T_k) = W(T_k) - W_a(T_k)$. The total demand of the nodes outside T_k, $W - W(T_k)$, is greater than or equal to $W(T_k)$. The demand of each node v_j outside T_k can arbitrarily be split between v_a and v_b. Let x_j be the part of w_j allocated to v_a. Any solution to the following continuous knapsack problem, $\sum_{v_j \in T - T_k} x_j = W/2 - W_a(T_k)$, $0 \leq x_j \leq w_j$, for all $v_j \in T - T_k$, will yield an objective value of $W/2$ to problem P_2.

Next, suppose that $v_a \in T_k$ and $v_b \in T_q$ for some pair of distinct indices $k, q = 1, \dots, m$. The demands of the nodes in T_k (T_q) are fully allocated to v_a (v_b). Again, as above, the demand of each node v_j outside $T_k \cup T_q$ can arbitrarily be split between v_a and v_b. Let x_j be the part of w_j allocated to v_a. Any solution to the following continuous knapsack problem, $\sum_{v_j \in T - (T_k \cup T_q)} x_j = W/2 - W(T_k)$, $0 \leq x_j \leq w_j$, for all $v_j \in T - (T_k \cup T_q)$, will yield an objective value of $W/2$ to problem P_2. This concludes the proof. □

From the above results we conclude that problem P_2 on a tree with $p = 2$ can be solved in $O(n \log n)$ time by the following algorithm.

Algorithm 3

Step 1: Use the algorithms in Goldman (1971) or Kariv and Hakimi (1979 to compute the weighted median set, $V^M \subseteq V$.

Step 2: If $|V^M| > 1$, let v_s and v_t be two adjacent nodes in V^M. Set facilities in v_s and v_t. Stop.

Step 3: Let $V^M = \{v_M\}$. Compute and sort the set of distances $\{d(v_j, v_M) : j = 1, \dots, n\}$. By scanning the list of the sorted distances, check whether there is a pair of distinct nodes v_a and v_b such that $d(v_a, v_M) = d(v_b, v_M)$.

Step 4: If there is a pair of distinct nodes v_a and v_b such that $d(v_a, v_M) = d(v_b, v_M)$, set facilities at v_a and v_b. Otherwise, remove v_M, and find the connected component with the maximum total node weights, say T_1. Set facilities at v_M and at the node of T_1, adjacent to v_M.

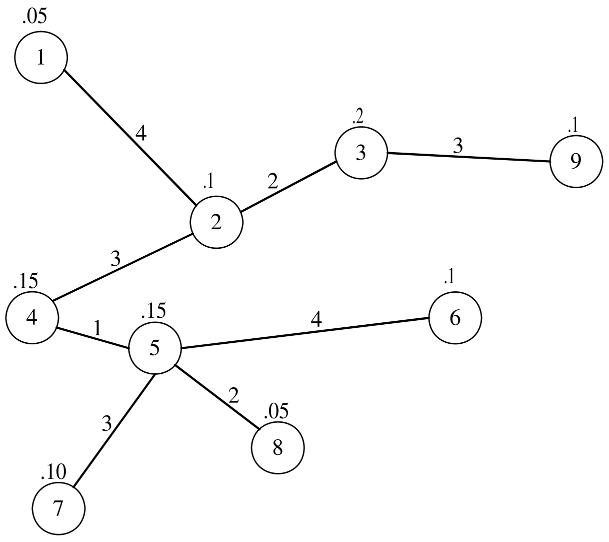

Fig. 2 The 9-node tree

The complexity of the above algorithm is dominated by the $O(n \log n)$ effort spent in Step 3 to sort the set of distances from the median node v_M.

To illustrate the procedure we refer to the 9-node tree example depicted in Fig. 2. The numbers next to the nodes and the links are demand weights and links lengths respectively. By applying Goldman's algorithm it is easy to verify that node 4 is the unique median. Since $d_{34} = d_{46} = 5$, we can locate the two facilities at nodes 3 and 6 and split the demand at node 4 which is equal to 0.15 by allocating $\frac{1}{3}$ of the demand (0.05) to the facility at node 3 and $\frac{2}{3}$ of the demand (0.15) to the facility at node 6. The optimal value of the objective function is 0.5.

4 Using metaheuristics

In this section we assume that if there is a tie in the closest distance to a facility, then the weight of the demand point is evenly divided among the tying facilities.

We propose to apply a descent algorithm, an improved descent algorithm, and a tabu search for the solution of this problem. Various metaheuristics were successfully applied for the solution of location problems such the p-median location problem which has a structure similar to our problem. Among these papers we mention Alp et al. (2003), Chiyoshi and Galvao (2000), Murray and Church (1996), Rolland et al. (1997), Hansen and Mladenovic (1997), Mladenovic et al. (2003, 2007).

In the following section a short cut for the calculation of the value of the objective function is described.

4.1 The short cut

When a facility is moved to another node, we first calculate the weights at each selected node by adding the portion of the weight assigned to the original node to the remaining selected nodes (only for nodes that were closest to the original node). Then, the weight of the newly selected node is calculated by considering all nodes that the new one is closest to them, and "moving" the weight from the previous closest node to the new one. Ties in the shortest distance have to be calculated properly. This approach is especially useful when all possible moves of facilities have to be evaluated. We need to calculate the effect of de-selecting the original node only once per all the evaluations the selection of new nodes. This scheme improved run times by orders of magnitude. We did not examine simulated annealing (Kirpatrick et al. 1983) because we lose the advantage of the short cut. Every iteration involves just one facility move and we lose the advantage of checking all possible moves efficiently.

4.2 A descent algorithm

A straight forward descent algorithm can be constructed based on the principle employed by Teitz and Bart's (1968) descent algorithm for the p-median problem.

1. Select p distinct nodes as a starting solution.
2. Go over all pairs of (selected node, non selected node) in random order and evaluate the change in the value of the objective function by moving a facility from the selected node to the non-selected node.
3. If an improving move is found, move the facility to the non-selected node which now becomes a selected node and go to Step 2.
4. The algorithm terminates when no facility move improves the value of the objective function.

4.3 An improved descent

The above descent algorithm is not that effective, because when there are several facilities which are tied for the maximum total weight, it is unlikely to be able to reduce all their weights by one move. We therefore suggest to establish ranking between two solutions which have the same maximum total weight. One possible ranking is the count of how many facilities tie for the maximum total weight. A lower number of tying facilities is better because it increases the chance that an additional move will improve the value of the objective function. We opted to have as a tie breaker, the variance of the total weights among the facilities. This idea is similar to the one suggested in Dear and Drezner (2000). Since the mean of the total weight is the same for any selection of nodes, the ranking of the variance is identical to the ranking of the sum of squares. The sum of squares of the total weights SS and the mean of the total weight (a constant W) are calculated. $SS/p - W^2$ is the variance of the vector of the total weights. We add to the value of the objective function $10^{-6} \times (SS - pW^2)$, and apply the same descent algorithm. Such an improved descent algorithm may perform moves that do not improve the value of the objective function, but improve the ranking, possibly resulting in better solutions. Since the number of iterations increases, the required run time for the improved descent is increased. See the computational results section.

4.4 A tabu search

We propose to extend the improved descent by a tabu search approach (Glover 1986; Glover and Laguna 1997). The tabu search proceeds from the terminal solution of the improved descent algorithm, by allowing upward moves in the hope that following several upward moves downward moves will be possible leading to a solution at a better local minimum. The tabu search can be visualized by the following description. A landscape has a lot of craters representing local minima. The search (for example, the terminal solution of a decent algorithm) is at the bottom of one crater when a local minimum is encountered. In order to move the search to a deeper crater we "must" go uphill, at least initially. The reverse of each selected move (constituting the tabu list) is disallowed for a given number of iterations (the tabu tenure T), so that the search will not "slide back" into the same crater. However, once T iterations have passed, the reverse move is allowed because we assume that we are already "outside" the crater and there is no need to restrict the search.

In our application of the tabu search, the tabu list consists of all nodes removed from the selected p nodes during recent iterations so that they cannot be selected again for T iterations.

4.4.1 The tabu search algorithm

1. Select the result of the improved decent algorithm as a starting solution and as the best found solution. Empty the tabu list.
2. Select the tabu tenure, T, in the range $[t_{\min}, t_{\max}]$. Go over all pairs of (selected node, non-selected node) in random order and evaluate the change in the objective function by moving the facility from the selected node to the non-selected one.
3. If a move (removing a facility from selected node i_{out}, and assigning a facility to a presently non-selected node i_{in}) yields a solution better than the best found one, move the facility, update the best found solution, empty the tabu list, and go to Step 2.
4. If no move yields a solution better than the best found solution, select the move (i_{out}, i_{in} as described above) which leads to the best value of the objective function (whether improving or not) as long as node's i_{in} tenure, if in the tabu list, does not exceed T.
5. Add node i_{out} to the tabu list, and if the length of the tabu list exceeds $t_{\max}$ remove the most tenured node from the tabu list. Go to Step 2.
6. Repeat Steps 2–5 until the prespecified number of iterations is reached.

We tested the tabu search with many variants of the parameters. The best parameters in our tests, that are reported in the computational experiments section, were $t_{\min} = 0.1 \times n$, $t_{\max} = 0.2 \times n$, and $5n$ iterations.

5 Computational experiments

We experimented with the 40 problems from the OR library suggested by Beasley (1990) for p-median experimentations. The problems range between 100 and 900 nodes, and p ranges between 5 and 200 facilities. Programs in FORTRAN were coded and run on a 2.8 GHz computer. In Table 1 we report the best found results for the descent algorithm, the improved descent algorithm (both were run for 1000 replications) and the tabu search (was run for 10 replications). The lower bound (LB) is simply $\frac{n}{p}$. In Table 2 we report the averages for each of the algorithms. For each problem and algorithm we report the percentages of: the best solution found, the average solution found, and the maximum solution found, over the best known solution reported in Table 1. We also report the total run time in seconds for all replications.

By examining Tables 1 and 2 we conclude that, in terms of solution quality, the tabu search is best, the improved descent is second, and the simple decent is third. The best solution found by the simple descent algorithm was, on the average, 20% over the best known, while the improved descent was only 0.8%, and the tabu search was only 0.16% over the best known. However, in terms of the run times they are ranked in reverse order.

Table 1 Best solutions found

n	p	LB	BK	Descent	Improved	Tabu
100	5	20.000	20.000	20.000	20.000	20.000
100	10	10.000	10.500	11.000	10.500	11.000
100	10	10.000	10.000	10.500	10.500	10.000
100	20	5.000	6.000	6.000	6.000	6.000
100	33	3.030	4.000	5.000	4.000	4.000
200	5	40.000	40.000	40.000	40.000	40.000
200	10	20.000	20.833	20.833	20.833	20.833
200	20	10.000	10.833	12.000	11.000	10.833
200	40	5.000	6.000	8.000	6.000	6.000
200	67	2.985	4.000	6.000	4.000	4.000
300	5	60.000	60.333	60.833	60.500	60.333
300	10	30.000	30.500	30.500	31.000	30.833
300	30	10.000	11.000	12.000	11.000	11.000
300	60	5.000	6.000	9.000	6.000	6.000
300	100	3.000	4.000	6.000	4.000	4.000
400	5	80.000	80.333	80.500	80.833	80.333
400	10	40.000	40.833	41.000	40.833	41.000
400	40	10.000	11.000	13.000	11.333	11.000
400	80	5.000	6.000	9.500	6.000	6.000
400	133	3.008	4.000	6.000	4.000	4.000
500	5	100.000	100.500	101.167	101.167	100.500
500	10	50.000	51.000	51.333	51.333	51.000
500	50	10.000	11.167	13.000	11.500	11.167
500	100	5.000	6.000	9.500	6.000	6.000
500	167	2.994	4.000	6.500	4.000	4.000
600	5	120.000	121.500	121.667	121.500	121.500
600	10	60.000	61.000	61.750	61.583	61.000
600	60	10.000	11.333	13.500	11.500	11.333
600	120	5.000	6.000	10.000	6.000	6.000
600	200	3.000	4.000	7.000	4.000	4.000
700	5	140.000	141.667	142.500	142.000	141.667
700	10	70.000	71.500	71.667	71.917	71.500
700	70	10.000	11.333	13.000	11.500	11.333
700	140	5.000	6.000	10.000	6.250	6.000
800	5	160.000	162.250	163.583	163.417	162.250
800	10	80.000	81.917	82.000	82.000	81.917
800	80	10.000	11.333	14.500	11.533	11.333
900	5	180.000	182.833	184.333	183.833	182.833
900	10	90.000	92.333	92.617	93.083	92.333
900	90	10.000	11.500	14.000	11.833	11.500

Table 2 Average performance

n	*p*	Descent				Improved descent				Tabu search			
		Percent over BK			Total	Percent over BK			Total	Percent over BK			Total
		Min.	Aver.	Max.	time	Min.	Aver.	Max.	time	Min.	Aver.	Max.	time
100	5	0.00	7.56	27.50	2.05	0.00	6.52	22.50	2.65	0.00	0.75	2.50	4.47
100	10	4.76	22.93	71.43	3.52	0.00	12.31	23.81	6.73	4.76	8.10	14.29	7.60
100	10	5.00	23.93	60.00	3.32	5.00	12.82	30.00	6.25	0.00	4.83	10.00	8.04
100	20	0.00	46.30	116.67	3.94	0.00	1.47	16.67	12.83	0.00	0.00	0.00	12.94
100	33	25.00	85.03	175.00	4.83	0.00	6.75	50.00	16.15	0.00	5.00	25.00	17.67
200	5	0.00	4.80	17.50	9.55	0.00	4.53	14.58	11.07	0.00	0.88	1.25	39.22
200	10	0.00	11.75	39.20	20.63	0.00	7.84	24.80	30.67	0.00	1.08	3.20	71.80
200	20	10.77	36.84	112.31	22.11	1.54	8.75	20.00	68.56	0.00	1.38	1.54	124.14
200	40	33.33	78.22	166.67	28.12	0.00	6.08	16.67	127.08	0.00	0.00	0.00	179.64
200	67	50.00	116.48	212.50	51.44	0.00	5.20	50.00	191.87	0.00	3.75	25.00	305.31
300	5	0.83	4.98	15.47	26.06	0.28	5.03	21.55	26.62	0.00	0.66	1.11	123.19
300	10	0.00	7.80	24.59	49.02	1.64	6.83	19.67	62.94	1.09	1.58	1.64	257.89
300	30	9.09	48.79	131.82	68.25	0.00	9.16	22.73	261.13	0.00	1.29	4.55	503.09
300	60	50.00	90.60	166.67	108.25	0.00	6.73	25.00	545.23	0.00	0.00	0.00	1045.33
300	100	50.00	119.18	225.00	194.17	0.00	3.68	62.50	856.99	0.00	1.25	12.50	1663.17
400	5	0.21	4.25	14.94	45.53	0.62	4.32	15.15	46.59	0.00	0.58	0.83	298.16
400	10	0.41	6.19	18.37	97.70	0.00	5.92	17.55	110.73	0.41	0.61	1.22	490.14
400	40	18.18	54.05	122.73	145.80	3.03	10.03	18.18	680.64	0.00	1.85	4.55	1653.02
400	80	58.33	102.90	183.33	324.77	0.00	9.28	25.00	1828.00	0.00	0.83	8.33	4040.35
400	133	50.00	105.48	225.00	485.05	0.00	4.25	37.50	2399.17	0.00	1.25	12.50	5749.32
500	5	0.66	3.74	10.28	73.78	0.66	3.81	16.17	73.94	0.00	0.67	1.66	591.25
500	10	0.65	5.52	21.57	164.43	0.65	5.33	15.69	178.97	0.00	0.51	1.63	1102.74
500	50	16.41	61.93	119.40	278.56	2.98	9.86	20.89	1494.34	0.00	1.93	2.98	4440.04
500	100	58.33	119.55	250.00	710.89	0.00	10.97	33.33	4033.61	0.00	2.22	8.33	9728.91
500	167	62.50	122.43	225.00	1075.36	0.00	3.48	25.00	5534.39	0.00	0.00	0.00	14696.25
600	5	0.14	3.09	13.10	106.15	0.00	3.18	16.94	110.20	0.00	0.21	0.55	976.71
600	10	1.23	6.41	21.04	258.25	0.96	6.18	24.64	281.33	0.00	1.22	1.91	1871.13
600	60	19.12	60.98	120.59	508.53	1.47	8.73	19.12	2823.51	0.00	1.03	1.47	9461.52
600	120	66.67	118.57	225.00	1273.14	0.00	12.17	25.00	7716.53	0.00	6.38	8.33	21810.53
600	200	75.00	132.73	312.50	2186.62	0.00	5.55	25.00	12280.75	0.00	0.00	0.00	36393.22
700	5	0.59	3.03	9.53	145.94	0.24	3.08	11.53	150.07	0.00	0.26	0.53	1482.17
700	10	0.23	4.63	15.15	344.22	0.58	4.55	11.65	362.43	0.00	0.43	0.93	2912.69
700	70	14.71	63.38	142.65	939.49	1.47	8.44	19.12	5442.67	0.00	1.10	1.47	20912.39
700	140	66.67	124.22	225.00	2374.57	4.17	12.20	25.00	14721.06	0.00	5.62	8.33	45469.84
800	5	0.82	3.26	15.00	199.22	0.72	3.44	12.24	198.43	0.00	0.28	0.62	2197.56
800	10	0.10	3.96	10.17	448.28	0.10	3.89	12.82	461.72	0.00	0.27	0.51	4266.83
800	80	27.94	64.38	129.42	1568.81	1.76	9.06	19.12	9251.69	0.00	0.72	1.47	40209.83
900	5	0.82	3.19	13.23	248.36	0.55	3.12	9.73	254.72	0.00	0.45	0.82	2674.08
900	10	0.31	4.28	12.28	565.69	0.81	4.20	12.85	585.71	0.00	0.60	1.26	5970.53
900	90	21.74	65.21	126.09	2429.27	2.90	8.23	17.39	15074.47	0.00	0.00	0.00	64472.25
Average		20.01	48.81	103.59	439.84	0.80	6.67	22.68	2208.06	0.16	1.49	4.32	7705.87

Run time for the improved descent was about five times longer than that for the simple descent. The total run time for the tabu search was about 3.5 times longer than that of the improved descent.

In only three cases the tabu search failed to find the best known solution, while the improved descent failed to find it in over half of the cases (and no case with $n \geq 700$). The descent algorithm found it only five times out of 40 problems. It is interesting that for one problem ($n = 300$, $p = 10$) the descent algorithm was the only one that found the best known solution. It seems that the most significant improvement is observed between the descent algorithm and the improved descent. It is not clear whether the extra improvement observed for the tabu search justifies the extra required computer time.

We also experimented with CPLEX in an attempt to solve the problems optimally. We were not able to solve even one problem out of forty. The five problems with $n = 100$ were terminated after two hours and the solutions at the time the run was stopped were far from the best known solutions reported in Table 1. They were (27, 23, 19, 11, 8) compared with (20, 10.5, 10, 6, 4), respectively. The problems with $n = 200$ were either out of memory or stopped after two hours with bad solutions. We also attempted problems with $n = 300, 400$ and the first phase was not completed because of "out of memory" error message so we did not even get an integer feasible solution.

6 Conclusions

We analyzed and solved the problem of locating p facilities such that each facility services about the same number of customers. The objective used for this equity measure is the minimization of the maximum number of customers serviced among all facilities. Properties and complexity analysis is presented for the general problem on a network. $O(n)$ algorithms are proposed for the problem on a path for the case $p = 2$ and an $O(pn^3)$ dynamic programming algorithm is presented for the general problem on a path. Optimal procedures are also presented for a tree network with $p = 2$ facilities.

Heuristic algorithms were suggested to solve large problems. We proposed a steepest descent, an improved descent, and tabu search. While the tabu search provides the best quality solutions, it takes the longest. The improved descent was second best both in terms of the quality of the solution and the run time. We recommend to apply the improved descent which yields pretty good solutions in a reasonable run time.

As future research we propose to design more efficient algorithms for the heuristic solutions of this problem. We also propose to consider other equity measures such as minimizing the variance or the range of the total weights. Another extension to this model is to consider a different selection rule by customers rather than the assumption that each customer chooses the closest facility. It may be more realistic to assume that customers select a facility according to the gravity rule (see for example, Drezner and Drezner 2001).

Acknowledgements This research was supported, in part, by the Natural Sciences and Engineering Research Council of Canada. Part of this research was accomplished while the second author was visiting the Graduate School of Management, University of California, Irvine.

References

Alp, O., Drezner, Z., & Erkut, E. (2003). An efficient genetic algorithm for the p-median problem. *Annals of Operations Research*, *122*, 21–42.

Azar, Y., Epstein, L., Richter, Y., & Woeginger, G. (2004). All-norm approximation algorithms. *Journal of Algorithms*, *52*, 120–133.

Baron, O., Berman, O., Krass, D., & Wang, Q. (2007). The equitable location problem on the plane. *European Journal of Operational Research*, *183*, 578–590.

Baron, O., Berman, O., & Krass, D. (2008, accepted). Facility location with stochastic demand and constraints on waiting time. *Manufacturing and Service Operations Management*.

Beasley, J. E. (1990). OR-library—distributing test problems by electronic mail. *Journal of the Operational Research Society*, *41*, 1069–1072. Also available at http://mscmga.ms.ic.ac.uk/jeb/orlib/pmedinfo.html.

Berman, O., & Larson, R. C. (1985). Optimal 2-facility network districting in the presence of queuing. *Transportation Science*, *19*, 261–277.

Berman, O., & Drezner, Z. (2007). The multiple server location problem. *Journal of the Operational Research Society*, *58*, 91–99.

Berman, O., Larson, R. C., & Parkan, C. (1987). The stochastic queue p-median problem. *Transportation Science*, *21*, 207–216.

Chiyoshi, F., & Galvao, R. D. (2000). A statistical analysis of simulated annealing applied to the p-median problem. *Annals of Operations Research*, *96*, 61–74.

Dear, R., & Drezner, Z. (2000). Applying combinatorial optimization metaheuristics to the golf scramble problem. *International Transactions of Operations Research*, *7*, 331–347.

Drezner, Z. (Ed.). (1995). *Facility location: a survey of applications and methods*. New York: Springer.

Drezner, T., & Drezner, Z. (2001). A note on applying the gravity rule to the airline hub problem. *Journal of Regional Science*, *41*, 67–73.

Drezner, T., & Drezner, Z. (2006). Multiple facilities location in the plane using the gravity model. *Geographical Analysis*, *38*, 391–406.

Frederickson, G. N., & Johnson, D. B. (1984). Generalized selection and ranking: sorted matrices. *SIAM Journal on Computing*, *13*, 14–30.

Gallo, G., Grigoriadis, M., & Tarjan, R. E. (1989). A fast parametric network flow problem. *SIAM Journal on Computing*, *18*, 30–55.

Garey, M. R., & Johnson, D. (1979). *Computers and intractability: a guide to the theory of NP-completeness*. New York: Freeman.

Garfinkel, R. S., & Nemhauser, G. L. (1970). Optimal political districting by implicit enumeration techniques. *Management Science*, *16*(8), B-495–B-508.

Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, *13*, 533–549.

Glover, F., & Laguna, M. (1997). *Tabu search*. Boston: Kluwer Academic.

Goldman, A. J. (1971). Optimal center location in simple networks. *Transportation Science*, *5*, 212–221.

Hansen, P., & Mladenovic, N. (1997). Variable neighborhood search for the p-median. *Location Science*, *5*, 207–226.

Hess, S., Weaver, J., Siegfeldt, H., Whelan, J., & Zitlau, P. (1965). Non-partisan political redistricting by computer. *Operations Research*, *13*, 993–1006.

Horowitz, E., & Sahni, S. (1976). Exact and approximate algorithms for scheduling nonidentical processors. *Journal of the ACM*, *23*, 317–327.

Kalcsics, J., Melo, T., & Nickel, S. (2002). Planning sales territories—a facility location approach. In ISOLDE IX Conference, Fredericton, New Brunswick, Canada, June 2002.

Kariv, O., & Hakimi, L. S. (1979). An algorithmic approach to network location problems. Part 2. The p-medians. *SIAM Journal on Applied Mathematics*, *37*, 539–560.

Kirpatrick, S., Gelat, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, *220*, 671–680.

Lenstra, J. K., Shmoys, D. B., & Tardos, E. (1990). Approximation algorithms for scheduling unrelated machines. *Mathematical Programming*, *46*, 259–271.

Meholtra, A. A., Johnson, E. L., & Nemhauser, G. L. (1998). An optimization based heuristic for political districting. *Management Science*, *44*, 1100–1114.

Mirchandani, P. B., & Francis, R. L. (Eds.). (1990). *Discrete location theory*. New York: Wiley.

Mladenovic, N., Labbe, M., & Hansen, P. (2003). Solving the p-center problem with tabu search and variable neighborhood search. *Networks*, *42*, 48–64.

Mladenovic, N., Brimberg, J., Hansen, P., & Moreno-Perez, J. A. (2007). The p-median problem: a survey of metaheuristic approaches. *European Journal of Operational Research*, *179*, 927–939.

Murray, A. T., & Church, R. L. (1996). Applying simulated annealing to location-planning models. *Journal of Heuristics*, *2*, 31–53.

Rolland, E., Schilling, D. A., & Current, J. R. (1997). An efficient tabu search procedure for the p-median problem. *European Journal of Operational Research*, *96*, 329–342.

Suzuki, A., & Drezner, Z. (2008, in press). The minimum equitable radius location problem with continuous demand. *European Journal of Operational Research*.

Teitz, M. B., & Bart, P. (1968). Heuristic methods for estimating the generalized vertex median of a weighted graph. *Operations Research*, *16*, 955–961.

van Roy, T. J. (1986). A cross decomposition algorithm for capacitated facility location. *Operations Research*, *34*, 145–163.

Wang, Q., Batta, R., & Rump, C. M. (2002). Algorithms for a facility location problems with stochastic customer demand and immobile servers. In O. Berman & D. Krass (Eds.), *Annals of operations research: Vol. 111. Recent developments in the theory and applications of location models part II* (pp. 17–34). Dordrecht: Kluwer Academic.