Chapter 2
The *p*-Median Problem

Mark S. Daskin and Kayse Lee Maass

Abstract The *p*-median problem is central to much of discrete location modeling and theory. While the *p*-median problem is $\mathbb{NP}$-hard on a general graph, it can be solved in polynomial time on a tree. A linear time algorithm for the 1-median problem on a tree is described. We also present a classical formulation of the problem. Basic construction and improvement algorithms are outlined. Results from the literature using various metaheuristics including tabu search, heuristic concentration, genetic algorithms, and simulated annealing are summarized. A Lagrangian relaxation approach is presented and used for computational results on 40 classical test instances as well as a 500-node instance derived from the most populous counties in the contiguous United States. We conclude with a discussion of multi-objective extensions of the *p*-median problem.

Keywords Algorithm • Center • Covering • Lagrangian relaxation • Median • Multi-objective

2.1 Introduction

The *p*-median problem is that of locating *p* facilities to minimize the demand weighted average distance between demand nodes and the nearest of the selected facilities. The problem dates back to the seminal work of Hakimi (1964, 1965). The *p*-median problem is one of several classical location problems which also include the capacitated and uncapacitated facility location problems (Chap. 3), the *p*-center problem (Chap. 4), covering problems (Chap. 5) and anti-covering problems (Chap. 6). The *p*-median problem lies at the heart of many practical location problems, and, as shown below (Sect. 2.7), some of the other classical location problems can readily be formulated as *p*-median problems, leading to multicriteria location problems as outlined in Chap. 9.

Our objective is not to review every paper and every result related to this seminal problem. Rather, we summarize key results, algorithms and important extensions.

M.S. Daskin (✉) • K.L. Maass

IOE Department, University of Michigan, 1205 Beal Avenue, Ann Arbor, MI 48109, USA
e-mail: msdaskin@umich.edu; leekayse@umich.edu

We refer the reader to ReVelle et al. (2008) for a fairly recent annotated bibliography of the *p*-median and related models.

The remainder of this chapter is organized as follows. Section 2.2 outlines several key properties of the problem. Section 2.3 discusses optimal solution algorithms for the problem on a tree. Section 2.4 formulates the *p*-median problem as an optimization problem. Section 2.5 outlines algorithms for the problem on a general network. Section 2.6 presents selected computational results. Section 2.7 outlines two key multi-objective extensions of the *p*-median problem. Finally, conclusions are briefly presented in Sect. 2.8.

2.2   Model Properties

There are three key properties of the *p*-median problem that are important to know. First, Kariv and Hakimi (1979) showed that the *p*-median problem is $\mathcal{NP}$-hard on a general graph. This is the bad news. The good news, as outlined below, is that there are many effective algorithms and approaches to solving the *p*-median problem.

Second, Hakimi (1965) showed that at least one optimal solution to the *p*-median problem consists of locating only on the nodes. To see that this is true, consider a solution that entails locating a facility somewhere on an edge between nodes A and B. Let D_A be the total demand served by this facility that enters the edge via node A, and let D_B be the total demand served by the facility that enters via node B. Clearly, if $D_A > D_B$ we can move the facility to node A and reduce the objective function. This contradicts the assumed optimality of the facility at an intermediate location on the edge. Similar arguments hold if $D_B > D_A$ in which case we move the facility to node B. If $D_A = D_B$ we can move the facility to either node without adversely impacting the objective function value. Note that moving the facility to one of the nodes may result in the reassignment of demands to or from the facility if doing so will reduce the objective function. Such reassignments will only improve the objective function. Also note that moving the facility to one of the nodes may also result in some demands that were served by the facility, and that entered via the other node, to now enter the facility directly without traversing the edge between A and B. This would occur if traveling directly to the facility is shorter than traveling via the edge between A and B. Finally, we note that the nodal optimality property holds if the distance between a demand node and a candidate facility site is replaced by any concave function of the distance.

Finally, the demand weighted total cost or distance (or the demand weighted average cost or distance) decreases with the addition of each subsequent facility. This is clearly true since, if there exists an optimal solution to the problem with p facilities, then adding a $p + 1^{st}$ facility at any of the candidate nodes that does not have a facility will decrease the demand-weighted total cost or distance and therefore will also decrease the objective function. Locating the $p + 1$ facilities optimally is clearly as good or better than first locating p facilities optimally and adding a subsequent facility to that solution.

Table 2.1 Median results for top 100 counties in US

p	Demand weighted average distance	Change	Sites
1	969.45		St. Louis, MO
2	450.65	518.80	San Bernardino, CA; Allegheny, PA
3	320.15	130.50	Los Angeles, CA; Shelby, TN; Hudson, NJ
4	257.23	62.92	Los Angeles, CA; Tarrant, TX; New York, NY; Jefferson, KY
5	190.22	67.01	Los Angeles, CA; Cook, IL; Dallas, TX; New York, NY; Orange, FL

We would also expect that the marginal improvement in the demand weighted total (or average) cost or distance would decrease monotonically as we add facilities. This is frequently the case, but not always. As an example of a situation in which this is not so, consider the *p*-median problem with the 100 largest counties in the contiguous United States based on the 2010 census. While these counties represent only 3.2 % of the 3,109 counties in the contiguous United States, they account for 42.2 % of the total population. Using great circle distances and population as a proxy for demand, we obtain the results shown in Table 2.1. The demand weighted average distance decreases with the number of facilities as shown in the second column. However, the *change* in the demand weighted average distance increases from about 63 miles to 67 miles as we increase from four to five facilities.

2.3   The *p*-Median Problem on a Tree

While the *p*-median problem is $\mathcal{NP}$-hard on a general graph, the problem can be solved in polynomial time on a tree. We illustrate this with a linear time algorithm for finding the 1-median on a tree, which was proposed by Goldman (1971). This algorithm also helps explain why the problem is called the "median" problem. If any node of the tree has half or more of the total demand of all nodes on the tree, then it is clearly optimal to locate at that node. Moving away from that node will move the facility further from half or more of the demand and closer to less than half of the demand, thereby increasing the objective function value.

To outline this algorithm, we define the following sets:

$$\mathcal{I} = \{1, \ldots, i, \ldots, m\} \text{ the set of candidate locations}$$

$$\mathcal{J} = \{1, \ldots, j, \ldots, n\} \text{ the set of demand nodes.}$$

Fig. 2.1 Example tree
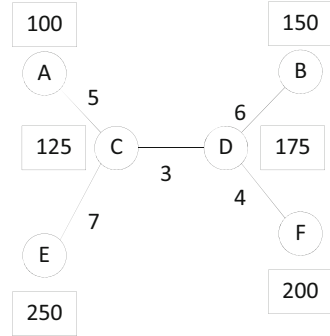
In addition, we define the following additional inputs:

d_j demand of customer j

c_{ij} unit cost of satisfying customer j from facility i.

Now suppose that no node has half or more of the total demand. We call any node that is connected to only one other node in the tree, a tip node. We let d'_j be the modified demand at node $j \in \mathcal{J}$. We also define $D_{total} = \sum_{j \in \mathcal{J}} d_j$. The algorithm is as follows.

Step 1: Let $d'_j = d_j$ for all nodes $j \in \mathcal{J}$.

Step 2: Select any tip node. Call the tip node, node A and the node to which it is connected node B. Remove node A and edge (A, B). Add the modified demand at node A to the modified demand at node B. If the new modified demand at node B equals or exceeds $D_{total}/2$, stop; node B is the 1-median of the tree. Otherwise repeat step 2.

This is clearly an $O(n)$ algorithm since Step 2 can be performed in constant time and each node is examined at most once in Step 2. The complexity of Step 1 is also clearly $O(n)$.

We can illustrate this algorithm with the tree shown in Fig. 2.1. The demand associated with each node is shown in a box beside the node and the edge distances are shown beside the edges. Nodes A, B, E and F are tip nodes. The total demand in the tree is $D_{total} = 1{,}000$. Clearly, no node has half or more of the total demand. We select node E as the first tip node to eliminate (since it has the largest demand of any tip node). We remove node E and link (C, E) from the tree and add 250 (the demand at node E) to the demand at node C. The modified demand at node C is now 375, which does not exceed half of the total demand. Next we can process node F, removing it as well as arc (D, F) and adding its demand to that of node D, resulting in a modified demand at node D of 375. Next we process node B, removing it as well as arc (B, D) and adding its demand to that of node D, resulting in a modified demand at node D of 525, which exceeds half of the total demand in the tree. Node D is therefore the 1-median of the tree.

Note that in computing the location of the 1-median we do not need to use the distances. In fact, node D would be the 1-median of the tree for any arc distances for the tree. To compute the objective function value, we clearly do need the distances. The objective function value for the 1-median located at node D in Fig. 2.1 is 5,375.

Kariv and Hakimi (1979) present an $O(n^2 p^2)$ algorithm for the p-median problem on a tree. Tamir (1996) improved the computation time and presented an $O(pn^2)$ algorithm for the problem of locating p facilities on a tree.

2.4   Model Formulation

In this section, we formulate the p-median problem. In addition to the notation defined above, we define the following additional input:

p the number of facilities to locate.

Finally, we define the following decision variables:

$$y_i = \begin{cases} 1 & \text{if a facility is located at candidate site } i \\ 0 & \text{otherwise} \end{cases}$$

x_{ij} the fraction of the demand of customer j that is supplied from facility i.

With this notation, we can formulate the p-median problem as follows:

$$\text{minimize} \quad \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} d_j c_{ij} x_{ij} \tag{2.1}$$

$$\text{subject to} \quad \sum_{i \in \mathcal{I}} x_{ij} = 1 \quad \forall j \in \mathcal{J} \tag{2.2}$$

$$\sum_{i \in \mathcal{I}} y_i = p \tag{2.3}$$

$$x_{ij} - y_i \leq 0 \quad \forall i \in \mathcal{I}; j \in \mathcal{J} \tag{2.4}$$

$$y_i \in \{0, 1\} \quad \forall i \in \mathcal{I} \tag{2.5}$$

$$x_{ij} \geq 0 \quad \forall i \in \mathcal{I}; j \in \mathcal{J}. \tag{2.6}$$

The objective function (2.1) minimizes the demand-weighted total cost. Constraints (2.2) mean that all of the demand at demand site j must be satisfied. Constraints (2.3) require exactly p facilities to be located. Constraints (2.4) state that demand nodes can only be assigned to open facilities. Constraints (2.5) stipulate that the location variables must be integer and binary. Finally, constraints (2.6) state that the assignment variables must be non-negative.

Note that we do not require the assignment variables to be binary variables. If the unit cost from a demand node to the nearest open facility is strictly less than the unit cost between that node and any other open facility, then the corresponding assignment variables for that demand node will naturally be binary. That is, all of the demand at that node will be assigned to the nearest open facility. If the unit costs between a demand node and two or more open facilities are the same, and the unit costs are less than the unit costs between the demand node and any other open facility, the assignment variables may indicate that the demand is to be split between the set of nearest facilities. We can always round all but one of these assignment variables down to 0 and round the last one up to 1 if we require all-or-nothing demand assignments or single sourcing.

2.5 Solution Heuristics for the *p*-Median Model on a General Network

In this section, we outline a number of heuristic algorithms for solving the *p*-median problem on a general network. We conclude the section by structuring a Lagrangian relaxation algorithm (Fisher 1981, 1985).

2.5.1 Basic Construction and Improvement Algorithms

The simplest algorithm is the myopic or greedy adding algorithm. In this algorithm, all candidate facility sites are examined and the one whose addition to the current solution reduces the demand-weighted total distance the most is added to the incumbent solution. The process continues until the solution includes *p* facilities. The following is pseudocode for the myopic algorithm. In this and all subsequent pseudocodes, we define $z(\mathcal{J}, X) = \sum_{j \in \mathcal{J}} d_j \min_{m \in X} \{c_{mj}\}$, where X is the current set of candidate facility sites. Note that the function depends on both the set of demand nodes to be considered and the candidate locations to be used.

Myopic Algorithm Pseudocode

1. **Set** $X \leftarrow \varnothing$. /* X is the set of locations to be used
2. **Find** $i^* = \mathrm{argmin}_{i \in \mathcal{J}} \{z(\mathcal{J}, X \cup \{i\})\}$.
3. **Set** $X \leftarrow X \cup \{i^*\}$.
4. **If** $|X| < p$, go to Step 2; else stop.

Step 1 initializes the set of locations to the empty set. Step 2 finds the best node to add to the emerging solution. Step 3 adds that site to the solution. Step 4 asks if less than *p* facilities have been added to the emerging solution. If so, the algorithm continues with Step 2; if not, the algorithm stops.

The myopic algorithm can readily paint itself into a corner. There is no guarantee of optimality for the myopic algorithm. As illustrated below in the computational results, the algorithm can perform quite poorly. That said, it is clear that it is optimal if we are locating only a single facility.

Exploiting the optimality of the myopic algorithm for the 1-median problem, Maranzana (1964) proposed a neighborhood improvement algorithm. Starting with any feasible solution to the *p*-median problem, the algorithm assigns each demand node to its nearest facility. Ties are broken arbitrarily. The set of nodes assigned to a facility constitutes the neighborhood of that facility. Within each neighborhood, the algorithm examines each candidate node and selects the one that minimizes the demand-weighted total distance among all nodes in the neighborhood. In other words, within each neighborhood, the algorithm solves a 1-median problem. If no facility locations have changed, the algorithm stops; otherwise, if any facility locations have changed as a result of solving the 1-median problem, the algorithm re-assigns all demand nodes to the nearest open facility. If no assignments have changed, the algorithm stops; otherwise, the algorithm continues by solving the 1-median problem in each neighborhood. This process of determining neighborhoods and solving 1-median problems within each neighborhood continues until no further improvement is possible. The pseudocode below outlines the neighborhood search algorithm.

Neighborhood Search Algorithm Pseudocode

1. **Input:** X /* X is a set of p facility locations
2. **Set:** $N_i \leftarrow \phi$, $\forall i \in \mathcal{I}$ /* N_i is the set of demand nodes for which
   /* candidate site i is the closest open facility
3. **For** $j \in \mathcal{J}$ **do**
4.   **Set** $i^* \leftarrow \text{argmin}_{i \in \mathcal{I}} \{c_{ij}\}$
5.   **Set** $N_{i^*} \leftarrow N_{i^*} \cup \{j\}$
6. **End For**
7. **Set** $X^{new} \leftarrow \phi$ /* X^{new} is the set of new facility locations
8. **For** $i \in \mathcal{I}$ **do**
9.   **If** $|N_i| > 0$ **then**
10.   **Find** $k^* = \text{argmin}_{k \in N_i} z(N_i, \{k\})$
11.   **Set** $X^{new} \leftarrow X^{new} \cup \{k^*\}$
12.   **End If**
13. **End For**
14. **If** $X \neq X^{new}$ **then set** $X \leftarrow X^{new}$ **and go to Step 2; else stop**

Step 1 initializes the solution with any set of p facilities. Steps 2 through 6 initialize and then set the neighborhoods. Step 7 initializes a new candidate set of facility locations. Steps 8 through 13 find the new candidate locations. In particular, in Step 10, the algorithm finds the 1-median within each neighborhood and adds that vertex to the emerging new solution in Step 11. The algorithm, as written, assumes that the sets of demand locations and candidate sites are the same. While the neighborhood search algorithm finds the optimal location within each

neighborhood, there is no guarantee that it will find the global optimum for the problem.

The exchange algorithm, proposed by Teitz and Bart (1968), is another heuristic improvement algorithm that tends to do better than the neighborhood search algorithm. The algorithm attempts to improve the current solution by removing a node that is in the solution and replacing it with a node that is not in the solution. If an exchange of this sort can be found and improves the solution (i.e., reduces the demand-weighted total distance), it is implemented. The algorithm terminates when there is no such exchange that improves the solution. The pseudocode for one variant of the exchange algorithm is shown below.

Exchange Algorithm Pseudocode

1. **Input:** X /* X is a set of p facility locations
2. **For** $i \in X$ **do**
3.   **For** $k \in \mathcal{J} \backslash X$
4.   **If** $z(\mathcal{J}, X) > z(\mathcal{J}, X \cup \{k\} \backslash \{i\})$ **then**
5.   **Set** $X \leftarrow X \cup \{k\} \backslash \{i\}$ **and stop**
6.   **End If**
7.   **End For**
8. **End For**

Step 1 initializes the solution with any set of p facilities. In Step 2 we loop over the sites in the current solution. In Step 3 we loop over candidate sites that are not in the solution. In Step 4, we ask if removing one site from the current solution and replacing it with a site not in the current solution will improve the objective function. If so, we make that substitution and the algorithm stops.

There are numerous ways of implementing an exchange algorithm. The algorithm might implement the *first* exchange that improves the solution, as shown in the pseudocode above. Alternatively, the algorithm might find the first node in the solution whose removal will result in an improvement to the solution and then find the *best* node to insert into the solution in place of the removed facility. Finally, the algorithm can find the best improving pair of nodes over all possible nodes to be removed and inserted into the solution.

If either of the first two approaches are adopted—that is, if the exchange algorithm does not find the best overall exchange possible—there are alternate ways in which the algorithm can proceed. One option is to continue the search with the next indexed node that is not in the solution, attempting to replace the node that was just inserted into the solution with another node. Another option is to proceed to the next node in the solution and attempt to find exchanges based on that node. A third option is to reinitiate the search from the first node in the solution. The various options for selecting an exchange to implement, as well as the different ways in which the algorithm can proceed once an improving exchange has been identified, result in numerous possible implementations of the exchange algorithm. Most of the literature does not identify which implementation was employed.

2.5.2   *Metaheuristics for the p-Median Problem*

The myopic algorithm is a construction algorithm. The neighborhood and exchange algorithms are improvement algorithms. A large variety of metaheuristic algorithms have been devised to find solutions to the *p*-median problem. Mladenović et al. (2007) provide a relatively recent review of these techniques. Below we highlight a few of the classic papers and approaches in this field.

Chiyoshi and Galvão (2000) present a statistical analysis of a simulated annealing algorithm (Kirkpatrick 1984) for the *p*-median model. They employed the 40-instance dataset proposed by Beasley (1990). The dataset includes instances ranging from 100 to 900 demand locations. They found that in 100 runs of a simulated annealing algorithm for each instance, the best solution found was the optimal solution in 26 of the 40 instances. The maximum deviation from optimality for the best of the 100 runs for the 40 instances was 1.62 %. Al-khedhairi (2008) also employed simulated annealing for the Beasley dataset and found the optimal solution in 33 of the cases. However, the maximum deviation was over 18 % for the seven instances for which the simulated annealing algorithm failed to find the optimal solution. Murray and Church (1996) also discuss the application of simulated annealing to the *p*-median problem as well as to the maximal covering problem.

Alp et al. (2003) propose an effective genetic algorithm (Goldberg 1989; Haupt and Haupt 1998; Holland 1975; Michalewicz 1994; Mitchell 1998) for the *p*-median problem. For the 40-instance Beasley dataset, they ran their algorithm 10 times for each instance. They found the optimal solution at least once in 28 of the 40 cases. In six of the cases, the genetic algorithm always identified the optimal solution. In the 12 cases in which the genetic algorithm failed to find the optimal solution, the best of the ten runs resulted in objective functions that deviated from the optimal value by 0.02–0.4 %.

Rolland et al. (1996) applied tabu search (Glover 1990; Glover and Laguna 1997) to the *p*-median problem. They tested their algorithm using randomly generated datasets ranging in size from 13 to 500 nodes. For instances with 100 nodes or fewer, the results were compared to two-exchange heuristics as well as to the optimal solution found using an integer programming algorithm. For the larger instances, optimal solutions were not obtained and the three heuristics were compared with each other. In all cases, the tabu search algorithm outperformed the other two heuristics. For the smaller instances (100 nodes or fewer) the tabu search algorithm averaged 0.5 % from optimality with a maximum deviation of 6 %. Tabu search found the optimal solution in 66 % of the smaller test cases. For the 12 larger test cases, tabu search found the best solution in all but one case.

If an improvement (e.g., the neighborhood search or exchange algorithm outlined above) is started with many different randomly generated solutions, the *p* facilities that are selected are often similar across the various solutions. In other words, some sites are selected in many of the runs and many other candidate sites are never selected. Using this observation, Rosing and ReVelle (1997) developed a heuristic

concentration algorithm for the p-median problem. The idea is to generate a number of good solutions based on randomized starting solutions. A subset of the nodes that are selected in the various runs is then used to reduce the number of location variables in formulation (2.1)–(2.6) above. In other words, the concentration set, or the set of candidate sites, is reduced from $\mathcal{J}$ to a smaller set consisting of a subset of the nodes selected as facilities in the various randomized runs.

Heuristic concentration is based on eliminating some of the location variables. Church (2008) proposed the BEAMR approach which attempts to eliminate some of the assignment variables. BEAMR attempts to utilize only the h_j closest assignment variables for each demand node. To ensure feasibility, the model also includes a variable for each demand node allowing the assignment to a dummy facility further than the h_j closest candidate facilities. This assignment does not need to satisfy constraints (2.4). The resulting model provides a lower bound on the objective function value for the p-median problem. An upper bound can be found by simply assigning every demand node to the nearest of the selected facility sites. If the bounds are not close enough, then some of the h_j values can be increased, particularly for those nodes for which assignment to one of the nearest h_j candidate sites was not possible. The algorithm typically results in provably optimal solutions using a fraction of the constraints and variables of the original formulation (2.1)–(2.6).

Rosing et al. (1998) compared heuristic concentration to tabu search in problems with 100 and 200 demand nodes and candidate sites. Heuristic concentration found the optimal (or best known) solution in 17 of the 21 test cases, while tabu search found the optimal (or best known) solution in only two cases.

Mladenović and Hansen (1997) introduced a variable neighborhood search algorithm. Hansen and Mladenović (1997) applied this algorithm to the p-median problem. They found that variable neighborhood search outperformed both a greedy interchange algorithm and two different tabu search-based algorithms.

Hansen and Mladenović (2001) reviewed the basics of variable neighborhood search algorithms and compared a variety of metaheuristic algorithms, including variable neighborhood search for the 12 largest of the 40 Beasley instances. They found that variable neighborhood search and heuristic concentration outperformed tabu search and a greedy interchange algorithm. Variable neighborhood search was slightly better than heuristic concentration.

2.5.3   A Lagrangian Heuristic for the p-Median Problem

In this subsection, we outline a Lagrangian relaxation algorithm to the p-median problem. The advantage of Lagrangian relaxation over any heuristic approach is twofold. First, at every iteration of the Lagrangian procedure we obtain lower and upper bounds on the objective function value. Second, the Lagrangian procedure can readily be embedded in a branch-and-bound algorithm to obtain provably optimal solutions.

We relax constraint (2.2) to obtain the following Lagrangian problem:

$$\operatorname{Max}_\lambda \operatorname{Min}_{x,y} \mathcal{L} = \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} d_j c_{ij} x_{ij} + \sum_{j \in \mathcal{J}} \lambda_j \left(1 - \sum_{i \in \mathcal{I}} x_{ij}\right)$$
$$= \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} \left(d_j c_{ij} - \lambda_j\right) x_{ij} + \sum_{j \in \mathcal{J}} \lambda_j \tag{2.7}$$

subject to (2.3)–(2.6).

For fixed values of the Lagrange multipliers, λ_j, we compute the value of being able to add a facility at node $i \in \mathcal{I}$. This value is given by $V_i = \sum_{j \in \mathcal{J}} \min\{0, d_j c_{ij} - \lambda_j\}$. We then select the p sites with the p most negative V_i values, breaking ties arbitrarily. This determines the values of the location variables, y_i. The assignment variables are determined by setting $x_{ij} = 1$ if (i) $y_i = 1$ and (ii) $d_j c_{ij} - \lambda_j < 0$, and setting $x_{ij} = 0$ otherwise. The resulting values can be used to evaluate (2.7), providing a lower bound on the objective function value. To obtain an upper bound on the objective function value, we simply assign every demand node to the nearest candidate facility for which $y_i = 1$ and evaluate (2.1) using these assignment values.

Some of constraints (2.2) are likely to be violated by the solution to the Lagrangian problem as outlined above. In particular, some demand nodes may not be assigned to any facility and some may be assigned to multiple facilities. This occurs when the Lagrange multipliers are not at their optimal values. Subgradient optimization can be used to improve the Lagrange multipliers. Daskin (2013) provides a detailed explanation of the Lagrangian algorithm for the *p*-median problem.

The Lagrange multipliers coupled with the best lower and upper bounds can be used to force candidate sites in and out of the solution at any point in the Lagrangian procedure. Typically, it is most useful to do so when the bounds are very close to each other but still differ by a small amount. Let *LB* and *UB* be the best-known lower and upper bounds, respectively. Using the Lagrange multipliers associated with *LB*, sort the V_i values so that $V_{[i]}$ is the *i*th smallest value. In other words, $V_{[1]}$ is the most negative value and $V_{[p]}$ is the last value that resulted in selecting a candidate facility site in the Lagrangian solution. Additionally, $V_{[p+1]}$ is the next largest value.

Consider a candidate site $i \in \mathcal{I}$ that is in the best-known solution. Then, if $UB < LB - V_i + V_{[p+1]}$, site $i \in \mathcal{I}$ can be forced into the solution; in other words, we can set $y_i = 1$ in all subsequent Lagrangian iterations and in any branching below the node at which this check is done (e.g., the root node of a branch-and-bound algorithm). Similarly, if site $i \in \mathcal{I}$ is not part of the best-known solution and $UB < LB - V_{[p]} + V_i$, then site $i \in \mathcal{I}$ can be forced out of the solution; in other words, we can set $y_i = 0$ in all subsequent Lagrangian iterations and in any branching below the node at which this check is done (e.g., the root node).

2.6 Computational Results

In this section, we provide sample results for some of the algorithms outlined above. We begin with Table 2.2 which shows the results of using a Lagrangian relaxation algorithm embedded in branch-and-bound for the Beasley dataset. The instances were all solved using an expanded version of the SITATION software (Daskin 2013) on a Macintosh computer running OS X version 10.8.5 with a 2.7 GHz Intel Core i7 processor and 16 GB of 1,600 MHz DDR3 memory using Parallels 7.0.15107. The average solution time was under 45 s. The longest solution time—for PMED36— was under 13 min. Seventeen of the 40 instances were solved at the root node and all but three of the instances required less than 40 branch-and-bound nodes. The average solution time is 44.9 s and the average number of branch-and-bound nodes needed is 21.5.

The second part of the table illustrates the impact of using the variable forcing rules outlined at the end of Sect. 2.5 at the end of the Lagrangian algorithm at the root node of the branch-and-bound tree. The rules are quite effective at eliminating candidate nodes; on average nearly 85 % of the candidate sites that could not be in the solution were excluded at the root node using these rules. (The number of candidate sites that could not be in the solution was equal to the total number of candidate sites minus the number of facilities). Overall, 81 % of the candidate sites were either forced in or out of the solution, on average.

Next we turn our attention to tests performed using the 500 most populous counties among the 3,109 counties in the contiguous United States. While these represent less than one sixth of the total counties, they encompass over 75 % of the population living in the contiguous United States. Great circle distances between the county centroids were employed. We used SITATION to solve the *p*-median problem for this dataset with the number of facilities increasing from 1 to 25. The solution time for each of these 25 runs was under 5 s and only two instances required branch-and-bound to obtain provably optimal solutions. In each of these two instances, only three nodes in the branch-and-bound tree needed to be explored after the root node forcing rules were employed. Figure 2.2 plots the results for five, 10, 15, 20 and 25 medians. The model locates the first five cities near the major cities of New York, Los Angeles, Dallas, Chicago and Miami. Additional facilities are then added to better serve the rest of the counties. Figure 2.3 plots the demand-weighted average distance versus the number of medians. As expected, the average distance decreases with the number of medians. Also, the marginal improvement decreases with the number of medians in this case.

Table 2.2 Lagrangian relaxation results for Beasley datasets

Dataset	# Dem	# Med.	Objective	Iterations	B&B nodes	CPU time (s)
Pmed1	100	5	5,819	1,200	1	2.94
Pmed2	100	10	4,093	3,500	9	8.92
Pmed3	100	10	4,250	2,958	7	7.70
Pmed4	100	20	3,034	1,200	1	3.06
Pmed5	100	33	1,355	1,200	1	3.03
Pmed6	200	5	7,824	5,758	19	15.09
Pmed7	200	10	5,631	1,200	1	3.08
Pmed8	200	20	4,445	1,200	1	3.09
Pmed9	200	40	2,734	4,981	15	14.73
Pmed10	200	67	1,255	1,200	1	5.31
Pmed11	300	5	7,696	1,788	3	4.81
Pmed12	300	10	6,634	5,927	19	17.3
Pmed13	300	30	4,374	1,200	1	4.80
Pmed14	300	60	2,968	1,747	3	8.70
Pmed15	300	100	1,729	1,200	1	7.94
Pmed16	400	5	8,162	8,447	29	24.55
Pmed17	400	10	6,999	9,220	29	27.89
Pmed18	400	40	4,809	1,200	1	6.55
Pmed19	400	80	2,845	1,200	1	9.30
Pmed20	400	133	1,789	2,401	5	24.50
Pmed21	500	5	9,138	1,200	1	3.70
Pmed22	500	10	8,579	13,687	39	55.86
Pmed23	500	50	4,619	1,200	1	8.64
Pmed24	500	100	2,961	3,995	10	41.42
Pmed25	500	167	1,828	4,721	11	72.44
Pmed26	600	5	9,917	5,380	15	22.25
Pmed27	600	10	8,307	2,925	7	12.53
Pmed28	600	60	4,498	1,200	1	12.30
Pmed29	600	120	3,033	1,200	1	18.81
Pmed30	600	200	1,989	2,001	4	57.55
Pmed31	700	5	10,086	6,517	19	29
Pmed32	700	10	9,297	3,212	7	15.41
Pmed33	700	70	4,700	1,200	1	19.88
Pmed34	700	140	3,013	1,200	1	33.02
Pmed35	800	5	10,400	9,680	31	47.64
Pmed36	800	10	9,934	140,011	437	767.16
Pmed37	800	80	5,057	5,754	14	97.06
Pmed38	900	5	11,060	17,905	57	107.78
Pmed39	900	10	9,423	22,018	65	136.27
Pmed40	900	90	6,128	1,200	1	32.89

Table 2.2 (continued)

| Dataset	# Dem	# Med.	No. sites forced in	No. forced out	% in	% out	% forced
Pmed1	100	5	4	94	80	99	98
Pmed2	100	10	2	79	20	88	81
Pmed3	100	10	3	71	30	79	74
Pmed4	100	20	15	75	75	94	90
Pmed5	100	33	25	59	76	88	84
Pmed6	200	5	0	161	0	83	81
Pmed7	200	10	8	189	80	99	99
Pmed8	200	20	18	178	90	99	98
Pmed9	200	40	1	71	3	44	36
Pmed10	200	67	52	116	78	87	84
Pmed11	300	5	0	280	0	95	93
Pmed12	300	10	0	257	0	89	86
Pmed13	300	30	27	267	90	99	98
Pmed14	300	60	9	160	15	67	56
Pmed15	300	100	78	178	78	89	85
Pmed16	400	5	0	336	0	85	84
Pmed17	400	10	0	327	0	84	82
Pmed18	400	40	24	314	60	87	85
Pmed19	400	80	67	307	84	96	94
Pmed20	400	133	49	163	37	61	53
Pmed21	500	5	5	495	100	100	100
Pmed22	500	10	0	397	0	81	79
Pmed23	500	50	44	444	88	99	98
Pmed24	500	100	14	308	14	77	64
Pmed25	500	167	36	191	22	57	45
Pmed26	600	5	0	542	0	91	90
Pmed27	600	10	0	539	0	91	90
Pmed28	600	60	50	496	83	92	91
Pmed29	600	120	97	450	81	94	91
Pmed30	600	200	24	131	12	33	26
Pmed31	700	5	0	639	0	92	91
Pmed32	700	10	0	645	0	93	92
Pmed33	700	70	13	603	19	96	88
Pmed34	700	140	98	459	70	82	80
Pmed35	800	5	0	684	0	86	86
Pmed36	800	10	0	478	0	61	60
Pmed37	800	80	10	610	13	85	78
Pmed38	900	5	0	780	0	87	87
Pmed39	900	10	0	707	0	79	79
Pmed40	900	90	85	805	94	99	99
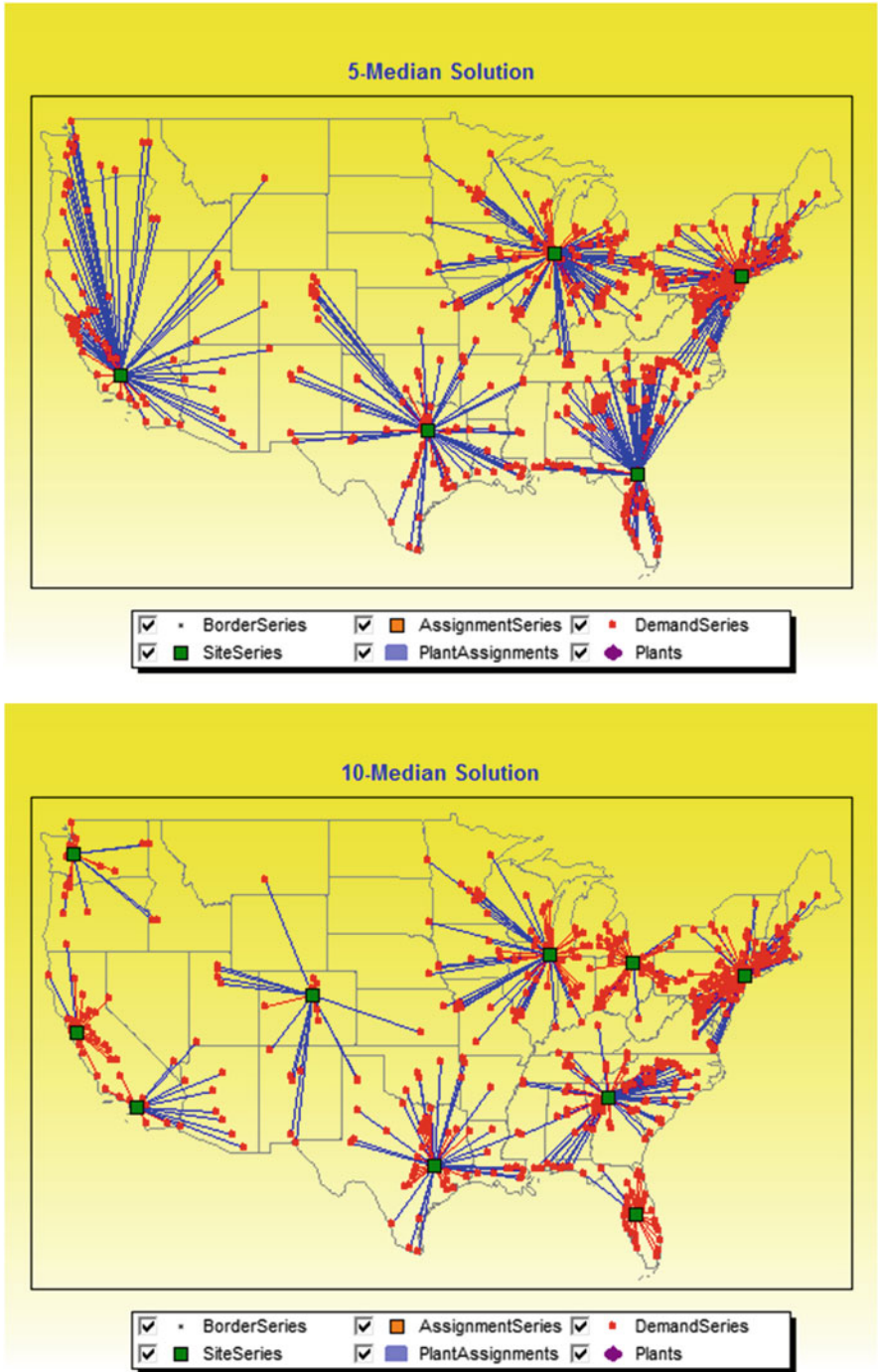
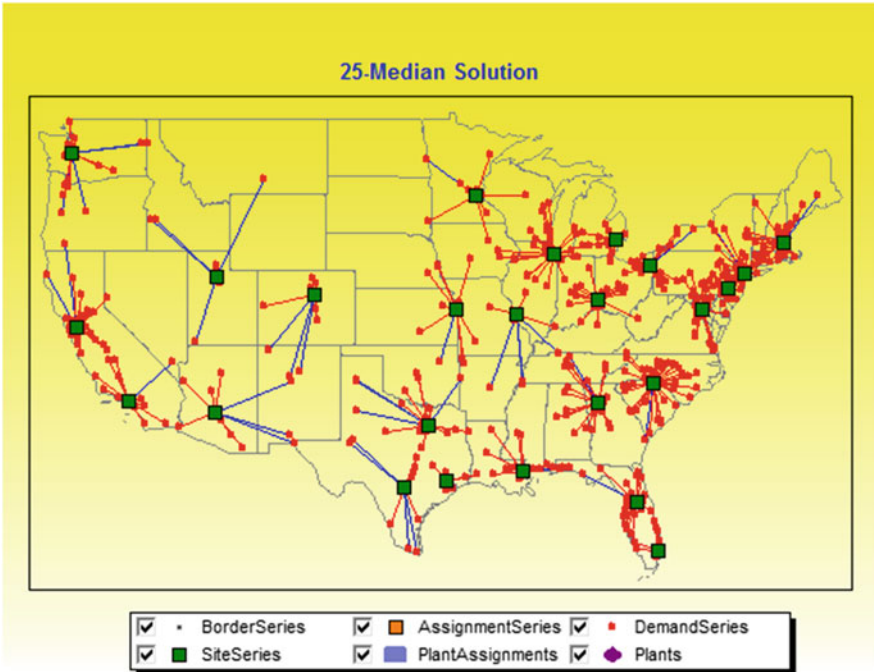**Fig. 2.2** (continued)
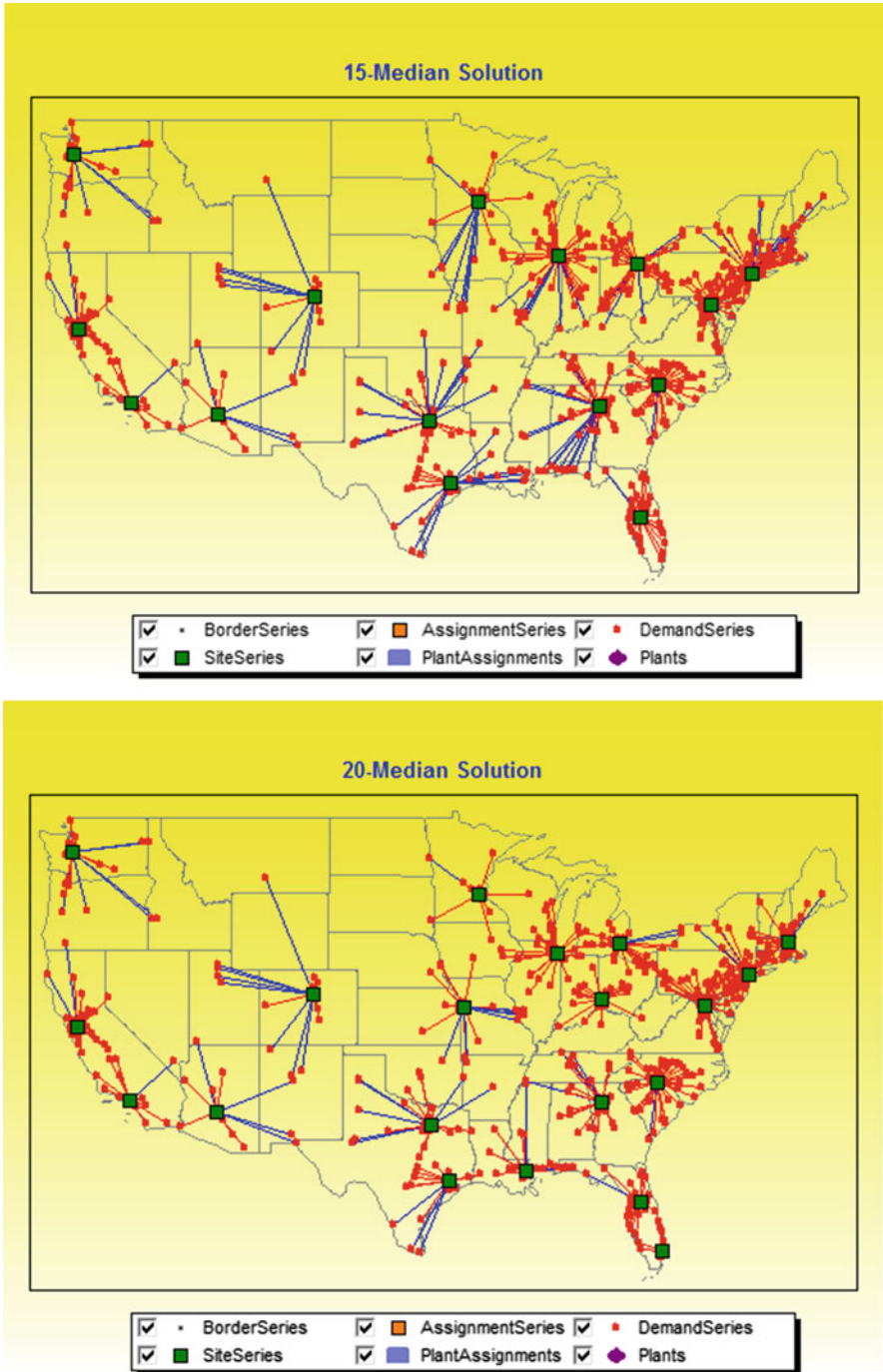
Fig. 2.2 (continued)

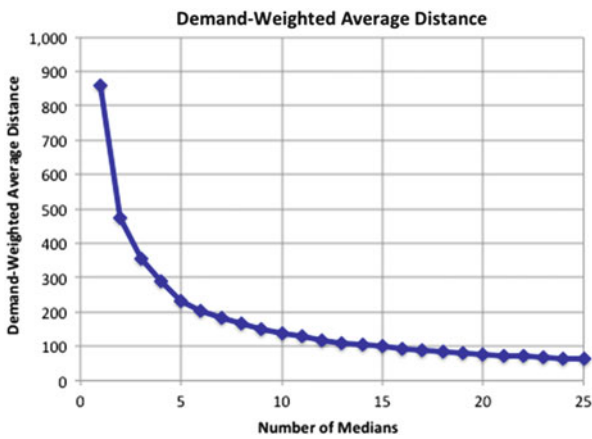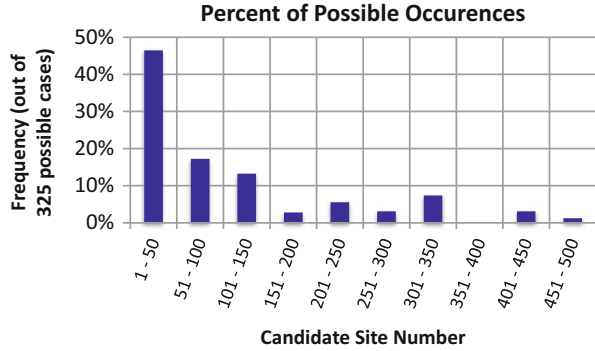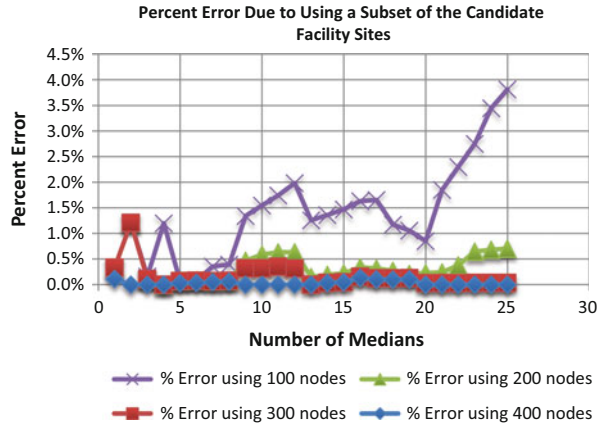**Fig. 2.2** Optimal locations for 5, 10, 15, 20 and 25 medians

Fig. 2.3 Demand-weighted average distance versus number of medians

Fig. 2.4 Histogram of the
frequency of county selection
out of 325 possible cases

Fig. 2.5 Percent error due to
limiting the candidate
node set

With 1–25 medians being selected, there conceivably could be 325 unique nodes
chosen as medians. This was not the case. Only 55 unique nodes were selected and
these were biased toward the larger demand nodes. With the dataset sorted from
the most populous to the least populous county, Fig. 2.4 plots the distribution of
the number of times nodes in different groupings were selected. Nearly half of the
counties selected were among the top 50 most populous counties. Over 75 % of the
selected counties were in the top 150 most populous counties.

Figure 2.5 plots the percent error due to limiting the candidate solution set to
the most populous 100, 200, 300 and 400 counties, compared to allowing all 500
counties to be in the solution. The errors are generally less than 1 % as long as at
least 200 nodes are in the candidate set. Even when the candidate set is limited to
only 100 nodes, the maximum error in the 25 runs was under 4 %, though the error
seems to be growing with the number of medians in this case.

**Fig. 2.6** Errors due to using various heuristic algorithms
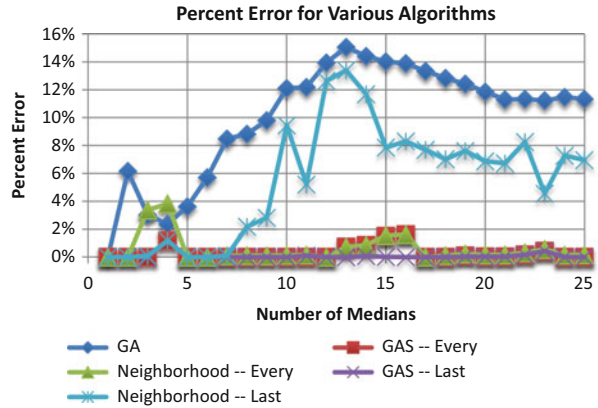
We next consider the impact of using different construction and improvement algorithms to solve the problem. Five different algorithms were tested: the greedy adding or myopic algorithm (GA), the greedy adding algorithm with the node exchange algorithm applied after every median is added (GAS-Every), the greedy adding algorithm with the neighborhood algorithm applied after every median is added (Neighborhood-Every), the greedy adding algorithm with the exchange algorithm applied after all nodes have been added (GAS-Last), and the greedy adding algorithm using the neighborhood algorithm only after all nodes have been added to the solution (Neighborhood-Last).

Figure 2.6 plots the results. Both the greedy adding algorithm (GA) and the Neighborhood algorithm applied after all nodes have been added to the solution (Neighborhood-Last) result in large errors, often exceeding 10 %. The other three algorithms perform much better and result in errors that are under 4 % and often under 2 %.

Figure 2.7 plots the results of using a genetic algorithm similar to that proposed by Alp et al. (2003). The variant employs a standard crossover operator. To ensure feasibility of the solution generated by the crossover operator, we randomly drop nodes from any solution that has more than *p* facilities (always retaining facilities that are in both parent's solutions) and randomly add facilities from the parents when the operator results in fewer than *p* facilities being selected. The standard genetic algorithm can result in large errors and the errors seem to grow with the number of medians. However, if the final genetic algorithm solutions are subject to an exchange algorithm, the errors are under 1 % and average under 0.1 % for the 25 cases.

Fig. 2.7 Errors due to using a genetic algorithm
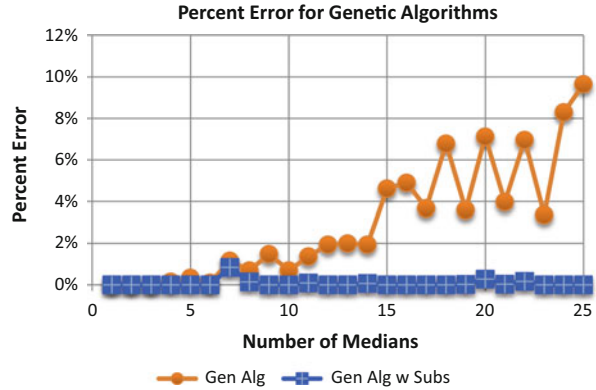
2.7 Multi-objective Extensions of the *p*-Median Model

The formulation above, (2.1)–(2.6), can be modified to obtain a formulation of the maximum covering problem (Church and ReVelle 1974). The maximum covering problem finds the location of p facilities to maximize the number of demand nodes that are covered within some coverage distance, d_c. In particular, we let d_{ij} be the distance between candidate site $i \in \mathcal{I}$ and demand node $j \in \mathcal{J}$. We then define

$$\widehat{c}_{ij} = \begin{cases} 0 & \text{if } d_{ij} \leq d_c \\ 1 & \text{if } d_{ij} > d_c. \end{cases}$$

If we now solve (2.1)–(2.6) with c_{ij} replaced by $\hat{c}_{ij}$, we will be able to solve a maximum covering problem. In essence, we are minimizing the total number of uncovered demands, which is equivalent to maximizing the number of covered demands.

We can also find the tradeoff between the covering and average cost (or average distance) objective by minimizing a suitable linear combination of the two cost terms. In particular, we minimize a weighted sum $\tilde{c}_{ij} = \alpha c_{ij} + (1 - \alpha) \hat{c}_{ij}$ of the original c_{ij} and the coverage term $\hat{c}_{ij}$, with $0 \leq \alpha \leq 1$. Clearly, if $\alpha = 1$, the model will simply minimize the demand weighted total distance or cost. Also, if $\alpha = 0$, the model will minimize the number of uncovered demands.

The choice of α is critical if we want to trace out the complete tradeoff curve. Many researchers and practitioners simply solve the problem for fixed values of α. For example, they might solve the problem for $\alpha = 0, 0.05, 0.1, \ldots, 1.0$. We do not recommend this approach because it is simultaneously likely to miss important points on the tradeoff curve and to result in obtaining many identical solutions.

Instead, one should solve the problem using $\alpha_1 = 1 - \epsilon_1$, where $\epsilon_1 > 0$ is a suitably small value so that we are guaranteed to get one of the (possibly) alternate optima for the *p*-median problem. Let Z^1 be the objective function value we obtain and let $D^1 = \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} d_j c_{ij} x_{ij}$ be the demand-weighted total distance corresponding to this solution and $U^1 = \sum_{i \in \mathcal{I}} \sum_{j \in \mathcal{J}} d_j \hat{c}_{ij} x_{ij}$ be the total uncovered demand corresponding to this solution. Next, solve the problem with $\alpha_2 = \epsilon_2$, where $\epsilon_2 > 0$ is a suitably small value such that we are guaranteed to get one of the (possibly) alternate optima for the maximum covering problem. Let Z^2 be the corresponding objective function value and let D^2 and U^2 be the demand-weighted total distance and uncovered demand corresponding to this solution. We then solve $\alpha_3 D^1 + (1 - \alpha_3) U^1 = \alpha_3 D^2 + (1 - \alpha_3) U^2$ for α_3. This results in $\alpha_3 = (U^2 - U^1)/(D^1 - D^2 + U^2 - U^1)$. We then use this value of α to weight the two objectives. This will either result in a new solution being found with a demand-weighted total distance of D^3 and uncovered demand U^3, or the solution will return one of the two original solutions on the tradeoff curve. If a new solution is found, the procedure continues by exploring the region between solution 1 and solution 3 (i.e., using $\alpha_4 = (U^3 - U^1)/(D^1 - D^3 + U^3 - U^1)$) and then between solution 3 and solution 2. If no new solution is found, then no new solution can be identified between solutions 1 and 2. This process continues until all adjacent solutions have been explored in this manner. As a final note, we observe that this is the weighting method, which will fail to find the so-called duality gap solutions (Cohon 1978).

The tradeoff between the demand weighted total distance and the maximum distance—the *p*-center objective—can also be found using formulation (2.1)–(2.6) if we suitably modify the distance (or cost) matrix, assuming all distances are integer valued. (This is not an overly restrictive assumption since we can approximate any real distances by integer values. For example, if we need distances accurate to the nearest 0.01 mile (or about 50 ft) we just multiply all distances by 100 and round the resulting values.) We do so by initially solving the problem as formulated, letting c_{ij} be the distance between demand node $j \in \mathcal{J}$ and candidate location $i \in \mathcal{I}$. We record the maximum distance, D^0_{max}. We then modify the distance matrix so that $c^{new}_{ij} = \begin{cases} c_{ij} \text{ if } c_{ij} < D^0_{max} \\ M \text{ if } c_{ij} \geq D^0_{max} \end{cases}$, where M is a very large number. We then resolve formulation (2.1)–(2.6) replacing the original costs or distances c_{ij} by c^{new}_{ij}. If M is sufficiently large, the new solution will not entail assignments with distances greater than or equal to D^0_{max}. Let $D^1_{max} < D^0_{max}$ be the new maximum distance. The process continues in this manner until no feasible solution can be found, indicating that the final value of D_{max} that was obtained is the solution to the *p*-center problem. While

Fig. 2.8 Sample tradeoff
between average distance and
percent covered

PERCENT COVERAGE VS. AVERAGE DISTANCE
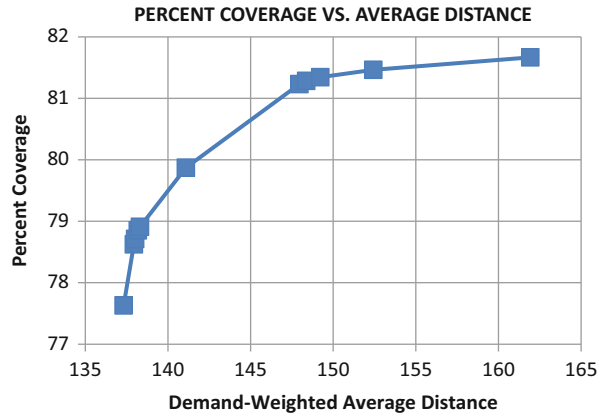
this approach seems to also be a weighting approach since we are assigning a large weight to any distance greater than or equal to the most recently found maximum distance, it is really the constraint method (Cohon 1978) since we are precluding the assignment of demand nodes to facilities that are too far away. This approach will find all non-dominated solutions.

We close this section by illustrating these two multi-objective problems. Figure 2.8 plots the tradeoff between the average distance and the percent of the demand covered within 200 miles using ten facilities with demand represented by the 500 most populous counties of the contiguous United States. The maximum covering solution results in nearly an 18 % increase in the average distance from 137.32 to 161.93 miles, while increasing the percent covered by approximately 4 %. Obtaining the 12 solutions shown in the figure took under 10 min of solution time.

Figure 2.9 is a sample center-median tradeoff curve using the 250 most populous counties in the contiguous US. While this is under 10 % of the counties, it still encompasses over 61 % of the total population in the contiguous US. The algorithm above found 22 solutions (shown with squares and a solid line), only nine of which (shown with circles and a dashed line) could be found using a weighting method. The average distance ranges from about 125 miles to 152 miles, while the maximum distance ranges from a low of 349 miles to a high of 553 miles. Several good compromise solutions are clearly shown at the bend in the curve. Figure 2.10 is an example of one such compromise solution. Obtaining the 22 solutions shown in the figure took nearly 16 h of computing time.
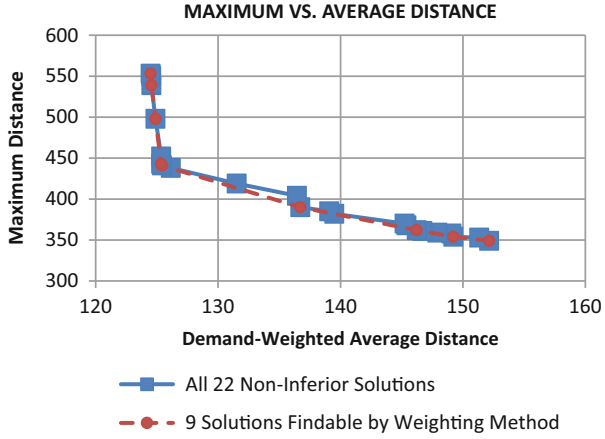
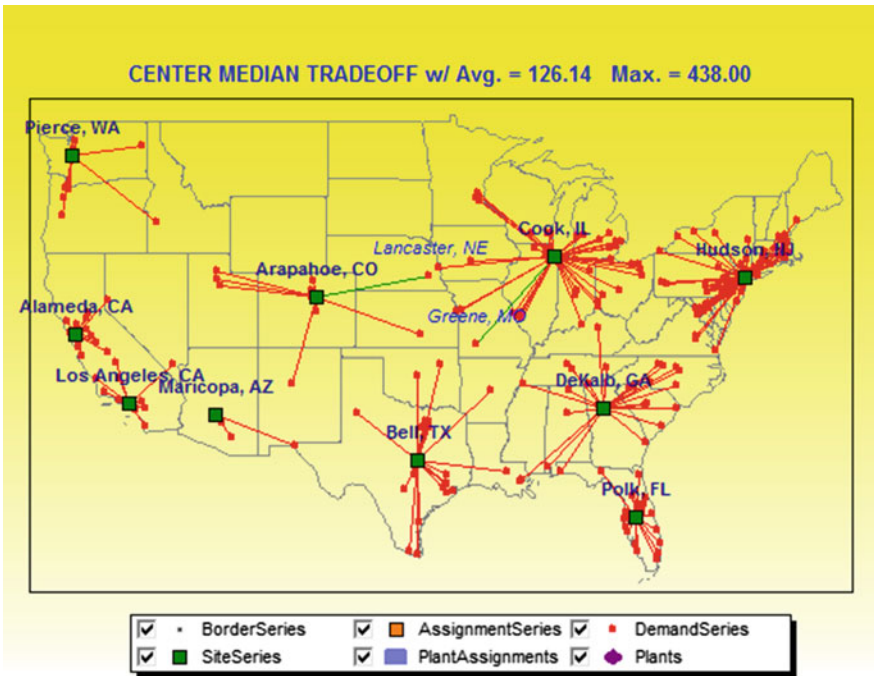**Fig. 2.9** Sample center-median tradeoff

Fig. 2.10 Sample compromise center-median solution

2.8 Conclusions

The p-median problem is central to much of discrete location theory and modeling. This chapter has outlined several important model properties and has reviewed a classic formulation of the problem. While the problem is $\mathcal{NP}$-hard on a general graph, it can be solved in polynomial time on a tree. We summarized a linear-time algorithm for the 1-median on a tree and cited results for the general p-median problem on a tree. The chapter then presented classic construction and improvement algorithms for the p-median problem and pointed the reader to literature on a number of modern heuristic algorithms that have been employed in solving the problem on general graphs. Computational results were presented for both the classical Beasley datasets as well as a 500-node instance based on the most populous counties in the contiguous United States. A well-constructed Lagrangian algorithm embedded in a branch-and-bound algorithm can solve problem instances with up to 1,000 demand nodes and 1,000 candidate sites in reasonable time. (For $p = 1$ the myopic algorithm—which amounts to total enumeration in this case—will find provably optimal solutions.) Larger problem instances may require the use of heuristic algorithms such as tabu search or simulated annealing.

The chapter concluded with two multi-objective extensions of the p-median problem. The first examines the tradeoff between the p-median objective and the maximum covering objective, while the second explores the tradeoff between the p-median objective and the p-center objective. For small instances it is often possible to solve bi-objective problems using extensions of the Lagrangian algorithm outlined above. For larger instances, using a genetic algorithm is often advisable since the population of solutions in a genetic algorithm automatically gives an initial approximation of the non-dominated set of solutions.

References

Al-khedhairi A (2008) Simulated annealing metaheuristic for solving P-median problem. Int J Contemporary Math Sci 3:1357–1365

Alp O, Erkut E, Drezner Z (2003) An efficient genetic algorithm for the p-median problem. Ann Oper Res 122:21–42

Beasley JE (1990) OR-library: distributing test problems by electronic mail. J Oper Res Soc 41:1069–1072

Chiyoshi F, Galvão RD (2000) A statistical analysis of simulated annealing applied to the p-median problem. Ann Oper Res 96:61–74

Church RL (2008) BEAMR: an exact and approximate model for the p-median problem. Comput Oper Res 35:417–426

Church RL, ReVelle CS (1974) The maximal covering location problem. Pap Reg Sci Assoc 32:101–118

Cohon JL (1978) Multiobjective programming and planning. Academic, New York

Daskin MS (2013) Network and discrete location: models, algorithms and applications, 2nd edn. Wiley, New York

Fisher ML (1981) The Lagrangian relaxation method for solving integer programming problems. Manag Sci 27:1–18

Fisher ML (1985) An applications oriented guide to Lagrangian relaxation. Interfaces 15:10–21

Glover F (1990) Tabu search: a tutorial. Interfaces 20:74–94

Glover F, Laguna M (1997) Tabu search. Kluwer Academic Publishers, Boston

Goldberg DE (1989) Genetic algorithms in search, optimization and machine learning. Addison-Wesley, Reading

Goldman AJ (1971) Optimal center location in simple networks. Transp Sci 5:212–221

Hakimi SL (1964) Optimum location of switching centers and the absolute centers and medians of a graph. Oper Res 12:450–459

Hakimi SL (1965) Optimum distribution of switching centers in a communication network and some related graph theoretic problems. Oper Res 13:462–475

Hansen P, Mladenović N (1997) Variable neighborhood search for the *P*-median. Locat Sci 5: 207–226

Hansen P, Mladenović N (2001) Variable neighborhood search: principles and applications for the *p*-median. Eur J Oper Res 130:449–467

Haupt RL, Haupt SE (1998) Practical genetic algorithms. Wiley, New York

Holland J (1975) Adaption in natural and artificial systems. The University of Michigan Press, Ann Arbor

Kariv O, Hakimi SL (1979) An algorithmic approach to network location problems. II: the *p*-medians . SIAM J Appl Math 37:539–560

Kirkpatrick S (1984) Optimization by simulated annealing: quantitative studies. J Stat Phys 34:975–986

Maranzana FE (1964) On the location of supply points to minimize transport costs. Oper Res Q 15:261–270

Michalewicz Z (1994) Genetic algorithms $+$ data structures $=$ evolution programs, 2nd edn. Springer, Berlin

Mitchell M (1998) An introduction to genetic algorithms. MIT Press, Cambridge

Mladenović N, Hansen P (1997) Variable neighborhood search. Comput Oper Res 24:1097–1100

Mladenović N, Brimberg J, Hansen P, Moreno-Perez JA (2007) The *p*-median problem: a survey of metaheuristic approaches. Eur J Oper Res 179:927–939

Murray AT, Church RL (1996) Applying simulated annealing to location-planning problems. J Heuristics 2:31–53

ReVelle CS, Eiselt HA, Daskin MS (2008) A bibliography of some fundamental problem categories in discrete location science. Eur J Oper Res 184:817–848

Rolland E, Schilling DA, Current JR (1996) An efficient tabu search procedure for the *p*-median problem. Eur J Oper Res 96:329–342

Rosing KE, ReVelle CS (1997) Heuristic concentration: two stage solution construction. Eur J Oper Res 97:75–86

Rosing KE, ReVelle CS, Rolland E, Schilling DA, Current JR (1998) Heuristic concentration and tabu search: a head-to-head comparison. Eur J Oper Res 104:93–99

Tamir A (1996) An $O(pn^2)$ algorithm for the *p*-median and related problems on tree graphs. Oper Res Lett 19:59–64

Teitz MB, Bart P (1968) Heuristic methods for estimating generalized vertex median of a weighted graph. Oper Res 16:955–961