SIAM J. ALG. DISC. METH.
Vol. 4, No. 2, June 1983

0196-5212/83/0402-0012 $01.25/0

THE MAXIMUM COVERAGE LOCATION PROBLEM*

NIMROD MEGIDDO,† EITAN ZEMEL‡ AND S. LOUIS HAKIMI§

Abstract. In this paper we define and discuss the following problem which we call the maximum coverage location problem. A transportation network is given together with the locations of customers and facilities. Thus, for each customer i, a radius r_i is known such that customer i can currently be served by a facility which is located within a distance of r_i from the location of customer i. We consider the problem from the point of view of a new company which is interested in establishing new facilities on the network so as to maximize the company's "share of the market." Specifically, assume that the company gains an amount of w_i in case customer i decides to switch over to one of the new facilities. Moreover, we assume that the decision to switch over is based on proximity only, i.e., customer i switches over to a new facility only if the latter is located at a distance less than r_i from i. The problem is to locate p new facilities so as to maximize the total gain.

The maximum coverage problem is a relatively complicated one even on tree-networks. This is because one aspect of the problem is the selection of the *subset* of customers to be taken over. Nevertheless, we present an $O(n^2p)$ algorithm for this problem on a tree. Our approach can be applied to other similar problems which are discussed in the paper.

1. Introduction. We shall discuss in this paper problems in which establishing new facilities in an existing network is aimed at attracting a maximum number of customers. There is thus some competitive flavor to such problems in that the existing facilities may belong to one company while a second company is trying to extract the maximum profit by locating its own facilities on the same network. For further discussion of this and related problems the reader is referred to [H2].

Consider a graph $G=(V, E)$ with edge-lengths d_{ij}. We identify each edge (i, j) of G with a line-segment of length d_{ij} so that we can speak of points (not necessarily vertices) on the edges of G. Each such point x is identified by its distances from the endpoints of the appropriate edge. For every two points x, y of G let $d(x, y)$ be the length of the shortest path between them along the edges of G.

We associate a "customer" with each vertex of G. We assume that there exists a threshold radius r_i for each customer i so that if a new facility is established within a distance of r_i from i then this customer would start using the new facility (unless, of course, an even closer facility is established; in any case, a customer uses one of the closest facilities). We say in such a case that customer i is "covered," with a resulting gain which we denote by w_i (the "weight" of i). The results of this paper are valid for any set of positive constants r_i. However, several simplifications are possible if the threshold radii are derived from distances to old facilities already positioned on G. This topic is addressed in Appendix 2.

Assume that p facilities can be located anywhere on G (including points on the edges other than vertices). We wish to locate these facilities so as to maximize the total weight of the customers which are covered. We will show that it is easy to identify a fairly small subset of points $Y=\{y_1, \cdots, y_m\}$ such that we need consider only points

* Received by the editors September 1, 1981, and in revised form July 1, 1982. This work was supported in part by the National Science Foundation under grants ECS7909724, SOC7905900, ENG7902506 and ECS8121741.

† Department of Statistics, Tel Aviv University, Tel Aviv 69978, Israel.

‡ J. L. Kellogg Graduate School of Management, Northwestern University, Evanston, Illinois 60201. The research of this author was partially supported by the J. L. Kellogg Center for Advanced Studies in Managerial Economics.

§ Department of Electrical Engineering and Computer Science, Northwestern University, Evanston, Illinois 60201.

of Y as possible location for facilities. Alternatively, the problem may be originally specified with respect to a finite set of feasible location sites.

Our maximum coverage problem is obviously NP-hard on a general graph since the problem of minimum dominating set (see [GJ]) can be formulated as that of minimizing the number of facilities required to gain $W = n$ (where n is the number of vertices). To that end we set $r_i = 1.5$, $w_i = 1$, $d_{ij} = 1$ for all i, j.

We shall present a polynomial-time algorithm for the maximum coverage problem on a tree. We note that unlike the problem of minimum dominating set on a tree (which is easily solvable in linear time), or even that of gaining the total weight (i.e., covering all the vertices), the existence of a polynomial-time algorithm for our problem is nontrivial. The relative difficulty is due to the fact that here we do not require covering of *all* vertices but only k of them (in the special case of unit weights and $W = k$, say). Thus, the large number of different combinations of k out of n complicates the problem. Further evidence to the difficulty of the problem even on a tree is given by the fact that, unlike in many other locational problems on trees (see [T], [K]), the integer linear programming problem associated with ours is *not* solvable as a regular linear program (as we demonstrate in Appendix 1).

In § 2 we discuss the set of potential points for the construction of new facilities. The basic routines of the algorithm are described in § 3. The algorithm itself is explained in § 4. In § 5 we discuss the complexity of the algorithm. Section 6 discusses further problems, related to the maximum coverage problem, which are solvable by a modified version of our algorithm. Appendix 1 describes the linear programming aspects and Appendix 2 discusses simplifications in the case where all threshold radii are implied from distances to (old) existing facilities.

2. Potential locations for new facilities. As we stated in the introduction, a new facility may be constructed at any point of the graph. We denote by $d(x, y)$ the distance (along the shortest route) between any pair of points (x, y). However, we shall identify a fairly small finite set from which an optimal combination can be selected. Our algorithm can also be applied to problems in which new facilities can be constructed at designated points, $y_1, \cdots, y_m$, only.

Let U_i be the r_i-neighborhood of vertex i, i.e., U_i is the set of all points x such that the distance between i and x is *less* than r_i. For every set S of vertices, let $U_S = \bigcap_{i \in S} U_i$. We say that U_S is *maximal* if $U_S \neq \emptyset$ and $U_T = \emptyset$ for every $T \supsetneq S$. Obviously, we may assume without loss of generality that a new facility will always belong to some maximal U_S. Moreover, we may select in advance a single point $y_S \in U_S$ from each maximal U_S and consider only these points y_S for locations of new facilities. We shall now prove that the number of maximal U_S's is not too large.

THEOREM. *On a general graph with e edges and n vertices the number of maximal U_S's is $O(en)$ while on a tree this number is $O(n)$.*

Proof. First, note that if x is a boundary point of a set U_S then there exists a vertex i such that $d(i, x) = r_i$. Thus, each vertex i can contribute at most two boundary points on each edge of the graph. Moreover, i cannot contribute the same boundary point to more than one maximal U_S. It follows that a single edge contains no more than $2n$ boundary points and hence the total number of boundary points is at most $2en$. This establishes the first claim of the theorem.

Consider now the case where G is a tree T. Note first that there are at most n maximal U_S's which contain a vertex, since these sets are pairwise disjoint. It thus suffices to show that the number of maximal U_S's which do not contain any vertex is $O(n)$. Every such U_S has precisely two boundary points x_i, x_j such that i is a vertex

with $d(i, x_i) = r_i > d(i, x_j)$ and j is a vertex with $d(j, x_j) = r_j > d(j, x_i)$. The set U_S could thus be identified with an interval (x_i, x_j). By removing this interval from T, our tree decomposes into two subtrees T_i, T_j, such that for every point $x \neq x_i$ in T_i, $d(i, x) > r_i$ and for every point $x \neq x_j$ in T_j, $d(j, x) > r_j$ (see Fig. 1). Thus, if vertex i contributes

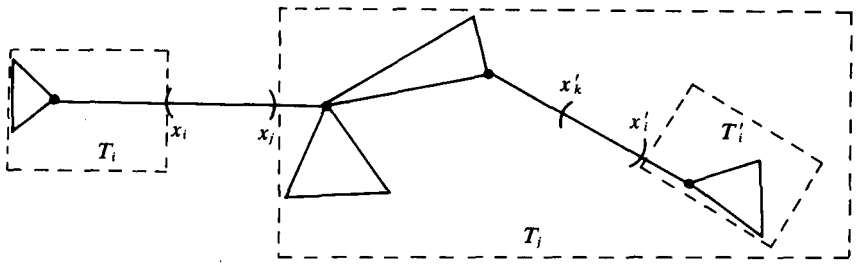

FIG. 1

a boundary point to another such interval, then that interval must be completely contained in T_j. Suppose that there is such an interval (x'_i, x'_k), where $d(i, x'_i) = r_i > d(i, x'_k)$ and k is a vertex such that $d(k, x'_k) = r_k > d(k, x'_i)$. It is easy to verify that there can be no interval to which both j and k contribute two distinct boundary points. In general, consider a graph G^* on the vertices of T such that vertices u and v are linked with an edge in G^* if and only if there exists an "interval" to which both u and v contribute distinct boundary points as previously described. Then, it can be proved by induction that G^* has no cycles. This implies that the number of those intervals is not greater than $n-1$ and that completes the proof.

We note that the determination of all the boundaries of the maximal U_S's can be easily carried out in $O(n^2)$ time.

3. The basic routines. The algorithm for the maximum coverage problem on a tree works in general as follows. Suppose that the potential locations are at the points $y_1, \cdots, y_m$. To simplify the presentation, let us assume that the potential locations are precisely the vertices themselves. Since $m \leqq 2n$ it follows that by adding all the y_j's as vertices we do not lose in terms of the asymptotic complexity.

Let the tree now be rooted at an arbitrary vertex u. If u is selected for a location of a new facility then we proceed, recursively, by looking at the subtrees rooted at the neighbors of u, taking into consideration the fact that there has been a new facility established at u. This requires, however, the solution of a "resource allocation" problem, i.e., optimizing the distribution of the $p-1$ remaining facilities among the subtrees rooted at the neighbors of u. If u is not selected as a location of a new facility, we proceed, recursively, to the subtrees and then we have to consider the interactions between these subtrees caused by the fact that vertices of one subtree may be covered by a facility located in another.

In order to overcome all these difficulties we define the following routines:

1. EXT (T, π, r). Here T is a rooted tree with parameters d_{ij}, r_i, w_i as explained in the introduction, π is an integer and r is a nonnegative number. The routine EXT finds the *maximal* total weight of vertices in T that can be gained by locating π new facilities in T, given that there is one additional facility outside of T at a distance r from the root. Thus, this gain consists of the total weight of vertices i, such that the distance between i and the root is less than $r_i - r$, plus the total weight of *other* vertices i such that the distance between i and one of the π new facilities is less than r_i.

2. Int (T, π, r). With T and π as in EXT (T, π, r) this routine solves the maximum coverage problem on T with π new facilities but with an additional requirement that at least one new facility *must* be located at a distance less than or equal to r from the root of T.

It is easy to verify that the routine EXT has at most n critical values for the parameter r, namely, the differences $\delta_i = r_i - d(i, u_T)$ (where u_T is the root of T), for all vertices i. In other words, it suffices to know the output of EXT for each δ_i in order to know that output for all values of r. Analogously, INT has just the distances $d(i, u_T)$ as the critical values for the parameter r.

3. ALLOC $(f_1, \cdots, f_k; \pi)$. This is a routine that solves the resource allocation problem with concave returns. Specifically, let $f_1, \cdots, f_k$ be monotone concave functions of discrete variables, and let π be a nonnegative integer. Then, ALLOC solves the following:

$$\text{Maximize} \quad \sum_{i=1}^{k} f_i(p_i)$$

$$\text{subject to} \quad \sum_{i=1}^{k} p_i = \pi$$

$$p_i \text{ is a nonnegative integer.}$$

Fast algorithms for ALLOC have been proposed by Galil and Megiddo [GM] and by Frederickson and Johnson [FJ]. The latter requires $O(k \log \pi)$ evaluations of the functions f_i. On the other hand, if one wishes to know the solutions of all problems for $\pi = 1, 2, \cdots, p$ (with $f_1, \cdots, f_k$ fixed), then only kp evaluations are required in addition to $O(p \log k)$ time for running the greedy algorithm.

4. The algorithm. The routines EXT and INT described in § 3 require recursive calls to each other. Our maximum coverage problem could be solved by either of these routines, if r is chosen sufficiently large.

We use the following notation. The tree T is rooted at the vertex u whose "sons" are $u_1, \cdots, u_k$. For $i = 1, \cdots, k$, T_i is the subtree rooted at u_i. Let n_i denote the number of vertices in T_i and let $d_1^i \leqq \cdots \leqq d_{n_i}^i$ denote the distances of vertices of T_i from u_i.

We first describe the routine INT (T, π, r). There are two cases to examine:

Case (i). A facility is established at u. Let $f_i(p_i) = \text{EXT}(T_i, p_i, d(u_i, u))$, $i = 1, \cdots, k$. It is easy to verify that the f_i's are monotone and concave. Obviously, in this case the total gain is $w_u + \text{ALLOC}(f_1, \cdots, f_k; \pi - 1)$.

Case (ii). No facility is established at u. Here, the routine INT considers k different subcases. In a typical subcase, a subtree $T_j (1 \leqq j \leqq k)$ is selected and for each $\rho \in \{d_1^j, \cdots, d_{n_j}^j\}$ such that $\rho + d(u_j, u) \leqq r$, the following is considered. For every $i \neq j$ $(i = 1, \cdots, k)$ let $f_i(p_i) = \text{EXT}(T_i, p_i, \rho + d(u_j, u_i))$. Also, let $f_j(p_j) = \text{INT}(T_j, p_j, \rho)$. Again, the functions f_i are monotone and concave. Note that these functions depend on the parameter ρ. Now define

$$A_j(\rho) = \text{ALLOC}(f_1, \cdots, f_k; \pi) + w_u \delta(\rho),$$

where $\delta(\rho) = 1$ if $\rho + d(u_j, u) < r_u$ and $\delta(\rho) = 0$ otherwise. Let $A_j = \text{Max}_\rho \{A_j(\rho)\}$, $j = 1, \cdots, k$.

The routine INT returns either the maximum of the A_j's or the optimal value of case (i), whichever is the larger.

The computation of EXT (T, π, r) is analogous. Again, two cases are distinguished.

Case (i). A facility is established within a distance of r from u. This, by definition, is identical with the situation solved by INT (T, π, r).

Case (ii). No facility can be established within a distance of r from the root. Note that in this case, since there is a facility already located at a distance r from the root, there are no interactions among the subtrees. On the other hand, such a constrained problem cannot be solved directly by our routines. Instead, we solve the relaxed problem (i.e., we remove the requirement of *not* constructing a facility within a distance of r from the root) but ignore the interactions among the subtrees. Specifically, let $f_i(p_i) = \text{EXT}(T_i, p_i, d(u_i, u)+r)$, $i = 1, \cdots, k$. Let $A = \text{ALLOC}(f_1, \cdots, f_k; \pi) + w_u \cdot \delta(r)$ (where the δ is as in the description of INT).

We now claim that $\text{EXT}(T, \pi, r) = \max\{A, \text{INT}(T, \pi, r)\}$. To see this note that if one of the π optimal locations for the problem solved by EXT (T, π, r) is at a distance of less than or equal to r from u then $\text{EXT}(T, \pi, r) = \text{INT}(T, \pi, r)$ and $A \leqq \text{INT}(T, \pi, r)$. Otherwise, we are in Case (ii) and no interactions exist among the subtrees. Therefore, $\text{EXT}(T, \pi, r) = A$ and $\text{INT}(T, \pi, r) \leqq A$.

5. The complexity of the algorithm. Suppose that $0 = d_0 \leqq d_1 \leqq \cdots \leqq d_{n_u}$ are the distances of vertices of T from u. Consider the function $g(r) = \text{INT}(T, \pi, r)$, where π is fixed. Obviously, g is a step-function with jumps only at $r = d_s (0 \leqq s \leqq n_u)$. Moreover, if $d_s (s \geqq 1)$ is a distance from a vertex in T_j, then

$$\text{INT}(T, \pi, d_s) = \max[\text{INT}(T, \pi, d_{s-1}), \text{ALLOC}(f_1, \cdots, f_k; \pi) + w_u \cdot \delta(u)],$$

where $f_j(p_j) = \text{INT}(T_j, p_j, d_s - d(u, u_j))$ and for $i \neq j$, $f_i(p_i) = \text{EXT}(T_i, p_i, d_s + d(u, u_i))$. This implies that when π is given, the evaluation of INT (T, π, r) for all critical values of r takes $O(n)$ computations of resource allocation. Similarly, if $\delta_0 \leqq \delta_1 \leqq \cdots \leqq \delta_{n_u}$ are the sorted values of $r_x - d(x, u)$, where x is a vertex in T, then the critical values of r in EXT (T, π, r) are in the set $\{d_0, \cdots, d_{n_u}, \delta_0, \cdots, \delta_{n_u}\}$; i.e., at a critical value either $r = d(x, u)$ or $d(x, u) + r = r_x$ for some vertex x in T. Since $\text{EXT}(T, \pi, r) = \max[\text{INT}(T, \pi, r), A]$, it follows that the evaluation of EXT (T, π, r) for all critical values of r (where π is fixed) also requires only $O(n)$ computations of resource allocation.

If the algorithm is recursively run as stated in § 4, then it requires superpolynomial time. However, this is only because the same subproblems are being solved over and over again in such an implementation. To avoid that, one just has to be careful not to compute the same thing more than once. Specifically, if we store the results of all computations then we run in polynomial-time by the following argument. The number of different problems that either EXT or INT has to solve is $O(n^2 p)$. This is because there are $O(n)$ subtrees to be considered, each with $O(n)$ critical values of r, and π may take on only the values $0, 1, \cdots, p$. When we have to compute INT (T, π, r), say, then it takes only one computation of resource allocation if all the necessary values that are returned recursively are known. This establishes a bound of $O(n^3 p \log n)$.

A more careful analysis shows that the algorithm can be implemented much faster. Consider the computation of INT (T, π, r) for example, where T is rooted at u and u has k sons. If we solve the necessary allocation problem only for one value of π, then it takes $O(k \log \pi)$ time, once the necessary EXT and INT values are known. However, we can solve the problem relative to all values of π in $O(k \min(p, \log k) + p \log k)$ time. For, once the values $f_i(1) - f_i(0)$, $i = 1, \cdots, k$ are sorted, it takes $O(p \log k)$ time to find the p largest marginal gains of the form

$f_i(m+1)-f_i(m)$ $(i=1,\cdots,k,\ m=0,\cdots,p-1)$; the initial sort should be eliminated if $kp<k\log k$, in which case those p largest values can be found in kp steps. Now, the solution of $\mathrm{EXT}(T,\pi,r)$ and $\mathrm{INT}(T,\pi,r)$ for all values of π and r takes $O(n(k\min(p,\log k)+p\log k))$, once the necessary values are known. The total effort is therefore $O(n\sum_u(\deg(u)\min(p,\log\deg(u))+p\log\deg(u)))$, where the summation is over all the vertices u and $\deg(u)$ is the degree of u. This is however $O(n^2p)$.

6. Related problems. A natural generalization of the problem treated in § 1 is as follows. Suppose that each vertex i has an additional parameter, c_i, which represents the cost of establishing a new facility at vertex i. We now replace the number p by some budget B and seek to find the maximum coverage subject to this budget. This problem, however, is NP-hard even on chain networks since the knapsack problem can be easily formulated as such a coverage problem. On the other hand, our algorithm can be modified to solve this problem on a tree in pseudo-polynomial time (see [GJ]), i.e., in polynomial time in terms of n and B. Also, the problem of covering a maximum number of vertices given a fixed budget can be solved in polynomial time on a tree by considering the equivalent problem of covering at least q vertices given a budget B. The latter is easily seen to be amendable to an algorithm similar to the one proposed in § 4.

Another related problem is that of covering all the vertices at minimum cost. More formally, we wish to select points at which facilities will be established such that every vertex i has a facility located within a distance r_i from i, and such that the total construction cost is minimized. Tamir [T] solves this problem in $O(n^3)$ time. Kolen [K] solves a related problem, where the threshold radii are associated with the facilities (rather than the demand points) in $O(nm)$ time, where m is the number of potential locations of facilities. Our approach easily provides $O(nm)$ algorithms for both these problems as follows. Define $\mathrm{INT}(T,r)$ to be the minimum total cost of facilities established in T so as to cover all the vertices of T, subject to a constraint that at least one of them has to be located within a distance of r from the root. Let $\mathrm{EXT}(T,r)$ denote the minimum total cost of covering all the vertices of T that are not covered by a facility located outside of T at a distance r from the root. For every u, these functions evaluated at the subtree rooted at u, have $O(m)$ critical values of r. These are simply the distances from u to any potential location of a facility. If $d_0\leqq d_1\leqq\cdots\leqq d_t$ are the distances from u to such locations in T from which u itself would be covered and if $d_s(s\geqq 1)$ is a distance from a vertex in T_j to u, then

$$\mathrm{INT}(T,d_s)=\min\Big[\mathrm{INT}(T,d_{s-1}),\ \mathrm{INT}(T_j,d_s-d(u,u_j))+\sum_{\substack{i=1\\ i\neq j}}^{k}\mathrm{EXT}(T_i,d_s+d(u,u_i))\Big].$$

For the routine EXT, let $A=\sum_{i=1}^{k}\mathrm{EXT}(T_i,r+d(u_i,u))$ if $r<r_u$, and $A=+\infty$ otherwise. Then, $\mathrm{EXT}(T,r)=\min[A,\mathrm{INT}(T,r)]$. It follows from the structure of these formulae that it takes $O(mk)$ to solve the problems at u for *all* values of r. The total effort is therefore $O(m\sum_v\deg(v))=O(nm)$.

Finally, consider the problem of maximizing the *net gain*; i.e., the total revenue (resulting from covering nodes) minus the total construction cost. This problem too is NP-hard on a general network since the minimum dominating set (see [GJ]) can be reduced to this problem by taking $w_j=2$, $c_j=1$, $j=1,\cdots,n$ and $B=k$. However, on a tree the solution is similar to that of the min-cost coverage of *all* vertices, i.e., the parameter p and the ALLOC routine can be eliminated. That leads to the same $O(nm)$ time bound.

Naturally, the same methods can be used to solve other types of problems defined on tree networks such as those which seek to locate facilities as far apart as possible from the various vertices. For instance, consider the problem of minimum coverage of obnoxious facilities. Here we are seeking to minimize the total weight of vertices who are "damaged" because an "obnoxious" facility [CG], [CT] is located too close, given that we have to locate p such facilities and the interfacility distances are also bounded from below. This can be solved in a way which is very similar to the one developed in the present paper.

Appendix 1. Linear programming considerations. Tamir [T] and Kolen [K] have recently shown that a fairly large class of location problems on trees can be solved by linear programming. Specifically, let $a_{ij} = 1$ if $d(i, j) < r_i$ and $a_{ij} = 0$ otherwise. Let $x_j = 1$ be interpreted as establishing a facility at j and $x_j = 0$ otherwise. Thus, the program

$$\text{Minimize} \sum_{j=1}^{n} c_j x_j$$

$$\text{s.t.} \quad Ax \geqq 1$$

$$x_j \in \{0, 1\}.$$

($A = (a_{ij})$) solves the problem of covering *all* vertices with minimum cost. It is known [G], [K], [T] that for a tree network the matrix A is balanced and hence the polytope $\{x : Ax \geqq 1, x \geqq 0\}$ has only integral extreme points. On the other hand, our coverage problem can be formulated as maximizing the number of vertices that would be covered by at most p facilities. Thus, if $y_i = 0$ is interpreted as "vertex i is covered" and $y_i = 1$ otherwise, then our problem is in fact

$$\text{Minimize} \sum_{i=1}^{n} y_i$$

$$\text{subject to } Ax + y \geqq 1$$

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

$$x_j, y_i \in \{0, 1\}.$$

Now, even though the underlying matrix in this problem, namely,

$$\begin{bmatrix} A & & I \\ 1 \cdots 1 & & 0 \end{bmatrix}$$

is still balanced, a linear programming solution may lead to a nonintegral solution, as we show in the following example. Consider the tree in Fig. 2. All weights $w_i = 1$ and r_i's and d_{ij}'s are shown in the figure. There are four potential points denoted by arrows (i.e., for every other point y there is one of the four points that covers at least those vertices covered by y).

Consider the problem of maximizing the number of vertices covered by two new facilities. With two "integral" facilities, namely, the center and another point of the four, we can cover at most nine vertices. However, by selecting one "half" of a facility to be located in each one of these four vertices, we manage to cover nine and a "half" vertices.

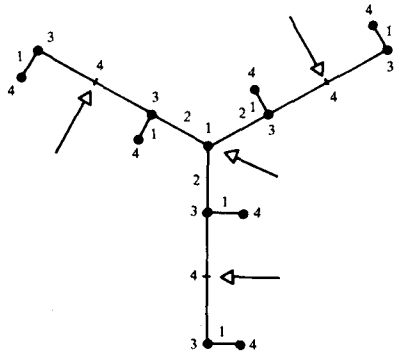

FIG. 2

Appendix 2. Threshold radii arising from distances to existing facilities. Suppose that the radii r_i are in fact the current distances from each vertex i to the nearest existing facility (which belongs to the first company) and the customer located at i would use a new facility if one were established by the second company at a distance less than r_i from i.

We claim that the problem can be decomposed in this case as follows. Consider the connected components of the tree induced by the locations of the existing facilities. Specifically, two points belong to the same component if there is no existing facility on the path which connects them. Let these components be denoted by $T_1, \cdots, T_k$. Obviously, a customer located in T_j would use a new facility only if it is located inside T_j. Thus, it suffices to consider maximum coverage problems on the components and then solve a resource allocation problem as follows. Suppose that $f_i(p_i)$ is the maximum gain possible by locating p_i new facilities in T_i. Then, the solution to our problem is by maximizing $\sum f_i(p_i)$ subject to $\sum p_i = p (p_i \geqq 0$ and integral). We note that in each component an existing facility is always located at a leaf. Furthermore, we can assume without loss of generality that every leaf contains a facility. For, assume on the contrary, that a customer i is located at a leaf in which no existing facility is located. Such a customer will switch to a new facility inside the component if and only if its unique neighbor does. Thus, we can eliminate the leaf i from the tree and add its weight to that of its neighbor. Continuing with this process, we eventually get components in which the existing facilities coincide with the leaves.

The case of a chain tree is extremely simple. Here we split the chain into subchains whose boundary points are the locations of the existing facilities. Each subchain should be assigned either 0, 1, or 2 new facilities. Thus, the resource allocation problem that has to be solved in this case is very simple and the problem can be solved by the greedy algorithm in linear time.

REFERENCES

[CG] R. L. CHURCH AND R. S. GARFINKEL, *Locating an obnoxious facility on a network*, Transport Sci., 1 (1978), pp. 107–118.

[CT] R. CHANDRASEKARAN AND A. TAMIR, *Locating obnoxious facilities*, Dept. Statistics, Tel-Aviv University, Tel Aviv, Israel, 1979.

[FJ] G. N. FREDERICKSON AND D. B. JOHNSON, *Optimal algorithms for generating quantile information in $X+Y$ and matrices with sorted columns*, Proc. 13th Ann. Conf. on Information Science and Systems, the Johns Hopkins Univ., Baltimore, 1979, pp. 47–52.

[GM] Z. GALIL AND N. MEGIDDO, *A fast selection algorithm and the problem of optimum distribution of effort*, J. Assoc. Comput. Mach., 26 (1979), pp. 58–64.

[GJ] M. R. GAREY AND D. S. JOHNSON, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, San Francisco, 1979.

[G] R. GILES, *A balanced hypergraph defined by subtrees of a tree*, Ars Combinatoria, 6 (1978), pp. 179–183.

[H1] S. L. HAKIMI, *Optimal location of switching centers and the absolute centers and medians on a graph*, Oper. Res., 12 (1964), pp. 450–459.

[H2] S. L. HAKIMI, *On locating new facilities in a competitive environment*, presented at ISOLDEII June 15–20, 1981, Skodsborg, Denmark.

[KH] O. KARIV AND S. L. HAKIMI, *An algorithmic approach to network location problems*, Part II: *p*-Medians, SIAM J. Appl. Math., 37 (1979), pp. 539–560.

[K] A. KOLEN, *An $O(nm)$ algorithm for the minimum cost covering problem on a tree with n vertices and m neighborhood subtrees*, ISOLDEII (see [H2]).

[T] A. TAMIR, *A class of balanced matrices arising from location problems*, Dept. Statistics, Tel Aviv University, Tel Aviv, Israel, 1980.