Optimal Location of Discretionary Service Facilities

by

Oded Berman, Richard C. Larson, and Nikoletta Fouska

OR 231-90 November 1990

Abstract

Automatic teller machines and gasoline service stations are two examples of a growing number of "discretionary service facilities." In consuming service from these facilities, a significant fraction of customers do so on an otherwise preplanned trip (e.g., on the daily commute to and from work). A system planner, in determining the best locations of such facilities, is more concerned with placing the facilities along paths of customer flow rather than, say, near the center of a cluster of residences or work places. We formally model this problem and present a method for determining the optimal locations of m discretionary service facilities so as to intercept the maximum possible potential customer flow. We also show how to determine the minimal number of facilities required to intercept a prespecified fraction of total customer flow. Computational results are included.

Introduction

Finding the optimal locations of service facilities on a transportation network has been a major concern of researchers since the pioneering work of Hakimi [1, 2]. Hakimi defined the m-median and m-center problems and proposed algorithms for their solution. The vast majority of location theory research in the past 25 years has built from this foundation.

Central to the Hakimi class of location models is a simple behavioral assumption: the individual consuming service at a facility travels from his (her) home or place of work on a one-stop tour to the service facility, consumes the service and returns home or to the work place; or, equivalent mathematically, the service provider travels on a one-stop tour from his (her) service facility location to the customer's location, provides the service there, and then returns directly back to the facility. The "customer traveling" assumption is often reasonable for major services activities such as automobile purchases, weekly grocery shopping, and going to a bank to apply for a loan. The "server traveling" assumption is often reasonable for ambulances, certain emergency repair services and certain home delivery services. Thus the median problem, which minimizes the average travel distance (time) over all customers consuming service, and the center problem, which minimizes the worst case travel distance (time), have been successfully employed to locate optimally automobile dealerships, bank branch locations, major retail outlets, ambulances and home locations for certain other "dispatched" mobile services.

But an increasing number of services are consumed in a discretionary way, in which the customer (consumer) does not embark on a one-stop tour solely to consume the service. More likely for these services the customer is already on a

preplanned trip from "A" to "B" and, if (s)he passes by one or more discretionary service facilities on his (her) preplanned route, (s)he may consume that service. Examples are automobile service stations ("gasoline stations"), automatic teller machines, and "convenience" stores. We recognize that on occasion a customer may embark on a one-stop tour to consume one of these services. But, we believe that a more compelling behavioral model for the majority of discretionary service purchases is based on a model of consumer "flow" due to preplanned trips. If this model is valid, then the provider of a discretionary service would not resort to a Hakimi class (one-stop tour) model to locate optimally service facilities, but rather would attempt to locate facilities to maximize the flow of different potential customers that pass his organization's facilities.

This is the motivation for the model and algorithms developed herein. We assume we know traffic levels (e.g., number of potential customers per day) traveling between all origin-destination node pairs on a network. This in turn gives us flow rates on each branch and through each node of the network. The objective is to locate m discretionary service facilities on the network so as to maximize the flow of (different) potential customers that pass at least one of the service facilities. (We do not value multiple passings: a customer traveling past three facilities is counted as one "captured" customer in our model).

In the remainder of the paper, we provide two equivalent formulations of the model, propose a "greedy" heuristic which has been found to provide near optimal solutions in the vast majority of test problems, develop a branch-and-bound exact procedure, apply the methodology to determining the number of facilities necessary to intercept a prespecified fraction of flow, and discuss computational results.

2. The Problem

Let G(N, A) be a transportation network where N is the set of nodes with cardinality n and A is the set of arcs. P is the set of non-zero flow paths through the network arcs and nodes and f_p is the rate (per unit time) of flow travel along any path $p \in P$. The total network flow is $f = \sum_{p \in P} f_p$. We let m be the number of facilities to be located on the network. All the facilities are assumed to provide identical service and thus no customer needs on any given trip to stop at more than one of them. It is therefore crucial to avoid double counting of the common flow past two or more facilities. Also, traffic flow on paths is assumed to be independent of service facility locations.

The objective of this paper is to find a set of locations for the m facilities on the network so as to maximize the total flow intercepted by the facilities excluding any double counting.

Before proceeding to formulations of the problem we state a very useful result:

Theorem 1: An optimal set of locations exists on the nodes of the network.

This theorem is easy to prove. We therefore provide just an informal reasoning for its validity. Suppose a facility is located on the interior of an arc (a, b). Then, by moving this location to either a or b the same flow must be intercepted since any flow along (a, b) passes through both a and b. Moreover the facility in the new location might intercept additional flow as well. Therefore the new solution, created by moving the facility to a node will result in an objective function value greater than or equal to the old one.

Due to Theorem 1, we can reduce the search from an infinite set to a finite set of feasible solutions, namely the possible of m out of n combinations of network nodes.

2.1. Problem P1: All Integer

We present two formulations of the problem, the first involving two sets of integer variables:

$$x_j = \begin{cases} 1 & \text{if there is a facility located on node j} \\ 0 & \text{otherwise,} \end{cases}$$

$\forall j \in N$, and

$$y_p = \begin{cases} 1, & \text{if at least one of the facilities located by a particular feasible solution is on path p} \\ 0, & \text{otherwise,} \end{cases}$$

$\forall p \in P$.

The problem can be formulated as an integer linear problem

(P1)

$$\max \sum_{p \in P} f_p y_p$$

$$\text{s. t.} \quad \sum_{j=1}^{n} x_j = m \qquad \textbf{(1)}$$

$$\sum_{j \in p} x_j \geq y_p, \quad \text{for all } p \in P \qquad \textbf{(2)}$$

$$x_j = 0, 1 \qquad y_p = 0, 1$$

The second set of constraints ensures that y_p will be zero if all the x_j for $j \in p$ are zero, that is , if there is no facility located on path p. On the other hand, if there is at least one facility located on path p, at least one x_j will be one and therefore, y_p could take on the value of zero or one. But, since ours is a maximization problem, y_p will be forced to be equal to one. Since y_p can only be 0 or 1, there is no "double counting" when the same flow passes more than one facility. If we assume that only single (e.g., shortest) paths between origins and destinations of flow are allowed, then the number of constraints is $n^2 + 1 - n$, and the number of variables is n^2. However, our formulation allows any subset of network paths to have positive flow. We note that links do not have any significance in the formulation as long as specific paths are specified for the flows in the network.

It is interesting to observe that the formulation of the problem above is identical to that of another problem in location theory which is called the Maximal Covering Location Problem (MCLP) [3]. In contrast to our problem where the total path flow covered is maximized, in the MCLP the objective is to find a set of m locations on a network that maximizes the total nodal demand that is (covered) within a prespecified distance from the closest facility. In [3] the search for the optimal set of locations is restricted to the set of nodes. In a later paper [4] it is shown that at least one optimal set of locations exists that is composed entirely of points belonging to a finite set of points called the Network Intersect Point set that include non nodal points of the network. It is when the MCLP is restricted to nodes (or when artificial nodes are created at the Network Intersect Points) that the formulation of our problem and of the MCLP are identical.

As mentioned in [5] the maximum coverage location problem is obviously NP-hard on a general network. Therefore our problem is NP-hard as well. In the MCLP

the set of constraints (2) correspond to the n nodes of the network. Therefore our problem is still NP-hard on general networks even for the case when paths are restricted to be the shortest paths between 0-D pairs (number of constraints (2) is $n^2 - n+1$). Reference [5] includes a polynomial algorithm to solve the MCLP problem on a tree network.

2.2. Problem P2: Relaxed

The linear formulation (P1) can be recast into an equivalent nonlinear formulation (P2) with far fewer variables and constraints. We allow fractional facilities by defining

x_j = fraction of a facility located at node j, $j \in N \quad 0 \le x_j \le 1$.

Define b_j to be the flow that passes through node j,

$$b_j \equiv \sum_{p \in P_j} f_p \qquad j = 1, 2, \ldots, n,$$

where $P_j \subset P$ is the set of all paths that contain node j. If flow b_j is passing through node j, then we assume that the flow intercepted at node j is $b_j x_j$.

Suppose we have along path $i \rightarrow k$, flow f_p that encounters along the path fractional facilities x_1, x_2 and x_3. After first flowing through node 1, $f_p x_1$ represents the intercepted flow. The fractional facility at node 2 intercepts an additional $f_p(1 - x_1) x_2$ units of flow. Finally, the fractional facility at node 3 intercepts an additional $f_p(1 - x_1 - (1 - x_1) x_2) x_3$ units of flow. Total flow intercepted along this path is $f_p [1 - (1 - x_1) (1 - x_2) (1 - x_3)]$. Note that if any x_j (j = 1, 2, 3) is equal to one, then all the flow f_p is intercepted; if all x_j's are equal to zero, then no flow is intercepted. There is no "double counting" of intercepted flows; for instance, 2 or 3 x_j's could be equal to one, and still the intercepted flow is correctly evaluated as f_p.

The product term $(1 - x_1)(1 - x_2)(1 - x_3)$ is intuitive. At each node j a fraction $(1 - x_j)$ of the flow is not intercepted. The total fraction of flow along the $i \rightarrow k$ path not intercepted is $(1 - x_1)(1 - x_2)(1 - x_3)$. The complementary fraction, $1 - (1 - x_1)(1 - x_2)(1 - x_3)$, represents the fraction intercepted by at least one facility.

P2 can now be stated succinctly,

$$\max \sum_{p \in P} f_p \left(1 - \prod_{n \in p} (1 - x_n)\right) \qquad \textbf{(P2)}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} x_j = m$$

$$0 \leq x_j \leq 1.$$

To demonstrate that this formulation is equivalent to (P1), we must prove that this relaxed formulation (i.e., no x_j integer constraint) produces integer ("whole facility") results and that the two formulations have the same objective function value when the solutions are all integer.

Theorem 2. An optimal solution to P2 exists that is entirely integer, i.e., $x_j = 0$ or 1 for all $j = 1, 2, \ldots, n$.

Proof. Suppose we have a noninteger solution $(x_1, x_2, \ldots, x_n)$ such that $\sum_{j=1}^{n} x_j = m$. Then at least two x_j's, say x_a and x_b, are fractional: $0 < x_a < 1$, $0 < x_b < 1$. Define $c_{ab} = x_a + x_b$, $0 < c_{ab} < 2$. Partition the flow paths into four sets, $P = P_{ab} + P_{ab'} + P_{a'b} + P_{a'b'}$, where P_{ab} is the set of paths containing both nodes a and b. $P_{a'b'}$ is the set of paths containing neither node a or b, and $P_{ab'}$ and $P_{a'b}$ each contain one but not the other node. Replacing x_b with $(c_{ab} - x_a)$ in the objective function, and holding $(n - 2)$ x_j's constant $(j \neq a, b)$, we can write for the total intercepted flow

$$F = F(x_a) = \sum_{p \in P_{a'b'}} f_p \left(1 - \prod_{n \in p} (1 - x_n)\right) + \sum_{p \in P_{ab'}} f_p \left(1 - (1 - x_a) \prod_{\substack{n \in p \\ n \neq a}} (1 - x_n)\right)$$

$$+ \sum_{p \in P_{a'b}} f_p \left(1 - (1 - c_{ab} + x_a) \prod_{\substack{n \in p \\ n \neq b}} (1 - x_n)\right)$$

$$+ \sum_{p \in P_{ab}} f_p \left(1 - (1 - x_a)(1 - c_{ab} + x_a) \prod_{\substack{n \in p \\ n \neq a, b}} (1 - x_n)\right).$$

Taking the partial derivatives of $F(x_a)$ with respect to x_a we obtain

$$\frac{\partial F(x_a)}{\partial x_a} = \sum_{p \in P_{ab'}} f_p \prod_{\substack{n \in p \\ n \neq a}} (1 - x_n) - \sum_{p \in P_{a'b}} f_p \prod_{\substack{n \in p \\ n \neq b}} (1 - x_n)$$

$$+ \sum_{p \in P_{ab}} f_p \, (2x_a - c_{ab}) \prod_{\substack{n \in p \\ n \neq a, b}} (1 - x_n)$$

$$= \alpha_1 + \alpha_2 x_a, \text{ where } -\infty < \alpha_1, < \infty, \ \alpha_2 \geq 0.$$

$$\frac{\partial^2 F(x_a)}{\partial x_a^2} = \alpha_2 \geq 0.$$

Since the second derivative of $F(x_a)$ is nonnegative, maximization of $F(x_a)$ will always occur at a boundary point, i.e., either $x_a = 0$ or $x_a = 1$. Repeat the above process up to $n - m - 1$ times, each time driving at least one x_j to integer and improving (or not decreasing) the objective function. ■

There may be two or more all integer optimal solutions to P2. A noninteger solution can be optimal only if the flows intercepted directly by each of the fractional facilities are equal and no flows are shared between fractional facilities; but in such a case, there also exists two or more optimal all integer solutions.

Lemma 1. The formulations P1 and P2 have the same objective function value when the solution to P2 is all integer.

Proof. Suppose we have a feasible solution to P2 having $x_j = 0, 1$ and

$$\sum_{j=1}^{n} x_j = m.$$

Then for any path $p \in P$ having at least one $x_n = 1, n \in p$, the corresponding flow f_p is included in the objective function. Hence the value of the objective function for P2 is the sum of path flows f_p in which each path p contains at least one facility, i.e., $x_n = 1$ for at least one $n \in p$.

But for each such path we must have

$$\sum_{j \in p} x_j \geq 1,$$

implying in P1 that $y_p = 1$ and hence that the value of the objective function in P1 is the sum of path flows for paths containing at least one facility.

In the next section, we present a very simple and efficient "greedy" heuristic to solve the problem that is very similar to the "greedy" heuristic to solve the MCLP [3]. This heuristic can be also used to provide an initial good solution for the branch-and-bound algorithm that is presented in Section 4.

3. The Greedy Heuristic

The main idea of the greedy heuristic is to locate the facilities sequentially at nodes which intercept most of the unintercepted flow that remains in the network. The heuristic locates the first facility at the node which intercepts the maximum flow in the network. Afterwards, it removes the flow intercepted by the previously located facility and locates the second one at the node which intercepts most of the residual flow. The process is repeated until either all the m facilities are located or no unintercepted flow remains in the network.

3.1 The Algorithm

Recall that b_j is the flow intercepted at node j. The steps of the heuristic are as follows:

Step 1: $l = 1$.

Step 2: Compute $b_j \; \forall \, j \in N$.

Step 3: Find $b_{j_{max}} = \max_{j \in N} \{b_j\}$; locate facility l at node j_{max} (in a case of a tie, choose j_{max} to be the node with the smallest index j).

Step 4: Delete from P the set $P_{j_{max}}$ and delete from N node j_{max} (i.e., $P = P - P_{j_{max}}$, $N = N - j_{max}$).

Step 5: If $l = m$ or if $P = \emptyset$, stop. Otherwise, set $l = l + 1$ and return to Step 2.

By the very nature of its "greediness," one easily sees that the heuristic's solutions demonstrate "diminishing returns to scale," i.e., incrementally less new flow intercepted with each new facility located. More formally, if $\bar{f}(m)$ is the total

flow intercepted by the greedy heuristic with m facilities located, m = 0, 1, 2, . . ., the function $\bar{f}(m)$ is monotone nondecreasing concave.

3.2. Demonstration of Non Optimality

The greedy heuristic does not guarantee an optimal solution. To show this let us refer to the simple example in Figure 1, where $P = \{p_1, p_2, p_3, p_4\}$, $m = 2$, $N = \{A, B, C, D, E, F\}$, $p_1 = A - C - D$, $p_2 = B - C - D$, $p_3 = A - E$, $p_4 = B - F$, $f_{p1} = f_{p2} = 1 + \varepsilon$ and $f_{p3} = f_{p4} = 1 - \varepsilon$.

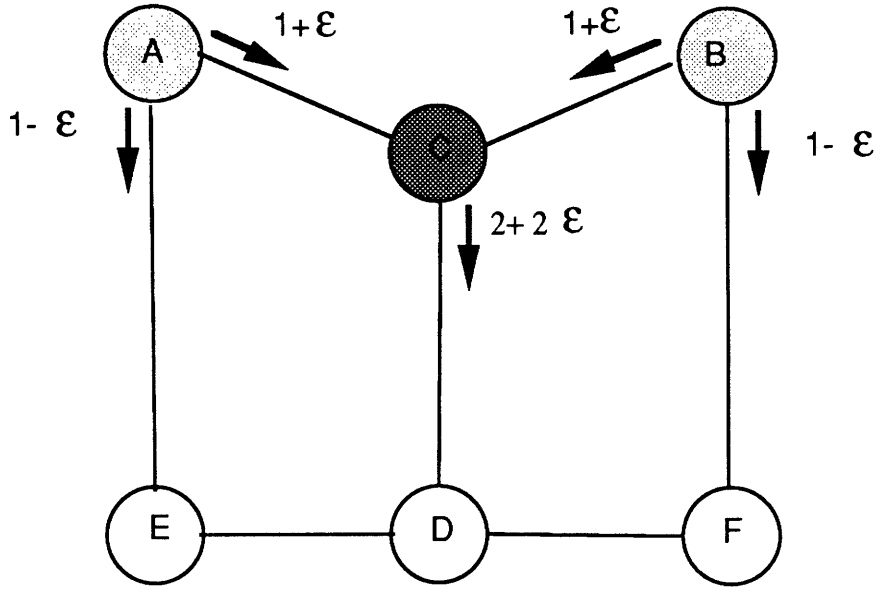

Figure 1: The Network for the Counter Example

Since $b_A = b_B = 2$, $b_E = b_F = 1 - \varepsilon$, $b_C = b_D = 2 + 2\varepsilon$, we select node C (or D) as the location of the first facility. Now $P = \{p_3, p_4\}$, $N = \{A, B, D, E, F\}$, $b_A = b_B = b_E = b_F = 1 - \varepsilon$, $b_D = 0$ and therefore if we locate the second facility in A (or B or E or F), $P = \emptyset$

and thus the algorithm stops with the solution (C, A) and an objective function value of $2 + 2\varepsilon + 1 - \varepsilon = 3 + \varepsilon$. However it is easy to verify that the optimal solution is (A, B) with an objective function value of 4.

It is interesting to note that in a case of nodal interception ties, the choice in Step 3 can influence the result of the algorithm. To see this let us refer back to the previous example with the only difference that now $f_{p_2} = 1 - \varepsilon$ and $f_{p_4} = 1 + \varepsilon$. In the first iteration of the algorithm, there occurs a tie among nodes A, B, C, and D for the first location. If we choose node A(B) as the first facility, then we would have to locate the second one at node B(A) and therefore obtain the optimal solution, whose objective function value is again 4 units of flow. But if we start with node C or D, we would locate the second facility at node B and would therefore, obtain a solution for which the objective function becomes $[3 + \varepsilon]$ units, as opposed to the optimal 4 units of flow.

3. Nonconcavity of the Optimal Solution

The greedy heuristic produces solutions $\bar{f}(m)$ that are concave in the number of facilities m. However, the optimal solution $f^*(m)$ is not necessarily concave.

As an illustration of nonconcavity, consider the example in Figure 2. Here each "source node" (1, 2, 3) produces one unit of flow directed to respective destination node (4, 5, 6). In each case, half of the flow goes directly over one arc to the destination node and the other half flows first through node 7 before being routed on to the appropriate destination node. With one facility, i.e., m = 1, $f^*(1) = 3/2$, corresponding to one facility at node 7. With two facilities we have $f^*(2) = 2$, derived from a facility at node 7 and one at any of the other 6 nodes (there are several other optimal solutions, as well). But with three facilities, we get

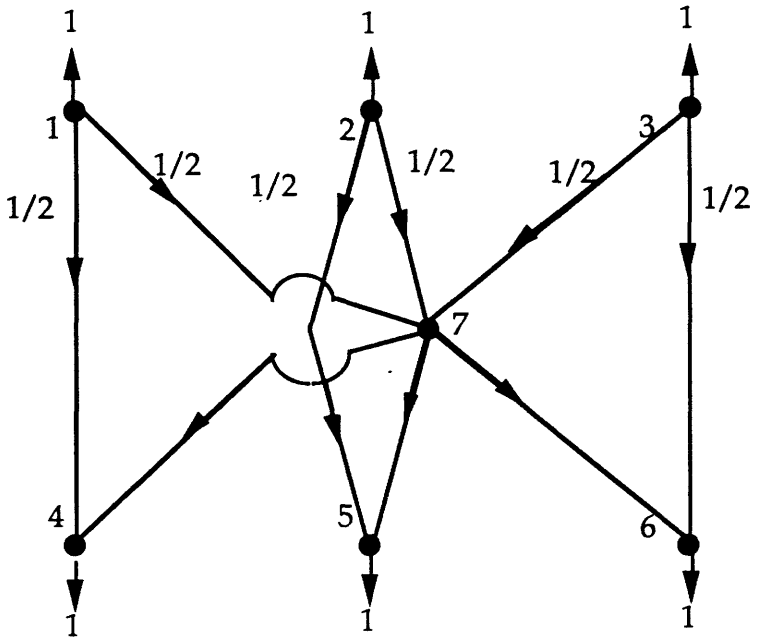

Figure 2: Example Illustrating NonConcavity of Optimal Value of Objective Function

$f^*(3) = 3$, obtained with $x_1 = x_2 = x_3 = 1$ (and with several other solutions). We see $f^*(2) - f^*(1) = 1/2$, whereas $f^*(3) - f^*(2) = 1$, clearly not concave. Such complex behavior is a prime motivation for the exact branch-and-bound algorithm of Section 4.

3.4. Worst Case Analysis

In this section, we derive the worst case performance of the greedy heuristic. In locating m facilities over the n-node network having total flow f, suppose f^* is the flow intercepted by optimally locating the facilities and that $\bar{f} \leq f^*$ is the flow intercepted by the greedy heuristic. Suppose further that f_1 is the flow through the node having the most nodal flow, i.e., $f_1 = \max_{j \in N} \{b_j\}$.

Lemma 2. $\bar{f} \geq \left(\frac{n-m}{n-1}\right) f_1 + \left(\frac{m-1}{n-1}\right) f.$ (*)

Proof: Define $f_k \equiv$ the new flow added to the objective function through the kth selected node in the heuristic. Clearly (*) holds (with equality) for m = 1. We now prove the general result by induction.

Suppose for m = j,

$$\bar{f}_j \equiv f_1 + f_2 + \ldots + f_j \geq f_1 \left(\frac{n-j}{n-1}\right) + f\left(\frac{j-1}{n-1}\right),$$

then we shall prove that for m = j + 1,

$$\bar{f}_{j+1} = f_1 + f_2 + \ldots + f_{j+1} \geq f_1 \left(\frac{n-j-1}{n-1}\right) + f\left(\frac{j}{n-1}\right).$$

Given the inductive hypothesis, some $\Delta \geq 0$ we can write

$$\bar{f}_j = f_1 \left(\frac{n-j}{n-1}\right) + f\left(\frac{j-1}{n-1}\right) + \Delta.$$

Recalling that the heuristic always selects next the most heavily trafficked node in the residual network, we may write

$$\bar{f}_{j+1} = \bar{f}_j + f_{j+1} \geq \bar{f}_j + \left(\frac{f - \bar{f}_j}{n-j}\right)$$

or, substituting for $\bar{f}_j$,

$$\bar{f}_{j+1} \geq f_1 \left(\frac{n-j}{n-1}\right) + f\left(\frac{j-1}{n-1}\right) + \Delta + \left(\frac{f - \left\{f_1\left(\frac{n-j}{n-1}\right) + f\left(\frac{j-1}{n-1}\right) + \Delta\right\}}{n-j}\right)$$

$$= f_1 \left(\frac{n-j-1}{n-1}\right) + f\left(\frac{j}{n-1}\right) + \Delta\left(1 - \frac{1}{n-j}\right)$$

$$\geq f_1 \left(\frac{n-j-1}{n-1}\right) + f\left(\frac{j}{n-1}\right)$$

■

Lemma 2 is intuitive. After selecting f_1 at the first iteration, the average flow per node in the residual networks is $(f - f_1)/(n-1)$. The remaining $(m-1)$ node selections must do at least as well as average, hence

$$\bar{f} \geq f_1 + (m-1)\ \{(f-f_1)/(n-1)\},$$

which reduces to (*).

Lemma 3. $f^* \leq \min(f, mf_1)$.

Proof. Omitted

Worst case peformance is now computed in

Theorem 3. $\dfrac{f^* - \bar{f}}{f^*} \leq 1 - \dfrac{n-m}{(n-1)\,m} - \left(\dfrac{m-1}{n-1}\right)$

Proof. $\dfrac{f^* - \bar{f}}{f^*} = 1 - \dfrac{\bar{f}}{f^*} \leq 1 - \dfrac{\left\{\dfrac{n-m}{n-1} f_1 + \dfrac{m-1}{n-1} f\right\}}{\min(f, m f_1)}$

If $f \leq mf_1$,

$$\frac{f^* - \bar{f}}{f^*} \leq 1 - \frac{\left\{\dfrac{n-m}{(n-1)m} f + \dfrac{m-1}{n-1} f\right\}}{f}$$

$$= 1 - \left\{\frac{n-m}{(n-1)m} + \frac{m-1}{n-1}\right\}$$

If $mf_1 < f$,

$$\frac{f^* - \bar{f}}{f^*} \leq 1 - \frac{\left\{\frac{n-m}{n-1} f_1 + \frac{m-1}{n-1} m f_1\right\}}{m f_1}$$

$$= 1 - \left\{\frac{n-m}{(n-1)m} + \frac{m-1}{n-1}\right\}$$

Hence,

$$\frac{f^* - \bar{f}}{f} \leq 1 - \frac{n-m}{(n-1)m} + \left(\frac{m-1}{n-1}\right). \quad \blacksquare$$

As a test for Theorem 3, when m = 1, we have

$$1 - \frac{n-m}{(n-1)m} - \left(\frac{m-1}{n-1}\right) = 1 - \frac{n-1}{(n-1)} - 0 = 1 - 1 = 0,$$

which is the expected since the greedy heuristic is optimal for m = 1. When m = n,

$$1 - \frac{n-m}{(n-1)m} - \frac{m-1}{n-1} = 1 - 0 - 1 = 0,$$

which is again expected.

For more realistic cases, the worst case performance is within 82% of optimal for n = 100, m = 10; and within 59% of optimal for n = 20 and m = 3.

4. The Branch-and-Bound Algorithm

The decision variables of our branch-and-bound algorithm are the binary variables x_j, $j \in N$. Within the context of formulation P1, the binary variables y_p, $p \in P$ do not need to be calculated directly as they are calculated from the x_j's. In the process of the algorithm we create partial solutions by specifying some of the facility locations, i.e., assigning only some of the x_j to zero or one, and leaving the rest of them undefined. Then we use the objective function upper bounds, which are

computed using the partial solutions, in order to determine whether or not the partial solution under examination could be part of the optimal one.

In order systematically to construct and examine these partial solutions, we use a binary tree structure, i.e., a tree whose nodes have exactly two, or zero, children nodes. We start with the root node corresponding to a solution with all variables undefined. Then, we move on to create the nodes of the first depth level, which are the left and right children of the root node. These two nodes correspond to partial solutions with one of the decision variables, say x_{j1}, set first to zero and then to one, respectively. We repeat this procedure, as if these nodes were the root nodes of two distinct new trees, in order to create their four children nodes, which now correspond to partial solutions with two defined decision variables, x_{j1} (zero or one) and some new x_{j2}, (zero or one). These are the nodes of the second depth level. It should be clear, that each level corresponds to a new decision variable x_{jd}, for d = 1, 2, . . ., n, being defined as zero and one, in order to bring the feasible partial solutions one step closer to completion. Since the maximum possible depth level reached by the algorithm is n, the maximum number of tree nodes possibly created is:

$$2^0 + 2^1 + 2^2 + \cdots + 2^n = \sum_{d=0}^{n} 2^d = 2^{n+1} - 1.$$

Thus, each node in the tree corresponds to a partial solution with as many defined variables as the depth level of the node.

As we create these partial solutions, we will use, apart from constraint (1), another fathoming test, in order to determine whether we should further consider them or not. This test is the computation of objective function value upper bounds, derived from the partial solution, and its comparison with the current lower bound on the optimal objective function value.

4.1 Upper Bounds

We calculate two upper bounds for our partial solutions. Let us assume that we have a partial solution with d of the x_j variables defined to be zero or one.

The first upper bound which we call the zero-ubound becomes tighter as the partial solution includes more x_j variables assigned to zero. The idea is that when no variables are defined, the potential value of the objective function ranges from zero to f. As soon as at least two variables x_j are assigned to zero in the same partial solution, any flow that passes through paths containing only the corresponding nodes, cannot be intercepted by any solution, that includes this partial one. Therefore the zero-ubound becomes:

$$f - \sum_{p \in P'} f_p , \tag{3}$$

where P' is the set of paths p, such that x_j is set to zero for all nodes j in p, i.e.,

$$P' = \{ p \in P \mid j \in p \rightarrow x_j = 0 \}.$$

The second upper bound for a partial solution, which we call the one-ubound, becomes tighter as the solution includes more decision variables assigned to one. The idea behind it is that "$x_j = 1$" corresponds to a facility already located at node j. This facility intercepts the flow of all paths that include node j. If more than one of the decision variables are set equal to one, the corresponding facilities might intercept common flow. If, for instance, x_{j_1} and x_{j_2} are both equal to one, then the flow of any path that includes both nodes j_1 and j_2, is intercepted by both facilities. Let us assume that in a partial solution with d of the decision variables defined, *l* of the m facilities have already been located, i.e., there are *l* decision variables set to

one. In the remaining [n – d] decision variables to be defined, there should be exactly [m – *l*] set equal to one, in order for the solution to be a feasible one. When we compute the one-ubound, we assume that the [m – *l*] facilities will be located at the nodes that present the [m – *l*] highest flow interception values among the [n –d] nodes corresponding to the undefined variables.

To express the one-ubound, let D be the set of nodes that comprise a given partial solution and let U be the set of nodes N – D. Then, the one-ubound for a partial solution corresponding to a node of depth level d, with *l* of the d defined x_j variables set to one is:

$$\left[\sum_{j \in D} b_j x_j\right] - \left[\sum_{p \in P} \left\{\left(\sum_{j \in p} x_j - 1\right)\right\}^{+} f_p\right] + L$$

where L is the sum of the [m – *l*] largest remaining nodal flow interceptions b_j for $j \in U$ (after removing from P all the flow intercepted by the *l* already located facilities). The expression $\{K\}^+$ is equivalent to max(K,0) and ensures that, if there is no facility located by the partial solution on a path p, then its flow f_p will contribute zero to the upper bound of the objective function, not a negative amount.

Finally, we can take the minimum of these two upper bounds and declare it the global upper bound (ub) for the objective function value of the partial solution of interest. As mentioned earlier, the zero-ubound favors partial solutions with many x_j's set to zero and the one-ubound favors those with many x_j's set to one. Thus, the intention is that the two bounds will complement each other in providing one useful upper bound for any partial solution.

4.2 The Algorithm

For any node in the tree, let **d** denote its depth level and **c** denote the count (number) of facilities already located by the corresponding partial solution. The branch-and-bound algorithm can be presented as follows:

Step 1: Guess an initial complete feasible solution $[x_1, x_2, \ldots, x_n]$ and compute its objective function value. Set the lower bound [LB] of the optimal objective function at this value and declare this initial solution as the best solution found so far.

Step 2: Choose as node 1 of the branch and bound tree the variable that intercepts the maximum flow in the network, and compute its upper bound ub(1), which, since no variable is yet defined, is set to be equal to the total network flow. Go to Step 6.

Step 3: Create the next tree-nodes, by choosing variable x_j which intercepts the maximum remaining flow. Compute the depths, counts and upper bounds for the two children of x_j, x_{j_1} and x_{j_2}.

Step 4: For the child with the larger upper bound, perform the following fathoming tests:

(i) If ub $\leq$ LB, or if $n - d < m - c$ then fathom the tree node and move to Step 6,

(ii) Else if $c = m$, then fathom the tree node and move to Step 5. Otherwise go to Step 6.

Step 5: Since c = m, all the facilities are already located and therefore the tree node corresponds to a complete, not partial solution, with all the undefined variables set to be zero. Compute the objective function value of this solution. If it is equal to LB, then move to Step 6. If it is greater than LB, then update LB to resume its value and this solution becomes the best solution found so far. In this case, go back and repeat the first fathoming test for all the tree nodes which are currently not fathomed leaves in the tree structure, i.e., the current subtree root candidates. Then move on to Step 6.

Step 6: If no unfathomed leaves are left in the tree, then stop; the best solution found so far is indeed the optimal one and its objective function value is the current LB. If there exist more unfathomed leaves in the tree, then choose that leaf of the tree with maximum upper bound and go to Step 3.

4.3 Example and Computational Results

Figure 3 presents the branch-and-bound tree for the example of Figure 1. This example was first presented as a case where the "greedy" heuristic fails to provide the optimal solution. Now, we will show how the branch-and-bound algorithm will produce the optimal solution. Let us assume that ε is 0.2. We arbitrarily choose an initial feasible solution, say $x_C = 1$ and $x_D = 1$. The initial objective function value is 2.4 flow units, which is also the initial lower bound (LB) for the optimal objective function value. Since the upper bound for all the nodes up to node 9 is larger than that of the lower bound set by the initial solution at 2.4 units, no fathoming occurs before node 9. At this point, c = m = 2 and we obtain the first

feasible complete solution in the tree: $x_A = 1$ and $x_B = 1$. Its objective function value is four flow units, greater than the current LB and therefore the solution [$x_A = 1$ and $x_B = 1$] becomes the new best solution found so far and LB gets updated to four flow units. But now, nodes 3, 5, 6, and 8 which are the current unfathomed tree leaves, get fathomed, because their upper bound is not greater than the current LB. Having done so the tree cannot be expanded any more and the algorithm has to stop. Its current best solution is indeed the optimal one: nodes A and B are the optimal locations for the two facilities.

Any feasible solution can be selected to provide the initial lower bound for the algorithm. Based on our computational experience we recommend the feasible solution produced by the "greedy" heuristic.

The computer code implementing the algorithm is written in PASCAL TURBO and tests were run on an IBM PC/XT with 512K.

In order to produce test results we generated randomly network sizes, their paths and corresponding flows. In Table 1 we illustrate a typical sample of our test cases. The table provides the CPU time and number of nodes generated in the branch and bound tree for networks with number of nodes and number of paths ranging from 10 to 100 and number of facilities ranging from 2 to 5.

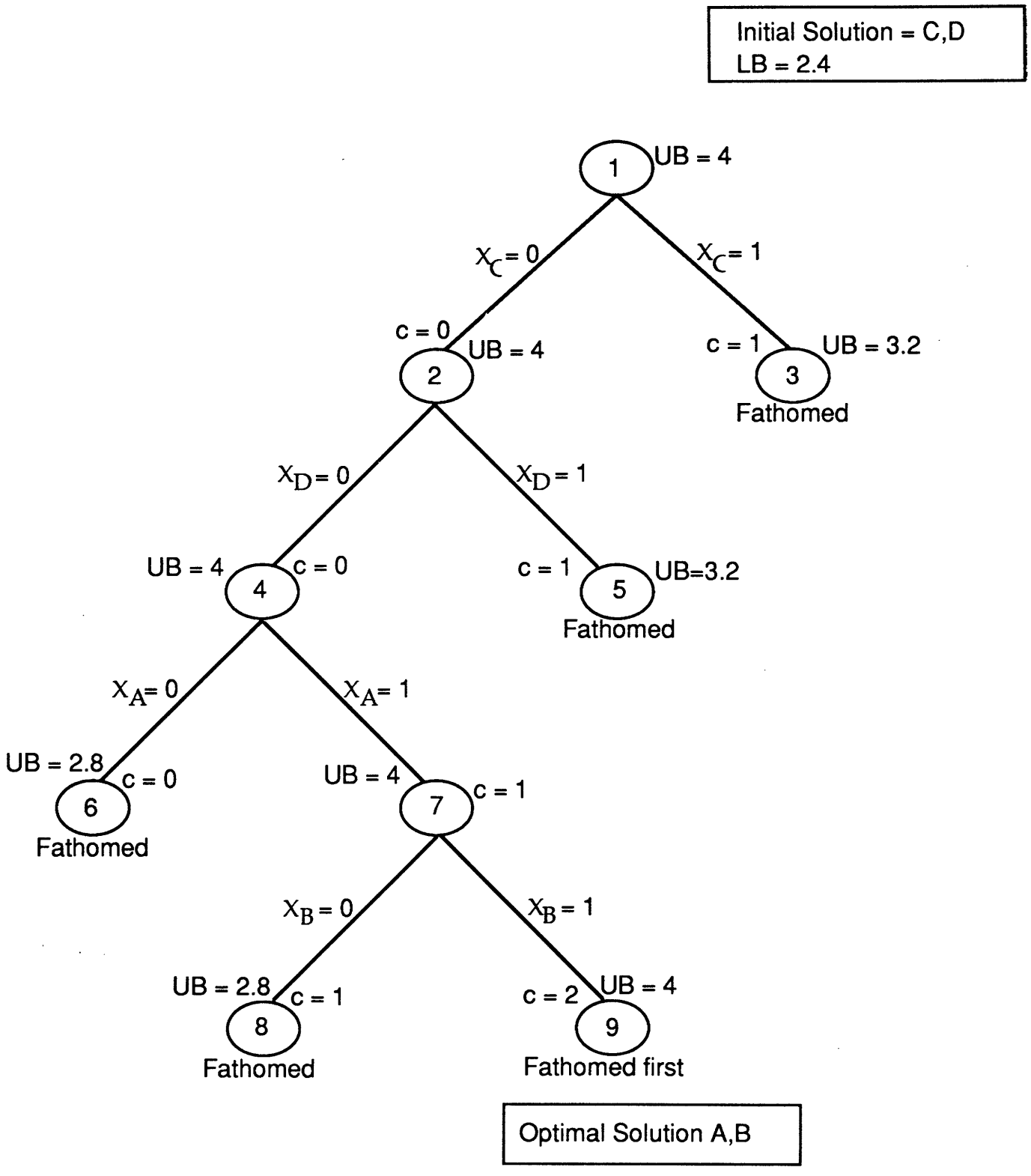

Figure 3: Branch and Bound Tree for the Example in Figure 1

m	2		3		4		5	
Network n, |P|	CPU Time	# of nodes	CPU Time	# of nodes	CPU Time	# of nodes	CPU Time	# of nodes
10,10	0:0:1:39	4	*	*	*	*	*	*
20,20	0:0:4:87	14	0:0:3:92	11	*	*	*	*
30,30	0:0:15:08	28	0:0:48:01	103	*	*	*	*
40,40	0:0:30:18	39	0:1:10:14	99	0:3:20:68	302	0:12:11:12	865
50,50	0:0:42:96	40	0:1:22:17	83	0:6:15:70	372	0:14:40:13	877
60,60	0:1:21:40	55	0:5:45:53	279	0:12:32:01	620	0:40:12:10	1470
70,70	0:2:10:86	69	0:11:42:08	359	0:29:12:55	1105	1:19:15:01	2539
80,80	0:3:14:92	80	0:14:58:37	432	1:14:24:19	2231	2:36:1:48	4800
90,90	0:4:29:31	89	0:29:40:08	698	1:35:17:1	2449	3:46:20:90	5900
100,100	0:5:34:20	90	0:21:10:16	396	2:15:8:21	3100	6:8:25:29	7623

Table 1: CPU Time and Number of Nodes on the Branch and Bound Tree for $n = |P| = 10, \ldots, 100$ and $m = 2, 3, 4, 5$
(Time presented as Hours: Minutes: Seconds: Hundredths of Second)
*The greedy heuristic gives a solution that intercepts all the flow in the network.

In 65% of cases tested (more than 200 runs), the "greedy" heuristic produced the optimal solution. In the cases for which the heuristic's solution was suboptimal the difference in the objective function value was always less than 3.5% and in most cases less than 1%. The CPU time of the heuristic for a network with $n = |P| = 100$ and $m = 5$ is 24 seconds. Also, the one bound proved in general to be more useful than the zero bound.

5. The Required Number of Facilities

We can combine our greedy heuristic with our exact branch-and-bound procedure to answer, as well, the following related question: what is the minimum

number of facilities needed to "capture" an externally specified fraction of total flow? The problem is to find the minimum number of facilities and their respective locations that intercept a predetermined total flow at least equal to

$(1-\alpha) \sum_{p \in P} f_p \equiv c^*_\alpha$, where α = fraction of total network flow not intercepted.

Invoking nodal optimality, the problem can be formulated as:

$$\min \sum_{j=1}^{n} x_j \qquad \text{(Problem } C^*_\alpha \text{)}$$

s. t.
$$\sum_{j \in p} x_j \geq y_p \qquad \forall p \in P \qquad (4)$$

$$\sum_{j \in P} y_p f_p \geq c^*_\alpha \qquad (5)$$

$$x_j = 0, 1, \quad \forall j \in N; \quad y_p = 0, 1 \quad \forall p \in P$$

Notice that if $\alpha = 0$ constraint (5) becomes redundant and y_p can be replaced by 1 for all $p \in P$ in (4).

First we consider problem C^*_0, that is, the problem for which all the network flow must be intercepted. Subsequently we solve C^*_α for all $0 \leq \alpha \leq 1$.

5.1 Solving Problem C_0^*

Consider problem C_0^*

$$\min \sum_{j=1}^{n} x_j \qquad \text{(Problem } C_0^*\text{)}$$

s. t.
$$\sum_{j \in p} x_j \geq 1 \qquad \forall p \in P \qquad (6)$$

$$x_j = 0, 1 \qquad \forall j \in N \qquad (7)$$

Obviously C_0^* is a set covering problem. However due to its special structure it is a very easy problem to solve. First, even though the set of constraints (6) can be huge since there may be many possible paths between any pair of source and destination nodes, this is not a real concern, as shown in

Lemma 4. P can be reduced to P' where $| P' | \leq \frac{n2 - n}{2}$ ($| A |$ denotes the cardinality of a set A).

Proof. For any pair of paths p_1, $p_2 \in P$ if $p_1 \subseteq p_2$, p_2 can be deleted from P in C_0^*. Therefore the set P' cannot contain more elements than the number of one-link paths in P, numbering at most $(n^2 - n) / 2$.

Let m_0^* denote the optimal number of facilities. For a complete graph a direct consequence of Lemma 1 is given in

Corollary 4.1. If G is a complete graph and there is a positive flow between all pairs of nodes in N, then $m_0^* = n - 1$ and any subset of $n - 1$ nodes is optimal.

Let us define a coverage matrix C with rows corresponding to paths in P' and columns corresponding to nodes in N, where

$$c_{pj} \equiv \begin{cases} 1 & \text{if } j \in p \\ 0 & \text{otherwise} \end{cases}$$

For the set covering problem there are three simple reduction operations that can allow us to reduce the size of C [6] . Here we present them using set operations (instead of column and row operations that are traditionally used [6].) The first reduction operation, corresponding to elimination of rows (used in the proof of Lemma 1) states that if $p_1 \subseteq p_2$, p_2 can be deleted. Let $P_j = \{p \in P' \mid j \in p\}$, i.e., the set of all positive flow paths in P' containing node j. Then the second reduction operation, corresponding to elimination of columns, can be stated as follows: If $P_j \supseteq P_i$ delete column i, set $x_i = 0$ and delete node i from each p in P'. The third operation, focusing on zero-length paths found during the reduction, corresponds again to row elimination: If $\exists p, p = j \quad j \in N$ set $x_j = 1$, and delete from P' all paths that include node j.

5.2 Solving Problem C^*_α

By modifying the greedy heuristic, we now find the minimal number of facilities, m^*_α corresponding to the solution to problem C^*_α, for any α. Define the relaxed P1 problem RP1 with noninteger constaints $0 \le x_j \le 1$ and $0 \le y_p \le 1$.

An Ascent Algorithm to Solve C^*_α

Step 1. $l = 1$, Set $x_j = 0 \quad j = 1, \ldots, n$.

Step 2. Compute $b_j \quad \forall j \in N$.

Step 3. Find $b_{max} = \max_{j \in N} \{b_j\}$; locate facility l at a node j_{max} having flow b_{max}, i.e., $x_{j_{max}} = 1$ (in a case of a tie, choose j_{max} to be the node with the smallest index j).

Step 4. Delete from P the set $P_{j_{max}}$ and delete from N node j_{max}.

Step 5. If $\sum_{j=1}^{n} x_j b_j \geq c_\alpha^*$ set $m_{max} = l$ and go to Step 6. Otherwise, set $l = l + 1$ and return to step 2.

Step 6. Set $l = l - 1$ and solve RP1. If the objective function value of RP1 is less or equal than c_α^*, go to Step 7, otherwise solve P1 using the exact branch and bound algorithm. If the optimal objective value of P1 is larger than c_α^* repeat Step 6. Otherwise go to 7.

Step 7. $m_\alpha^* = l + 1$ is the optimal objective function value.

We note that the first four steps of the algorithm are identical to the "greedy" heuristic.

The optimality of the algorithm to solve Problem C_α^* should be obvious as shown in

Theorem 3. The ascent algorithm produces the optimal solution of Problem C_α^*.

Proof. m_{max} is an upper bound on the optimal value of the objective function of C_α^*. RP1 provides an upper bound on problem P1. Therefore if this upper bound is

less than or equal to c^*_α the exact solution of P1 cannot provide a feasible solution with optimal objective function value larger than c^*_α. ■

Once the optimal solution of C^*_α, m^*_α is obtained, P1 can be resolved to provide a set of m^*_α locations possibly with better total flow.

5.3 Numerical Example

As an example consider Table 2 that includes all the paths with positive flows of a network with 7 nodes.

PATH	FLOW
1-3-5-2	30
1-3	20
1-2-5	10
1-2-4	40
1-6	30
2-1	20
2-5-3	25
2-4	30
2-6	10
3-1	10
3-5-2	20
3-5	10
3-4	20
3-7	30
3-7-4	25
6-1-3-7	20
4-2	10
6-1-2	20
7-4	30
7-3-5	25
7-4-2-6	20
TOTAL	455

Table 2. Path and Flows for the Example.

Now we solve the problem with $\alpha = 10\%$ or $c^*_{10\%} = (.9)\ (455) = 409.5$

STEP 1: $l = 1$; **STEP 2**: $b_1 = 200,\ b_2 = 235;\ b_3 = 235,\ b_4 = 175,\ b_5 = 120,\ b_6 = 100,$ $b_7 = 150$; **STEP 3**: $b_{max} = b_2,\ x_2 = 1$; **STEP 4**: P = {1-3, 1-6, 3-1, 3-5, 3-4, 3-7, 3-7-4, 6-1-3-7, 7-4, 7-3-5}, N = {1, 3, 4, 5, 6, 7}; **STEP 5**: $\sum_{j=1}^{7} x_j b_j = 235$, $l = 2$; **STEP 2**: $b_1 = 80,\ b_3 = 160,\ b_4 = 75,\ b_5 = 35,\ b_6 = 50,\ b_7 = 130$; **STEP 3**: $b_{max} = b_3,\ x_3 = 1$; **STEP 4**: P = {1-6, 7-4}, N = {1, 4, 5, 6, 7}; **STEP 5**: $\Sigma x_j b_j = 235 + 100 = 395$, l=3; **STEP 2**: $b_1 = 30,\ b_4 = 30,\ b_5 = 0,\ b_6 = 30,\ b_7 = 30$; **STEP 3**: $b_{max} = b_1,\ x_1 = 1$; **STEP 4**: P = {7-4}, N = {4, 5, 6, 7}; **STEP 5**: $\Sigma x_j b_j = 395 + 30 = 425,\ m_{max} = 3$; **STEP 6**: $l = 2$, the objective function value of RP1 is 395 and therefore the optimal objective function is $l = 3$ in STEP 7. The optimal solution is to locate at nodes 1, 2, 3. If we solve problem P1 with $l = 3$ we obtain an optimal objective function value of 445 and optimal locations at nodes 1, 3, and 4.

5.4 Computational Results

The computer code implementing the algorithm is written in TURBO PASCAL and tests were run on an IBM PC. In order to produce test results we randomly generated networks, travel paths and flows. Table 3 summarizes the test results for number of nodes (n) and number of paths ($|P|$) ranging from 10 to 100 with $\alpha = 5\%$. In the table we summarize the average CPU time of the algorithm and the average CPU time of the five first steps of the algorithm (which we call henceforth the greedy heuristic) based on 10 problems solved in each category of n and $|P|$ (to the total of 100 problems).

NETWORK n, |P|	CPU TIME hr:min:sec: hundreths of sec.	CPU TIME GREEDY hr:min:sec: hundreths of sec.
10,10	0:0:18:90	0:0:4:77
20, 20	0:0:20:42	0:0:5:43
30, 30	0:1:23:12	0:0:5:86
40, 40	0:2:3:05	0:0:6:44
50, 50	0:2:57:24	0:0:7:50
60, 60	0:3:38:08	0:0:8:33
70, 70	0:8:26:70	0:0:9:52
80, 80	0:22:31:60	0:0:10:74
90, 90	0:38:21:12	0:0:12:88
100, 100	1:5:46:02	0:0:15:42

Table 3: Computational Results for Random Problems

In addition, the following results not shown in Table 3 were obtained:

1. The algorithm terminates without the need to utilize the branch and bound routine in approximately 75 percent of the test problems.

2. In all problems solved the optimal number of facilities was either the same as or one less than that obtained with the greedy heuristic.

3. In all problems solved, when RP1 resulted in an objective function greater than C_α^*, so did P1.

Acknowledgements

The authors gratefully thank two referees whose suggestions improved this paper. The work of the second and third authors was supported by the National Science Foundation, grants ECS-8714649 and SES-8709811.

References

1. S. L. Hakimi, "Optimum Locations of Switching Centers and the Absolute Centers and Medians of a Graph," *Operations Research*, **12**, pp. 450-459 (1964).

2. S. L. Hakimi, "Optimum Distribution of Switching Centers in a Communication Network and Some Related Graph Theoretic Problems," *Operations Research*, **13**, pp. 462-475 (1965).

3. R. L. Church and C. ReVelle, "The Maximal Covering Location Problem," *Papers Regional Science Association*, **32**, pp. 101-118 (1974).

4. R. L. Church and M. E. Meadows, "Location Modeling Using Maximum Service Distance Criteria," *Geograph. Anal.*, **11**, pp. 358-373 (1979).

5. N. Megiddo, E. Zemel, and S. L. Hakimi, "The Maximum Coverage Location Problem," *SIAM J. Alg. Disc. Meth.*, **4**(2), pp. 253-261 (1983).

6. Garfinkel, R.S., and Nemhauser, G.L., *Integer Programming*, New York: John Wiley and Sons, 1972.