Optimally Locating a Structured Facility of a Specified Length in a Weighted Tree Network*

Shan-Chyun Ku and Biing-Feng Wang

Department of Computer Science, National Tsing Hua University
Hsinchu, Taiwan 30043, Republic of China

Abstract

In this paper, we propose efficient parallel algorithms on the EREW PRAM for optimally locating in a weighted tree network a tree-shaped facility of a specified length. Two optimization criteria are considered: minimum eccentricity and minimum distancesum. Let n be the number of vertices in the tree network. Both the two algorithms we shall propose take $O(\log n \log\log n)$ time using $O(n)$ work.

1. Introduction

Optimally locating a facility in a network is an important problem in the fields of transportation and communication [10]. The criteria for optimality extensively studied in the literature are the *minimum eccentricity* criterion in which the distance to the farthest vertex from the facility is minimized and the *minimum distancesum* criterion in which the total distance to the vertices from the facility is minimized.

Traditionally, network location theory has been concerned with the optimal location of a single-point facility. In [9], Slater extended the network location theory to include a facility that is not merely a single point but a path. This extension has practical applications in improving a stretch of road in a highway and in establishing a high-speed transmission line in a communication network. Slater's work confined to tree networks. A path in a tree network with the minimum distancesum is defined as a *core*.

Slater placed no constraint on the length of the path selected as the core. Minieka and Patel [5] first studied the problem of finding in a tree network a core of a specified length. Later, in [4], Minieka extended the studies in [5] to consider the problems of optimally locating in a tree network a path and a tree of a specified length. In Minieka's paper, two kinds of facilities are discussed: paths and trees; and four optimization criteria are considered: minimum eccentricity, minimum distancesum, maximum eccentricity, and maximum distancesum. Totally, there are eight different problems, including the one proposed in [5]. Minieka showed that all the eight problems can be solved in polynomial time except the one of finding the maximum distancesum tree of a specified length, which was proved to be NP-hard later in [8]. Recently, in [6,7], Peng and Lo have proposed efficient parallel algorithms on the EREW PRAM for optimally locating in a tree network a path and a tree of a specified length. They considered all the eight problems studied in Minieka's paper. In [6,7], it was assumed that the tree network is unweighted. Peng and Lo's results are summarized in Table 1. Note that under the assumption that the tree network is unweighted, the problem of finding a maximum distancesum tree of a specified length is not NP-hard.

In this paper, efficient parallel algorithms on the EREW PRAM are proposed for finding in a tree network the minimum eccentricity tree and the minimum distancesum tree of a specified length. Edges in the tree network have arbitrary positive lengths. Thus, as compared with Peng and Lo's results, our algorithms are more general. Besides, our algorithms are more efficient from the aspect of work. Both the two algorithms we shall propose take $O(\log n \log\log n)$ time using $O(n)$ work. In the sequential case, our algorithms give improvements on Minieka's $O(n^2)$ time algorithms in [4].

The remainder of this paper is organized as follows. In the next section, notation and preliminary results are presented. Then, parallel algorithms for finding the minimum eccentricity tree and the minimum distancesum tree are proposed in sections 3 and 4, respectively. Finally, in section 5, some concluding remarks are given.

2. Notation and Preliminary Results

Let $T=(V, E)$ denote the tree network under consideration, where V is the vertex set and E is the edge set. Let $n=|V|$. The tree network T is undirected. The n-1 edges in E are labeled with 1, 2, ..., and n-1, respectively. Let $W(e)$ denote the length of an edge $e \in E$.

A *tree-shaped facility* in T is a subtree in T, possibly with partial edges. The *length* of a facility denotes the sum of the lengths of all the edges and partial edges in the facility. Let l be the specified length of the facility to be

* This research is supported by the National Science Council of the Republic of China under grant NSC-87-2213-E-007-066.

located in T.

For any two points a and b on the edges of T, the distance of a and b, denoted by $d(a, b)$, is the length of the unique path connecting a and b. The distance from a vertex to a facility is defined as the shortest distance from the vertex to any point in the facility. We denote $d(v, F)$ as the distance from a vertex v to a facility F. The *eccentricity* of a facility is defined as the distance from the farthest vertex to the facility. The *distancesum* of a facility is defined as the total distance to the vertices from the facility. The eccentricity and distancesum of a facility F are denoted by $ECC(F)$ and $Sum(F)$ respectively.

Using the Euler-tour technique and tree-contraction [3], the following two lemmas can be easily obtained on the EREW PRAM.

Lemma 1 [3]: An undirected tree can be oriented into a rooted tree with a specified root in $O(\log n)$ time using $O(n)$ work.

Lemma 2 [3]: For each vertex v in a rooted tree, in $O(\log n)$ time using $O(n)$ work, we can compute the depth of v (the distance from v to the root), the height of the subtree rooted at v, and the number of vertices contained in the subtree rooted at v.

In the algorithms we shall propose in the next two sections, T will be oriented into a rooted tree frequently. In case T is oriented into a rooted tree, for each vertex v in T, we denote $p(v)$ and $depth(v)$ as the parent and the depth of v respectively. And, we denote $height(v)$ and $size(v)$ as the height of the subtree rooted at v and the number of vertices contained in the subtree rooted at v respectively.

The *center* of T is the unique point of T whose eccentricity is minimum. A *median* of T is any point of T whose distancesum is minimum. In [6], Peng and Lo had proposed parallel algorithms for finding the center and a median of an unweighted tree. Their algorithms can be easily extended to a weighted tree to obtain the following lemma.

Lemma 3: Finding the center and a median of a tree can be done in $O(\log n)$ time using $O(n)$ work.

3. Minimum Eccentricity Tree

Trivially, l is feasible only when l is not larger than the total length of the edges in T. Let c be the center of T. For easy discussion, we assume that c is located at a vertex of T. (In case c is a point on an edge (u, v), we can simply introduce a new vertex at c and split the edge (u, v) into two edges (u, c) and (c, v).) It was shown by Minieka [4] that for any feasible l, the minimum eccentricity tree of a length l is unique and contains the center c. Throughout this section, we assume that T is rooted at c. Note that since c is the center of T, $ECC(c)=height(c)$. For each edge $e=(u, p(u))$ in T, we define the function $R(e)$ to be $R(e)=height(u)+W(e)$. Note that $R(e)$ is the distance from $p(u)$ to the farthest leaf in the subtree rooted at u. We call $R(e)$ the *importance* of e. We say that an edge e is *more important than* another edge g iff (1) $R(e)>R(g)$ or (2) $R(e)=R(g)$ and e is with a larger label than g. The *rank* of an edge $e \in E$, denoted by $rank(e)$, is the number of edges in E that are more important than or as important as e. In other words, $rank(e)=i$ iff e is the i-th most important one among the edges in E, $1 \le k \le n-1$. Throughout this subsection, we denote a_k, $1 \le k \le n-1$, as the k-th most important one among the edges in E. Besides, for easy description, we denote a_n as a null edge with importance 0.

For each non-leaf vertex v in T, let $M(v)$ denote the most important one among the edges connecting v and v's children. Clearly, we have the following lemma.

Lemma 4: For each non-leaf vertex v in T, $height(v)=R(M(v))$.

Let F_1 be the tree-shaped facility that contains only the vertex c. And, for $k=2, 3, ..,$ and n, let F_k be the tree-shaped facility obtained from F_{k-1} by adding a_{k-1}. Note that F_k contains all the edges that are more important than a_k in E. And, a_k is the most important one in E that is not contained in F_k. Also note that $F_n=T$.

Lemma 5: $ECC(F_k)=R(a_k)$, $1 \le k \le n$.

Proof: Since $F_n=T$, we have $ECC(F_n)=R(a_n)=0$. In the following, we show that the eccentricity of F_k is $R(a_k)$, for $1 \le k \le n-1$.

Let x be a vertex in T that is not in F_k and y be the vertex in F_k that is closest to x. Let w be the second last vertex on the path from x to y and T_w be the subtree rooted at w. Note that $y=p(w)$, (w, y) is not in F_k, and x is a vertex in T_w. Since $R((w, y))$ is the distance from y to the farthest leaf in T_w, we have $d(x, y) \le R((w, y))$. Clearly, the distance $d(x, y)$ is maximized by letting $(w, y)=a_k$ and x be the farthest leaf from y in T_w. Q.E.D.

An edge $e=(u, p(u))$ in F_k is *external* if u is a leaf in F_k, and is *internal* otherwise. Clearly, an edge $e=(u, p(u))$ in E is an external edge in F_k iff e is more important than a_k and all the edges connecting u and u's children are not.

Let $g=(u, p(u))$ be an external edge in F_k. Let q be the point on g that is at a distance of $R(a_k)$-$height(u)$ from u. Note that $R(a_k)$-$height(u) \ge 0$, since $height(u)=R(M(u))$ and $M(u)$ is not more important than a_k. Also note that $W(g)$, the length of g, is not smaller than $R(a_k)$-$height(u)$, since $height(u)+W(g) = R(g) \ge R(a_k)$. Let T_u be the subtree rooted at u. Clearly, the distance from q to the farthest vertex in T_u is $height(u)+(R(a_k)-height(u))=R(a_k)$. Thus, removing the partial edge (u, q) from F_k does not increase the eccentricity. Let G_k, $1 \le k \le n$, be the rooted tree that is obtained from F_k by cutting down each external edge $g=(u, p(u))$ by a length of $R(a_k)$-$height(u)$. We have $ECC(G_k)=R(a_k)$. Let I_k and X_k be the sets of internal edges and external edges in F_k respectively. The length of

G_k, denoted by L_k, is

$$L_k = \sum_{e\in I_k} W(e) + \sum_{e=(u,p(u))\in X_k} (W(e) - (R(a_k) - height(u)))$$

$$= \sum_{e\in I_k} W(e) + \sum_{e=(u,p(u))\in X_k} (R(e) - R(a_k))$$

(since $R(e)=height(u)+W(e)$.)

$$= \sum_{e\in I_k} W(e) + \sum_{e\in X_k} R(e) - |X_k| R(a_k) .$$

Clearly, removing any partial edge from G_k will increase the eccentricity. We have the following lemma.

Lemma 6: G_k is the unique minimum eccentricity tree of length L_k in T. The eccentricity of G_k is $R(a_k)$.

Lemma 7: Let z be the integer satisfying $L_{z-1} \le l < L_z$. The length of each external edge in G_z is not smaller than $(L_z-l)/|X_z|$.

Proof: Let $g=(u, p(u))$ be an external edge in F_z. To obtain G_z from F_z, g is cutting down by a length of $(R(a_z)-height(u))$. Let g' be the partial edge leftover after g is cutting down. The length of g' is $W(g)-(R(a_z)-height(u))$, which is equal to $R(g)-R(a_z)$. Since a_{z-1} is the edge with the smallest importance in F_z, we have $R(g)\ge R(a_{z-1})$. Thus, each external edge in G_z is with a length not smaller than $R(a_{z-1})-R(a_z)$. To complete the proof, in the following, we show that $R(a_{z-1})-R(a_z) \ge (L_z-l)/|X_z|$.

Let Q be the tree-shaped facility obtained from G_z by cutting down each external edge by a length of $R(a_{z-1})-R(a_z)$. There are $|X_z|$ external edges in G_z. Thus, the length of Q is $L_z-|X_z|(R(a_{z-1})-R(a_z))$. The eccentricity of Q is $ECC(G_z)+(R(a_{z-1})-R(a_z)) = R(a_{z-1})$. Removing any partial edge from Q will increase the eccentricity. Thus, from lemma 6, we can easily conclude that $Q=G_{z-1}$ and $L_z-|X_z|(R(a_{z-1})-R(a_z)) = L_{z-1}$. And thus, we have $(R(a_{z-1})-R(a_z)) = (L_z-L_{z-1})/|X_z| \ge (L_z-l)/|X_z|$ (since $L_{z-1}\le l$.). The lemma holds. Q.E.D.

Lemma 7 provides the basis of our following algorithm for finding the minimum eccentricity tree of a specified length l.

ALGORITHM Min_Ecc_Tree($T=(V, E)$, l)

Step 1: Compute $L_n=\sum_{e\in E} W(e)$. If $l>L_n$ return 'no solution'. Otherwise, if $l=L_n$ return T as the minimum eccentricity tree.

Step 2: Find the center c of T, and then orient T into a rooted tree with root c.

Step 3: Determine $height(v)$ for each vertex v in V. And then, compute $R(e)$ for each edge e in E.

Step 4: For each non-leaf vertex v in V, determine $M(v)$.

Step 5: Find the edge w in $E\cup\{a_n\}$ satisfying $L_{rank(w)-1} \le l < L_{rank(w)}$.

Step 6: // Construct the minimum eccentricity tree. //

6.1: Compute $F_{rank(w)}$ as $\{e \mid e\in E$ and e is more important than $w\}$.

6.2: For each edge e in $F_{rank(w)}$, determine whether it is an external edge.

6.3: Obtain the minimum eccentricity tree of a length l from $F_{rank(w)}$ by cutting down each external edge $g=(u, p(u))$ by a length of $R(w)-height(u)+(L_{rank(w)}-l)/|X_{rank(w)}|$. And then, compute the minimum eccentricity as $R(w)+(L_{rank(w)}-l)/|X_{rank(w)}|$.

The correctness of the above algorithm is ensured by lemmas 5~7. The parallel running time of the algorithm is discussed as follows.

Step 1 needs to sum up n-1 values. Thus, it requires $O(\log n)$ time using $O(n)$ work. By lemmas 1~3, steps 2 and 3 take $O(\log n)$ time using $O(n)$ work. Step 4 can be implemented by finding for each vertex v in T the second most important edge among the edges incident to v, which are stored in the adjacent list of v. Note that the most important edge in the list is $(v, p(v))$. With the help of Cole and Vishkin's optimal algorithm for computing the prefix sums of a linked list [2], step 4 can be done in $O(\log n)$ time using $O(n)$ work. Step 5 is the most critical step of algorithm Min_Ecc_Tree. We will discuss its running time later. In step 6.1, we determine for each edge e in E whether it is more important than w. Since broadcasting of w is needed, step 6.1 requires $O(\log n)$ time using $O(n)$ work. An edge $g=(u, p(u))$ in $F_{rank(w)}$ is external iff $M(u)$ is not in $F_{rank(w)}$. Thus, step 6.2 requires only $O(1)$ time using $O(n)$ work. In step 6.3, we need to compute the values of $|X_{rank(w)}|$ and $L_{rank(w)}$. Clearly, the computation can be done in $O(\log n)$ time using $O(n)$ work. Therefore, except step 5, all steps of algorithm Min_Ecc_Tree can be performed in $O(\log n)$ time using $O(n)$ work.

In the following, we discuss the running time of step 5. First, we give a lemma that is useful for computing L_k's.

Lemma 8: Let $a_{k-1}=(u, p(u))$. If $p(u)$ is the root c or $a_{k-1} \ne M(p(u))$, we have $I_k=I_{k-1}$ and $X_k=X_{k-1}\cup\{a_{k-1}\}$. Otherwise, we have $I_k=I_{k-1}\cup\{g\}$ and $X_k=X_{k-1}\cup\{a_{k-1}\}-\{g\}$, where $g=(p(u), p(p(u)))$.

Proof: By definition, from F_{k-1}, we can obtain F_k by adding a_{k-1}. Clearly, in F_k, a_{k-1} is an external edge. Thus, if $p(u)$ is the root c, we can easily conclude that $I_k=I_{k-1}$ and $X_k=X_{k-1}\cup\{a_{k-1}\}$. In the following, we assume that $p(u)$ is not the root c. Let g be the edge connecting $p(u)$ and $p(u)$'s parent. Since $R(g)>R(a_{k-1})$, g is in F_{k-1}. If g is an external edge in F_{k-1}, after a_{k-1} is added to obtain F_k, it becomes an internal edge. Thus, if g is an external edge in F_{k-1}, we have $I_k=I_{k-1}\cup\{g\}$ and $X_k=X_{k-1}\cup\{a_{k-1}\}-\{g\}$. Otherwise, we have $I_k=I_{k-1}$ and $X_k=X_{k-1}\cup\{a_{k-1}\}$. It is not difficult to see that g is an

external edge in F_{k-1} iff $a_{k-1}=M(p(u))$. Q.E.D.

In the proof of lemma 7, we have shown that $L_k=L_{k-1}+|X_k|(R(a_{k-1})-R(a_k))$. On the basis of the equation and lemma 8, a simple way to implement step 5 of algorithm Min_Ecc_Tree is as follows.

Step 5:

Step 5.1: For each edge $e=(u, p(u))$, if $p(u)$ is the root c or $e \neq M(p(u))$, set $delta\_xn_e=1$; otherwise set $delta\_xn_e=0$. (Note that by lemma 8, $|X_k|=\sum_{1\le i\le k-1} delta\_xn_{a_i}$.)

Step 5.2: Obtain the sequence $(a_1, a_2, ..., a_n)$ by sorting the edges in $E\cup\{a_n\}$.

Step 5.3: Compute the values of $|X_k|$'s as $\sum_{1\le i\le k-1} delta\_xn_{a_i}$'s. And then, compute the values of L_k's as $\sum_{1\le i\le k-1}|X_i|(R(a_{i-1})-R(a_i))$'s.

Step 5.4: Set w as the edge a_t with the property that $L_{t-1}\le l< L_t$.

Step 5.1 takes $O(1)$ time using $O(n)$ work. Using Cole's parallel merge sort [1], step 5.2 can be done in $O(\log n)$ time using $O(n\log n)$ work. Using an optimal algorithm for computing the prefix sums of n values, step 5.3 can be performed in $O(\log n)$ time using $O(n)$ work. Using binary search, step 5.4 can be done in $O(\log n)$ time using $O(\log n)$ work. Therefore, the above implementation of step 5 takes $O(\log n)$ time using a total of $O(n\log n)$ work. We have the following theorem.

Theorem 1: The problem of finding in a tree network the minimum eccentricity tree-shaped facility of a specified length can be solved in $O(\log n)$ time using $O(n\log n)$ work.

The above implementation of step 5 uses a total of $O(n\log n)$ work, since sorting is performed. Later, using the prune-and-search technique, we will give a second implementation of step 5, which uses only $O(n)$ work. Our second implementation is motivated by the parallel selection algorithm proposed by Vishkin [11]. The following lemma is needed for the implementation.

Lemma 9 [11]: Let Q be a set of m distinct elements. In $O(\log m)$ time using $O(m)$ work, we can find an element z in Q with the property $|Q_1|\le 3m/4$ and $|Q_2|\le 3m/4$, where Q_1 and Q_2 is the sets of elements smaller than and larger than z respectively.

Our second implementation of step 5 is as follows.

Step 5:

Step 5.1: For each edge $e=(u, p(u))$ in E, if $p(u)$ is the root c or $e \neq M(p(u))$, set $delta\_iw_e=0$, $delta\_xr_e=R(e)$, and $delta\_xn_e=1$; otherwise set $delta\_iw_e=W(g)$, $delta\_xr_e=R(e)-R(g)$, and $delta\_xn_e=0$, where $g=(p(u), p(p(u)))$.

Step 5.2: Set $Q=E$, $cumulate\_iw\_xr=0$, and $cumulate\_xn = 0$.

Step 5.3: Repeat the following sub-steps to prune away edges from Q until $|Q|<n/\log n$.

5.3.1: Find an edge z in Q with the property $|Q_1|\le 3|Q|/4$ and $|Q_2|\le 3|Q|/4$, where Q_1 (Q_2 resp.) is the set of edges that are more important than z (less important than z resp.).

5.3.2: Partition Q into three sub-sets Q_1, $\{z\}$ and Q_2

5.3.3: Compute $IW_{Q_1}=\sum_{e\in Q_1} delta\_iw_e$, $XR_{Q_1}=\sum_{e\in Q_1} delta\_xr_e$, and $XN_{Q_1}=\sum_{e\in Q_1} delta\_xn_e$. And then compute $L_{rank(z)}=(cumulate\_iw\_xr + IW_{Q_1} + XR_{Q_1})-(cumulate\_xn+ XN_{Q_1})R(z)$.

5.3.4: If $L_{rank(z)}\le l$, then set $Q=Q_2$, $cumulate\_iw\_xr =cumulate\_iw\_xr+ IW_{Q_1} +delta\_iw_z+ XR_{Q_1} + delat\_xr_z$, and $cumulate\_xn=cumulate\_xn+ XN_{Q_1} +delta\_xn_z$. Otherwise, set $Q= Q_1\cup\{z\}$.

Step 5.4: Use Cole's parallel merge sort to determine the edge w in Q.

It is easy to see that steps 5.1, 5.2, and 5.4 take $O(\log n)$ time using a total of $O(n)$ work. The running time of step 5.3 is estimated as follows. Consider an arbitrary iteration of step 5.3. Let $m=|Q|$. By lemma 9, step 5.3.1 takes $O(\log m)$ time using $O(m)$ work. Using an optimal prefix sum algorithm, step 5.3.2 and 5.3.3, can be easily done in $O(\log m)$ using $O(m)$ work. Clearly, step 5.3.4 can be implemented in $O(\log m)$ time using $O(m)$ work. Therefore, each iteration of step 5.3 takes $O(\log |Q|)$ time using $O(|Q|)$ work. Since the size of Q is reduced by a factor of 3/4, $O(\text{loglog } n)$ iterations of step 5.3 are sufficient to reduce the size of Q below $n/\log n$. Thus, step 5.3 takes $O(\log n\times\text{loglog } n)$ time, using a total of $O(n+(3/4)n+(3/4)^2n+ ...)=O(n)$ work. We have the following theorem.

Theorem 2: The problem of finding in a tree network the minimum eccentricity tree-shaped facility of a specified length can be solved in $O(\log n\text{loglog } n)$ time using $O(n)$ work.

Clearly, if we perform step 5.3 of our second implementation of step 5 only $r<\text{loglog } n$ times, we can obtain the following theorem.

Theorem 3: The problem of finding in a tree network the minimum eccentricity tree-shaped facility of a specified length can be solved in $O(\log n+r\log n)$ time using $O((3/4)^r\times n\log n)$ work, where $0\le r\le\text{loglog } n$.

4 Minimum Distancesum Tree

Let m be a median of T. It was shown by Minieka [4]

that for all feasible l, there is a minimum distancesum tree-shaped facility of a length l that contains m. For easy discussion, in this section, we assume that m is a vertex in T. And, we assume that T is a rooted tree with a root m.

In this section, we redefine the *importance* of an edge $e=(u, p(u))$ in T, denoted by $R(e)$, to be $R(e)=size(u)$. The *rank* of an edge $e \in E$, denoted by $rank(e)$, is the number of edges in E that are more important than or as important as e. Denote a_k, $1 \le k \le n-1$, as the k-th most important one among the edges in E.

Our algorithm for finding the minimum distancesum tree is similar to that we have proposed for finding the minimum eccentricity tree. It is as follows.

ALGORITHM Min_Sum_Tree($T=(V, E)$, l)

Step 1: Compute $L_n=\sum_{e \in E} W(e)$. If $l>L_n$ return 'no solution'. Otherwise, if $l=L_n$ return T.

Step 2: Find a median m of T, and then orient T into a tree that is rooted at m.

Step 3: Determine $R(e)=size(u)$ for each edge $e=(u, p(u))$ in T.

Step 4: Find an edge w in E with the property that $L_{rank(w)-1} < l \le L_{rank(w)}$, where $L_k=\sum_{1 \le i \le k} W(a_i)$ for $k=0, 1, ..., n-1$.

Step 5: Obtain the minimum distancesum tree of a length l from T by removing all edges that are not more important than w and cutting down w by a length of $L_{rank(w)}-l$.

It is not difficult to see the correctness of algorithm Min_Sum_Tree. With an analysis similar to that of algorithm Min_Ecc_Tree, we can obtain the following theorem.

Theorem 4: The problem of finding in a tree network the minimum distancesum tree-shaped facility of a specified length can be solved in $O(\log n + r \log n)$ time using $O((3/4)^r \times n \log n)$ work, where $0 \le r \le \log\log n$.

5. Concluding Remarks

In [4], eight different problems were considered by Minieka. In this paper, parallel algorithms were proposed for only two of the problems. It was showed that the problem of finding the maximum distancesum subtree of a specified length in a weighted tree network is NP-hard [8]. In the following, we discuss the parallel complexities of the other five problems on the EREW PRAM.

For the problems of finding the maximum eccentricity path and the maximum eccentricity subtree of a specified length, Peng and Lo [6] proposed an $O(\log n)$ time algorithm using $O(n)$ work for unweighted tree networks. It is not difficult to see that Peng and Lo's algorithm can be easily extended to weighted tree network without losing any efficiency. For the problems of finding the minimum eccentricity path of a specified length, an $O(\log n)$ time algorithm using $O(n)$ work can be found in [12], which improves that proposed in [6] from the aspect of work. For the problems of finding a minimum distancesum path and a maximum distance-sum path of a specified length in a tree networks, $O(\log n)$ time algorithms using $O(n^2)$ work can be found in [12].

References

[1] R. Cole, "Parallel merge sort," *SIAM Journal on Computing*, vol. 17, no. 4, pp. 770-785, 1988.

[2] R. Cole and U. Vishkin, "Approximate parallel scheduling, Part I: The basic technique with applications to optimal parallel list ranking in logarithmic time," *SIAM Journal on Computing*, vol. 17, no. 1, pp. 128-142, 1988.

[3] J. Jaja, *An Introduction to Parallel Algorithms*, Addison Wesley, 1992.

[4] E. Minieka, "The optimal location of a path or tree in a tree network," *Networks*, vol. 15, pp. 309-321, 1985.

[5] E. Minieka and N. H. Patel, "On finding the core of a tree with a specified length," *Journal of algorithms*, vol. 4, pp. 345-352, 1983.

[6] S. Peng and W.-T. Lo, "The optimal location of a structured facility in a tree network," *Parallel Algorithms and Applications*, vol. 2, pp. 43-60, 1994.

[7] S. Peng and W.-T. Lo, "Efficient algorithms for finding a core of a tree with a specified length," *Journal of Algorithms*, vol. 20, pp. 445-458, 1996.

[8] R. Rabinovitch and A. Tamir, "On a tree-shaped facility location problem of Minieka," *Networks*, vol. 22, pp. 515-522, 1992.

[9] P. J. Slater, "Locating central paths in a network," *Transportation Science*, vol. 16, no. 1, pp. 1-18, 1982.

[10] B. C. Tansel, R. L. Francis, and T. J. Lowe, "Location on networks: A survey," *Management Science*, vol. 29, pp. 482-511, 1983.

[11] U. Vishkin, "An optimal parallel algorithm for selection," *Advances in Computing Research*, JAI Press Inc., Greenwich, CT, 1987.

[12] B.-F. Wang, "Efficient parallel algorithms for optimally locating a path and a tree of a specified length in a weighted tree network," manuscript.

	Time	Work
Min-Ecc Path	$O(\log n)$	$O(n \log n)$
Min-Sum Path	$O(\log^2 n)$	$O(n \log n)$
Max-Ecc Path	$O(\log n)$	$O(n)$
Max-Sum Path	$O(\log^2 n)$	$O(n \log n)$
Min-Ecc Tree	$O(\log n)$	$O(n \log n)$
Min-Sum Tree	$O(\log n)$	$O(n \log n)$
Max-Ecc Tree	$O(\log n)$	$O(n)$
Max-Sum Tree	$O(\log^3 n)$	$O(n^3 \log^3 n)$

TABLE 2. Complexities of Peng and Lo's algorithms.