SIAM J. COMPUT.
Vol. 12, No. 4, November 1983

0097-5397/83/1204-0011 $01.25/0

NEW RESULTS ON THE COMPLEXITY OF p-CENTER PROBLEMS*

NIMROD MEGIDDO†‡ AND ARIE TAMIR†

Abstract. An $O(n \log^3 n)$ algorithm for the continuous p-center problem on a tree is presented. Following a sequence of previous algorithms, ours is the first one whose time bound in uniform in p and less than quadratic in n. We also present an $O(n \log^2 n \log\log n)$ algorithm for a weighted discrete p-center problem.

Key words. location, p-centers, parallel computation, tree partitioning, parametric combinatorial optimization

1. Introduction. The p-center problems are defined on a weighted undirected graph $G=(V,E)$. Each edge (i,j) has a positive length d_{ij}. An edge is identified with a line segment of length d_{ij} so that we can refer to any "point" on (i,j) at a distance of t from i and $d_{ij}-t$ from j $(0 \leqq t \leqq d_{ij})$. The set of all such points of the graph is denoted by A. If $x, y \in A$ then by $d(x,y)$ we mean the length of the shortest path from x to y.

The *continuous* p-center problem is to find p points $y_1, \cdots, y_p \in A$ so as to minimize

$$\max_{x \in A} \min_{1 \leqq j \leqq p} d(x, y_j).$$

Intuitively, we wish to locate p "centers" anywhere on the graph so as to minimize the maximum distance between any point and its respective nearest center. An optimal solution $y_1, \cdots, y_p$ is called a p-center and the corresponding largest distance is denoted r_p and is called the p-radius. Results on other versions of the p-center problem in which the supply points must be located on vertices can be found in [6], [7], [8], [9], [10], [11], [12], [13], [16]. Such problems have been shown to be NP-hard on general graphs [13] and it has been conjectured that the continuous p-center problem should be "difficult" on general graphs and, indeed, we include in the last section a proof of its NP-hardness. Our discussion here is hence limited to tree graphs. Our main result in this paper is a substantial improvement of the upper bound on the complexity of the continuous p-center problem on a tree. We devise in this paper an algorithm which runs in $O(n \log^3 n)$ time, $n=|V|$. As is apparent from Table 1, the

TABLE 1
Algorithms for the continuous p-center problem on a tree.

	Reference		Bound
(1)	Handler	[11], [12]	$O(n)$ for $p \leqq 2$
(2)	Chandrasekaran & Daughety	[2]	polynomial, not specified
(3)	Chandrasekaran & Tamir	[4]	$O(\min(n^2 \log^2 p, n^2 \log n + p \log^2 n))$
(4)	Megiddo, Tamir, Zemel & Chandrasekaran	[16]	$O(\min(n^2 \log p, pn \log^2 n))$
(5)	Chandrasekaran & Daughety	[3]	$O(n^2 \log p)$
(6)	Frederickson & Johnson	[6], [7], [8]	$O(n \min(p,n) \log(\max(\binom{p}{n}, \binom{n}{p})))$
(7)	Megiddo & Tamir		$O(n \log^3 n)$

* Received by the editors November 13, 1981, and in final form November 5, 1982.

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

‡ The work of this author was partially supported by the National Science Foundation under grant ECS-8121741. Currently visiting Department of Computer Science, Stanford University, Stanford, California 94305.

improvement is in two respects. First, our bound is the only one which is uniform with respect to p, while all previous algorithms run in time which tends to infinity with $\log p$, even when the tree is fixed. Secondly, all previous bounds are at least quadratic with n (for general p) while ours is $o(n^{1+\varepsilon})$ for any $\varepsilon > 0$; however, for values of p that are $O((\log n)^2)$, Frederickson and Johnson's algorithm is better.

In order to explain the contribution of the present paper, we start with an overview of the previous results. The first polynomial algorithm was given by Chandrasekaran and Daughety [2]. This algorithm is in fact an application of a general method of solving parametric combinatorial problems presented in Megiddo [14]. The parametrization enters here in the following way. Consider the problem $P(r)$ of locating a minimum number of "centers", $M(r)$, so that every point is within a distance r from at least one center. The p-center problem is to minimize the value of the parameter r subject to $M(r) \leqq p$. The function $M(r)$ is a step function and the problem $P(r)$ is solvable in $O(n)$ time for a fixed r [2]. The method presented in [14] simulates the computation of $M(r)$ where r belongs to a certain interval and is not just fixed at a unique value. The interval is repeatedly narrowed, always containing the minimal r such that $M(r) \leqq p$. It then finds that minimal r exactly. The details are further developed throughout the present paper.

Chandrasekaran and Tamir [4] proved that the jump points of $M(r)$ are of the form $d(x, y)/2l$ where l is integral and x, y are vertices of degree 1. In particular, the p-radius r_p belongs to the set

$$R = \left\{ \frac{d(x, y)}{2l} : x, y \in V, 1 \leqq l \leqq p \right\}.$$

Chandrasekaran and Tamir's algorithm is based on that fundamental result. Note that the cardinality of R is $O(n^2 p)$. However, R has a special structure which enables searching in R without generating the entire set in advance. Indeed, Megiddo, Tamir, Zemel and Chandrasekaran [16] found a succinct representation of all the distances $d(x, y)$, which allows for finding the kth longest path in the tree as well as solving discrete p-center problems in $O(n \log^2 n)$ time despite the cardinality of $\{d(x, y)\}$ being $O(n^2)$. This representation also yielded the $O(\min (n^2 \log p, pn \log^2 n))$ bound for the continuous p-center problem. Later, Frederickson and Johnson [5], [6], [7], [8] found an even better representation which yielded an $O(n \log n)$ algorithm for the unweighted discrete problems as well as the $O(n \min (n, p) \cdot \log (\max (p/n, n/p)))$ algorithm for the continuous p-center problem. Our algorithm in this paper is based on a more efficient search in the set R which exploits both the special structure of the set $\{d(x, y)\}$ and the monotonicity with respect to l (see the definition of R). The search employs parallel computation algorithms along the lines suggested in Megiddo [15].

Another related problem which we attack in the present paper is the following *weighted* discrete p-center problem on a tree. Assuming that every vertex i has a positive weight w_i associated with it, find p points $y_1, \cdots, y_p \in A$ so as to minimize

$$\operatorname*{Max}_{i \in V} \operatorname*{Min}_{1 \leqq j \leqq p} w_i d(i, y_j).$$

This problem is solved in [13] in $O(n^2 \log n)$ time. A better implementation yielding an $O(n^2)$ bound appears in [4]. The optimal value of the objective function (i.e. the analogue or r_p) is known [13] to belong to the set

$$\left\{ \frac{d(x, y)}{w_x^{-1} + w_y^{-1}} : x, y \in V \right\}$$

which is of cardinality $O(n^2)$. Applying methods similar to those used in the continuous case, we find the solution in this case in $O(n \log^2 n \log\log n)$ time. We remark that the case where the p-centers may be established only at the vertices is solved in [16] in $O(n \log^2 n)$ time.

2. The continuous problem.

2.1. An overview. It has been pointed out earlier that the p-radius is of the form $d(x, y)/2l$ where $x, y \in V$ and $1 \leqq l \leqq p$. It will be convenient for us to deal with the set R of all these candidates for r_p by looking at the functions $k_{xy}(r)$, defined for all $x, y \in V$:

$$k_{xy}(r) = \left\lfloor \frac{d(x, y)}{2r} \right\rfloor \qquad (r > 0).$$

Obviously, k_{xy} is a step function with jumps at $\lfloor d(x, y)/2l \rfloor$, $l = 1, 2, \cdots$. We will determine an interval $[a, b]$ such that $M(a) > p \geqq M(b)$ and such that all the k_{xy}'s are constant over $(a, b]$. These characteristics imply that $a < r_p \leqq b$ (since $r_p = \text{Min}\{r: M(r) \leqq p\}$) and hence $r_p = b$, since at least one of the functions k_{xy} must jump at r_p.

The determination of the final interval is carried out in two phases. During the first phase an interval $[a_0, b_0]$ is found such that $a_0 < r_p \leqq b_0$ and at least the $k_{ij}(r)$'s corresponding to *edges* (i, j) are constant over $(a_0, b_0]$. During the second phase the set of pairs (x, y) for which $k_{xy}(r)$ is constant is gradually increased (while the interval is gradually narrowed down) until we reach the final interval. The second phase is organized in the form of $O(\log n)$ stages determined by a centroid decomposition [16], [6] of the tree. During each stage we consider $O(n^2)$ pairs of vertices, however, only $O(\log^2 n)$ jump points are tested, i.e., $M(r)$ is evaluated at no more than $O(\log^2 n)$ jump points.

2.2. Phase 1. Let the tree be rooted at an arbitrary vertex u. We will use the convention that if (i, j) is an edge such that j is closer to u than i, then i belongs to the set of points of (i, j) while j does not. Thus every vertex except for u belongs precisely to one edge.

Consider the related problem $P(r)$ (see § 1). It is easy to verify the truth of the following:

ASSERTION 1. *In order for every point to be within a distance of r from at least one center, at least $k_{ij}(r)$ centers must be located on the edge (i, j) (including exactly one endpoint).*

ASSERTION 2. *In order to satisfy the requirement mentioned in Assertion 1 with respect to points of the edge (i, j) it is sufficient to allocate $k_{ij}(r) + 1$ centers to that edge.*

COROLLARY. *If $f(r) = \sum_{(i,j) \in E} k_{ij}(r)$ then*

$$f(r) \leqq M(r) \leqq f(r) + (n - 1).$$

Consider the case $r = r_p$. First, by definition $M(r_p) \leqq p$. We also claim that $p - (n-1) < M(r_p)$. For if $M(r_p) + (n-1) \leqq p$ then $r_p \leqq r_{M(r_p)+(n-1)}$ and $r_{M(r_p)+(n-1)} < r_{M(r_p)}$ since by adding one more center per edge the radius certainly decreases. On the other hand, obviously $r_{M(r_p)} = r_p$ and a contradiction has been reached. It now follows that

$$p - 2(n-1) < M(r_p) - (n-1) \leqq f(r_p) \leqq M(r_p) \leqq p.$$

For every r the function f jumps at r if and only if one of the k_{ij}'s jumps at r. Consider the number $r' = \frac{1}{2p} \sum_{(i,j) \in E} d(i, j)$. By the definition of f, $f(r') \leqq \sum_{(i,j) \in E} d(i, j)/2r' = p$. Also $f(r') > \sum_{(i,j) \in E} ((d(i,j)/2r') - 1) = p - (n-1)$. We have, thus,

obtained the following bounds,

$$p-2(n-1)<f(r_p)\leqq p,$$

$$p-(n-1)<f(r')\leqq p.$$

Compute $M(r')$ to determine whether $r'\geqq r_p$ or $r'<r_p$. Suppose first that $r'\geqq r_p$ and imagine that we continuously decrease r, starting from $r=r'$ until we reach $f(r)=p+1$. It follows from the above discussion that we will approach at most $p+1-f(r')<n$ jump points of f. All these jumps are at points of the form $\frac{1}{2}d_{ij}/(k_{ij}(r')+l)$ where $1\leqq l\leqq p+1-f(r')$. As a matter of fact, these jumps occur at the $p+1-f(r')$ largest elements of the set

$$R_0=\left\{\frac{1}{2}\frac{d_{ij}}{k_{ij}(r')+l}:(i,j)\in E,\ l=1,\cdots,p+1-f(r')\right\}.$$

Since R_0 is naturally partitioned into $n-1$ sorted subsets, corresponding to the $n-1$ edges, we can find and sort all these jumps in $O(n+(p+1-f(r'))\log n)=O(n+\min(n,p)\log n)$ time using a standard priority queue [1]. Similarly, if $r'<r_p$ imagine that we continuously increase r, starting with $r=r'$ until we reach $f(r)=\max(0,p-2(n-1))$. It follows that we will approach at most $f(r')-\max(0,p-2(n-1))\leqq\min(2n,p)$ jump points of f. Using the scheme of the previous case, all these jumps are found in $O(n+(f(r')-\max(0,p-2(n-1))\log n))$ time. In any case, the effort so far is $O(n+\min(n,p)\log n)$.

Once we have all the jumps in the relevant domain (depending on whether $r'<r_p$ or $r'\geqq r_p$), we can search for two consecutive jump points a_0, b_0 such that $a_0<r_p\leqq b_0$. The search amounts to $O(\log(\min(n,p)))$ evaluations of the function $M(r)$. This completes Phase 1, at the end of which we have the interval $(a_0,b_0]$ such that $k_{ij}(r)$, $(i,j)\in E$, is constant for all $r\in(a_0,b_0]$. The total effort during this phase uses $O(q\log n+n\log q)$ time where $q=\min(n,p)$. We note in passing that at this point, when we have $k_{ij}(r_p)$, $(i,j)\in E$, r_p can be found in $O(n^2)$ time using the method of [14] as described in § 1; this is because the evaluation of $M(r)$ takes $O(n)$ time and the method of [14] always leads to no more than the squared amount of time of the master algorithm used. We note that the number of centers on (i,j) is either $k_{ij}(r_p)$ or $k_{ij}(r_p)+1$ so that the value of $M(r)$ is computed $O(n)$ times.

2.3. Phase 2. When the second phase starts, we have an interval $(a_0,b_0]$ such that $a_0<r_p\leqq b_0$, over which the functions $k_{ij}(r)$ (for $(i,j)\in E$) are constant; we have to narrow this interval down until all the $k_{xy}(r)$'s become constant. We will describe the algorithm here in a recursive way.

Given is a tree T together with an interval $(\alpha,\beta]$ such that $\alpha<r_p\leqq\beta$ and the functions $k_{ij}(r)$ $((i,j)\in E)$ are known to be constant over $(\alpha,\beta]$. Our recursive routine in the present subsection will produce the following: A subinterval $(\alpha',\beta']$ $(\alpha\leqq\alpha'<r_p\leqq\beta'\leqq\beta)$ will be found such that all the functions $k_{xy}(r)$ (for any vertices x, y of T) will be constant on $(\alpha',\beta']$.

Let c be a centroid of T, i.e., T can be represented as a union of two subtrees T_1 and T_2 whose only common vertex is c, such that each has at most $\frac{2}{3}$ of the vertices of T. Such a vertex can be found in linear time [10], [13], [16]. We call this partition a centroid decomposition [7], [16]. The decomposition may proceed into the subtrees and their components and so on, and that whole hierarchy (done in $O(\log n)$ phases) will be called a total centroid decomposition.

We first apply the routine recursively to the trees T_1 and T_2. We thus obtain an interval $(\alpha_0,\beta_0]$ $(\alpha_0<r_p\leqq\beta_0)$ such that $k_{xy}(r)$ is constant on $(\alpha_0,\beta_0]$ whenever x and

y are in the same subtree (either T_1 or T_2). It takes linear time to find all the distances $d(x, c)$ and $d(c, y)$ $(x \in T_1, y \in T_2)$, and hence the constant value over $(\alpha_0, \beta_0]$ of $k_{xc}(r)$ for $x \in T_1$ and of $k_{cy}(r)$ for $y \in T_2$ is also assumed to be known. Moreover, $k_{xc}(r) + k_{cy}(r) \leqq k_{xy}(r) \leqq k_{xc}(r) + k_{cy}(r) + 1$. Thus, the function $k_{xy}(r)$ $(x \in T_1, y \in T_2)$ may have at most one jump in the interval $(\alpha_0, \beta_0]$, namely, when it jumps from $k_{xc}(r) + k_{cy}(r) + 1$ to $k_{xc}(r) + k_{cy}(r)$. This occurs at the value

$$\frac{1}{2} \cdot \frac{d(x, y)}{k_{xc}(r) + k_{cy}(r) + 1}.$$

Denote $a_x = \frac{1}{2}d(x, c)$, $b_y = \frac{1}{2}d(c, y)$, $c_x = k_{xc}(r)$ and $d_y = k_{cy}(r) + 1$. As a matter of fact, we now have to search for r_p within the set

$$R' = \left\{ \frac{a_x + b_y}{c_x + d_y} : x \in T_1, y \in T_2 \right\}.$$

In other words, we look for $\alpha', \beta' \in R' \cup \{\alpha_0, \beta_0\}$ such that $\alpha_0 \leqq \alpha' < r_p \leqq \beta' \leqq \beta_0$ and $(\alpha', \beta') \cap R' = \varnothing$. It is explained in the Appendix how to perform this search in $O(n \log^2 n)$ time. It follows from the definition of the centroid that the total centroid decomposition consists of $O(\log n)$ phases. Therefore our routine runs in $O(n \log^3 n)$ time.

This implies that the continuous p-center problem is solvable by our algorithm in $O(n \log^3 n)$ time.

3. The weighted discrete case. As pointed out in the Introduction the weighted discrete case is equivalent to searching for the optimal value, denoted here by r^*, in the set $R^* = \{d(x, y)/(w_x^{-1} + w_y^{-1}) : x, y \in V\}$. The solution process here is quite similar to that of the continuous case. It is easy to see that an adaptation of Phase 2 to the present problem solves the problem in $O(n \log^3 n)$. However, a further improvement is yet possible.

All the intervertex distances $d(x, y)$ can be organized in the form $d(x, y) = d(x, c_i) + d(c_i, y)$ where c_i is a centroid in the total decomposition of our tree (see § 2.3 and [7], [16]). Let I denote the index set of these centroids c_i and let T_i^1 and T_i^2 denote the two subtrees given by the decomposition of a previous subtree as induced by c_i (see § 2.3). We then have to search for r^* in a collection of sets of the form

$$R_i = \left\{ \frac{d(x, c_i) + d(c_i, y)}{w_x^{-1} + w_y^{-1}} : x \in T_i^1, y \in T_i^2 \right\},$$

$i \in I$, where $\sum_{i \in I} |T_i^1| = O(n \log n)$ and $\sum_{i \in I} |T_i^2| = O(n \log n)$.

The search procedure employed here is similar to the two-stage scheme described in the Appendix. In the first stage we identify an interval $[s_1, t_1]$ such that $s_1 < r^* \leqq t_1$ and such that for each i, the linear order induced on the vertices of T_i^1 by the numbers $w_x^{-1} r - d(x, c_i)$ is independent of r provided $r \in [s_1, t_1]$. For each $i \in I$ we employ $|T_i^1|$ "processors" for sorting $\{w_x^{-1} r - d(x, c_i) : x \in T_i^1\}$. We use Valiant's [18] parallel sorting algorithm which takes $O(\log |T_i^1| \log \log |T_i^1|)$ time. Thus, $\sum |T_i^1| = O(n \log n)$ processors suffice for parallel sorting of each of the $|I|$ sequences $\{w_x^{-1} r - d(x, c_i) : x \in T_i^1\}$ in $O(\log n \log \log n)$ time.

As in the Appendix, a single step of the parallel sorting scheme gives $n \log n$ critical values for the parameter r. Given these $n \log n$ values, and an interval $(s_0, t_0]$ which contains r^*, we can in $O(n \log n)$ time (as in the Appendix) narrow down the interval so that it will still contain r^* but none of the above $n \log n$ critical values in

its interior. Hence, the total time for the first stage is $O((\sum_{i\in I}|T_i^1|+n\log n)\log n\log\log n)=O(n\log^2 n\log\log n)$.

The second stage is the same as Stage 2 of the Appendix, with the exception that here we use $\sum_{i\in I}|T_i^2|=O(n\log n)$ processors. However, the total effort for this stage still remains $O(n\log^2 n)$. Thus, the total effort involved in finding r^* is $O(n\log^2 n\log\log n)$.

We note that if the tree is a path connecting two vertices v_1 and v_n then r^* is an element of the set

$$R=\left\{\frac{d(x,v_1)-d(y,v_1)}{w_x^{-1}+w_y^{-1}}: x,y\in V\right\}.$$

Therefore, using the scheme described in the Appendix, r^* is found in $O(n\log^2 n)$ time.

4. NP-hardness of the continuous p-center problem. The result of the present section was not particularly hard to achieve even though it has not been previously proved to our knowledge. The NP-completeness of other versions of p-center problems was proved in [13].

In order to establish that the p-center problem is NP-hard on general graphs, we will reduce the minimum dominating set problem (see [9]) to the following problem: Given a graph $G=(V,E)$, all of whose edges are of unit length, find out whether there exists a set of p "centers" (anywhere on the edges of G) such that every point of G is within a distance of 2 from some center. We should remark that it may be necessary to locate centers in the interior of an edge in order to solve the latter problem (see Fig. 1).

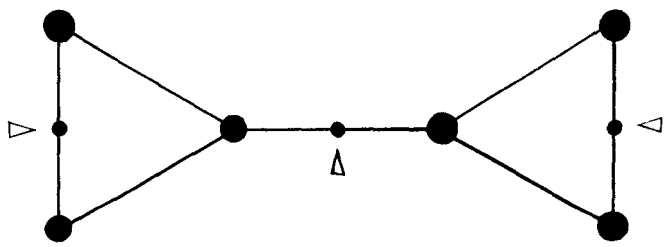

FIG. 1

The reduction is as follows. Suppose that $G=(V,E)$ is a graph for which we have to recognize whether there exists a dominating set of cardinality p. Let G' denote a graph obtained from G by adjoining one vertex v' per each vertex v of G, such that v' is adjacent only to v. All the edges of G' will be of unit length. We consider the problem of p centers on G'. Obviously, a dominating set for G will solve the center problem. Conversely, suppose that we have p centers on G' such that every vertex is at distance not greater than 2 from some center. Clearly, every vertex of the original graph G is within unit distance from some center. If we translate every center to one of its respective nearest vertices, then every vertex of G still has a center within unit distance from it. We thus have a dominating set of the appropriate cardinality.

Finally, we remark that since the dominating set problem is NP-complete even on bipartite planar graphs of maximum degree 3 (see [13, proof of Lemma 3.1]), then the above reduction implies NP-hardness of the continuous p-center problem even on bipartite planar graphs of maximum degree 4.

Appendix. Searching in $\{(a_i+b_j)/(c_i+d_j)\}$. In this appendix we describe how to search for the number r_p within a set of the form $S=\{(a_i+b_j)/(c_i+d_j): 1\leqq i,j\leqq n\}$. Thus there will be given $4n$ numbers, a_i, b_j, c_i, d_j $(1\leqq i,j\leqq n)$, and we will have to find two elements s, $t\in S$ such that $s<r_p\leqq t$ and no element of S is strictly between s and t.

We first note that the set S consists of the points of intersection of straight lines $y=(c_ix-a_i)+(d_jx-b_j)$ with the x-axis. The search will be conducted in two stages. During the first stage we will identify an interval $[s_1, t_1]$ such that $s_1<r_p\leqq t_1$ and such that the linear order induced on $\{1,\cdots,n\}$ by the numbers c_ix-a_i is independent of x provided $x\in[s_1, t_1]$. The rest of the work is done in Stage 2.

Stage 1. An equivalent description of Stage 1 is that we search for r_p among the points of intersection of lines $y=c_ix-a_i$ with each other. The method was introduced in [15]. It is based on Preparata's [17] parallel sorting scheme. This scheme employs $n\log n$ "processors" during $O(\log n)$ steps. Imagine that we sort the set $\{1,\cdots,n\}$ by the (c_ix-a_i)'s, where x is not known yet. Whenever a processor in Preparata's scheme has to compare some c_ix-a_i with c_jx-a_j, we will in our algorithm compute the critical value $x_{ij}=(a_i-a_j)/(c_i-c_j)$. Thus, a single step in Preparata's scheme gives rise to the production of $n\log n$ points of intersection of lines $y=c_ix-a_i$ with each other. Given these $n\log n$ points and an interval $(s_0, t_0]$ which contains r_p, we can in $O(n\log n)$ time narrow down the interval so that it will still contain r_p but no intersection point in its interior. This requires the finding of medians in sets of cardinalities $n\log n$, $\frac{1}{2}n\log n$, $\frac{1}{4}n\log n$, $\cdots$ plus $O(\log n)$ evaluations of $M(r)$. We then proceed to the next step in Preparata's scheme. We note that since the outcomes of the comparisons so far are independent of x in the updated interval, we can proceed with the sorting even though x is not specified. The effort per step is hence $O(n\log n)$ and the entire Stage 1 takes $O(n\log^2 n)$ time.

Stage 2. When the second stage starts we can assume without loss of generality that for $x\in[s_1, t_1]$ $c_ix-a_i\leqq c_{i+1}x-a_{i+1}$, $i=1,\cdots,n-1$. Let j $(1\leqq j\leqq n)$ be fixed and consider the set S_j of n lines $S_j=\{y=c_ix-a_i+d_jx-b_j, i=1,\cdots,n\}$. Since S_j is "sorted" over $[s_1, t_1]$, we can obviously find in $O(\log n)$ evaluations of $M(r)$ a subinterval $[s_1^j, t_1^j]$ such that $s_1^j<r_p\leqq t_1^j$, and such that no member of S_j intersects the x-axis in the interior of this interval. We will, however, work on the S_j's in parallel. Specifically, there will be $O(\log n)$ steps. During a typical step, the median of the remainder of every S_j is selected (in constant time) and its intersection point with the x-axis is computed. The set of these n points is then searched for r_p and the interval is updated accordingly. This enables us to discard a half from each S_j. Clearly a single step lasts $O(n\log n)$ time and the entire stage is carried out in $O(n\log^2 n)$ time.

At the end of the second stage we have the values $\{s_1^j\}$ and $\{t_1^j\}$, $j=1,\cdots,n$. Defining $s=\max_{1\leqq j\leqq n}\{s_1^j\}$ and $t=\min_{1\leqq j\leqq n}\{t_1^j\}$ we obtain $s<r_p\leqq t$, and no element of S is strictly between s and t.

REFERENCES

[1] A. V. AHO, J. E. HOPCROFT AND J. D. ULLMAN, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, MA, 1976.

[2] R. CHANDRASEKARAN AND A. DAUGHETY, *Problems of location on trees*, Discussion Paper, 357, Center for Mathematical Studies in Economics and Management, Northwestern Univ., Evanston, IL, 1978.

[3] ———, *Location on tree networks: p-center and n-dispersion problems*, Math. Oper. Res., 6(1981), pp. 50–57.

[4] R. CHANDRASEKARAN AND A. TAMIR, *An $O((n\log p)^2)$ algorithm for the continuous p-center on a tree*, SIAM J. Alg. Disc. Meth., 1(1980), pp. 370–375.

[5] G. N. FREDERICKSON AND D. B. JOHNSON, *Generalized selection and ranking*, in Proc. 12th Annual ACM Symposium on Theory of Computing, Los Angeles, April 1980, Assoc. Comput. Mach., New York, 1980, pp. 420–428.

[6] ———, *Generating and searching sets induced by networks*, in Proc. 7th EATCS Colloquium on Automatic Languages and Programming, Noordwijkerhout, the Netherlands, July 1980, Lecture Notes in Computer Science, 85, Springer-Verlag, Berlin, 1980, pp. 221–233.

[7] ———, *Generating and searching sets for path selection and p-center location*, Computer Science Department, Pennsylvania State Univ., University Park, PA, August 1981.

[8] ———, *Finding k-th paths and p-centers by generating and searching good data structures*, J. Algorithms, to appear.

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

[10] A. J. GOLDMAN, *Optimal center location in simple networks*, Transportation Sci., 5 (1971), pp. 212–221.

[11] G. Y. HANDLER, *Minimax location of a facility in an undirected tree graph*, Transportation Sci., 7 (1973), pp. 287–293.

[12] ———, *Finding two-centers of a tree: The continuous case*, Transportation Sci., 12(1978), pp. 1–15.

[13] O. KARIV AND S. L. HAKIMI, *An algorithmic approach to network location problems, Part I. The p-centers*, SIAM J. Appl. Math, 37(1979), pp. 513–538.

[14] N. MEGIDDO, *Combinatorial optimization with rational objective function*, Math. Oper. Res., 4(1979), pp. 414–424.

[15] ———, *Applying parallel computation algorithms in the design of serial algorithms*, in Proc. 22nd Annual IEEE Symposium on Foundations of Computer Science, 1981, IEEE Computer Society Press, Los Angeles, 1981, pp. 399–408; J. Assoc. Comput. Math., to appear.

[16] N. MEGIDDO, A. TAMIR, E. ZEMEL AND R. CHANDRASEKARAN, *An $O(n \log^2 n)$ algorithm for the k-th longest path in a tree with applications to location problems*, this Journal, 10(1981), pp. 328–337.

[17] F. P. PREPARATA, *New parallel-sorting schemes*, IEEE Trans. Comput., C-27(1978), pp. 669–673.

[18] L. G. VALIANT, *Parallelism in comparison problems*, this Journal, 4(1975), pp. 348–355.