Weighted Central Paths in a Tree

by

José A. Pino and Christian E. Moris
Departamento de Ciencias de la Computación
Universidad de Chile
Casilla 2777
Santiago, Chile

Abstract. A generalized procedure is given to find the central path in a tree. In particular, it is applied to find the weighted *path center* and the weighted *spine* of a tree. The approach taken is to build the path from an already found central vertex and computed *limb* weights. If these are provided, the presented algorithms are linear-time.

1. Introduction

There are facility location problems which can be modelled as the search for a central path in a graph, such as the search for pieces of roads, pipelines or railroad lines which optimize a given cost criterion. In a broad sense, typically these criteria are of the minimax type (minimize the largest distance from any vertex of the graph to the path) or the maxisum type (minimize the sum of the distances from any graph component which does not include the path, to the path itself)[7].

In the particular case of a tree graph, Slater [6] has presented linear-time algorithms to find the central paths for two definitions of it: the *path center* and the *spine* (path centroid). Hedetniemi *et al.* [1] also provide a linear-time algorithm for the path center of a tree. Morgan and Slater [4] develop a linear-time algorithm to find yet another kind of central path of a tree: the *core* (path median).

The cited works, however, do not consider the case of a tree graph with vertex weights. This situation may arise in practice. For instance, trying to replace with a subway line the most used segment of a road network, such as the one depicted in Fig. 1, should take into account the traffic generated at the vertices as well as the distances involved.

A recently published paper by Rosenthal and Pino [5] gives a generalized algorithm to find a CENTRAL VERTEX in a tree under a variety of cost criteria. This algorithm is used here to construct a procedure to find a weighted central path in a tree graph. In particular, it can be applied to find the weighted spine or the weighted path center.

This work has been partially supported by grant No. 89-1106 from Fondo Nacional de Ciencia y Tecnología (Chile)

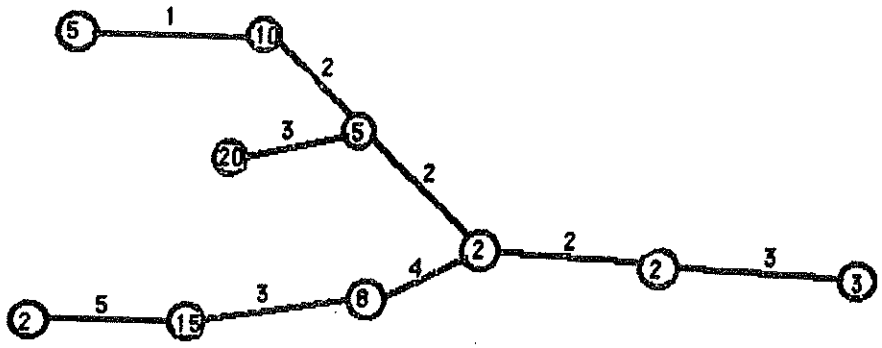

Figure 1. Values inside the nodes represent traffic units generated at the vertices. Figures on arcs represent distances.

2. Definitions and Terminology

Let $V = \{ v_i / i \in I = \{1,...,n\} \}$ be a vertex set, and $E = \{ (u,v) / u,v \in V \}$ an edge set.

A **path** $P(v_{i_1},v_{i_k})$ is a sequence $v_{i_1}, \ldots, v_{i_k}$ of $k \geq 1$ distinct elements of V such that there exists an edge $(v_p,v_q) \in E$ for any pair v_p,v_q of consecutive vertices in the sequence.

A rooted tree $T(V,E,r_o)$ is a triple such that

i) $r_o \in V$ (called the root),

ii) for any $(u,v) \in E$, also $(v,u) \in E$ (directed tree, but all edges have their symmetric pair),

iii) there exists a unique path P(u,v) for any $u,v \in V$.

We also define:

$Adj(v) \equiv \{ u / u \in V \text{ and } (u,v) \in E \}$

$degree(v) \equiv | Adj(v) |$

If $P = P(v_1,v_2)$ is a path of a tree and $u \in Adj(v_1)$, we denote by $u \,//\, P$ the path $u,v_1,\cdots,v_2$. A similar concatenation to the right defines $P \,//\, u$. |P| is the number of vertices of P.

$l_{(u,v)}$ is the length of the edge (u,v) (a positive real number for $u \neq v$). The edges satisfy $l_{(u,v)} = l_{(v,u)}$.

$d(u,v) \equiv \sum_{P(x,y) \subseteq P(u,v),\ (x,y) \in E} l_{(x,y)}$ is the distance from u to v.

If $S \subseteq V$, then the distance from $u \in V$ to S is defined $d(u,S) = \min \{ d(u,v), v \in S \}$.

$\{ c_j : V \rightarrow R^+ / j \in J = \{1,...,m\} \}$, is a set of vertex cost functions (vertex weights).

As in [5], for a tree $T(V,E,r_o)$, we define the limb $L_{(u,v)}(V_{(u,v)},E_{(u,v)})$ as a pair where $(u,v) \in E$ and $V_{(u,v)} = \{ w \in V / u \notin P(v,w) \}$ and $E_{(u,v)} = \{ (x,y) / (x,y) \in E;\ x,y \in V_{(u,v)} \} \cup \{ (u,v), (v,u) \}$ (see Figure 2).

In what follows, for simplicity, $T(V,E,r_o)$ and $L_{(u,v)}(V_{(u,v)},E_{(u,v)},(u,v))$ will be denoted T and $L_{(u,v)}$, respectively.

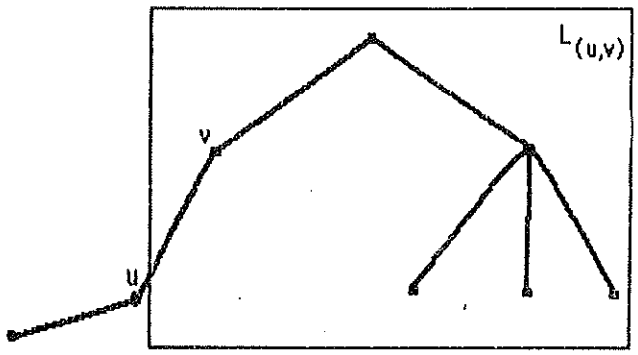

Figure 2
$L_{(u,v)}$: the limb pointed to by (u,v)

The problem of finding a central vertex in a tree is completely specified if there is a cost definition of a tree when the central vertex is located at each vertex $u \in V$. This cost is the real-valued quantity e(u), called the *eccentricity* of the tree at u. The goal, then, is to find a vertex $v \in V$ such that e(v) is minimum.

The vertex eccentricity for the *1-center* problem is:

$e(u) = \max_{x \in V} \{ d(u,x) \}, \quad u \in V.$

For the *weighted 1-center* problem, the vertex eccentricity is defined by

$e(u) = \max_{x \in V} \{ c(x)\, d(u,x) \}, \quad u \in V.$ (1)

The *weighted centroid* is the vertex which minimizes the maximum traffic carried by any single edge. If u is the centroid, the maximum traffic $t_{(u,x)}$ will be on the edge (u,x) out of u:

$e(u) = \max_{x \in V} \{ t_{(u,x)} \}, \quad u \in V.$ (2)

Other vertex centrality criteria are described in [5].

The definition of eccentricity can be extended also to subsets of vertices of T. In particular, if P(u,v) is a path, and the distance to a path is taken to the set of vertices of the path, then

$e(P(u,v)) = \max_{x \in V} \{ c(x)\, d(x, P(u,v)) \}$, for the weighted 1-center problem, (3)

$e(P(u,v)) = \max_{x \in P(u,v)} \{ \max t_{(x,y)} / y \in Adj(x), y \notin P(u,v) \}$, for the weighted centroid problem.

The CENTRAL VERTEX algorithm [5] computes quantities called *limb weights* wt(u,v), associated to each limb $L_{(u,v)}$ of the tree. The limb weight wt(u,v) is a summary of the information about the limb. The summary is sufficient in the sense that:

i) wt(u,v) contains all the information about $L_{(u,v)}$ relevant to computing e(u), and

ii) If $L_{(w,y)} \supseteq L_{(u,v)}$, wt(u,v) contains all the information about $L_{(u,v)}$ relevant to computing wt(w,y).

Intuitively, the limb weight expresses the cost contribution of the limb to the eccentricity of any vertex outside it. The next two expressions define useful sets of weights.

$WTS(v) \equiv \{ wt(v,x) \mid \text{such that } x \in Adj(v) \}$

$OTHER\_WTS(u,v) \equiv WTS(u) - \{wt(u,v)\}$

If P(u,v) is a path in a tree T with $e(P(u,v)) \leq e(P(x,y))$ for any path P(x,y) in T, and, if there does not exist a path P(x,y) in T with $|P(x,y)| < |P(u,v)|$ for which $e(P(x,y)) = e(P(u,v))$, then P(u,v) will be called weighted path center of T (eccentricity defined as in (1)) or weighted spine (eccentricity defined as in (2)).

3. Weighted Central Paths

The CENTRAL VERTEX algorithm [5] finds the weighted centroid of a tree in O(|V|) time and also computes the limb weights wt(u,v) for all $(u,v) \in E$. Although CENTRAL VERTEX can also be used to find the unweighted 1-center of a tree in linear time, the problem with weights violates the required conditions, since in this case it is impossible to compute a limb weight from the weights of the neighboring limbs, adjacent edge length and local vertex weight. Nevertheless, the same algorithm can be modified to find the weighted 1-center of a tree and corresponding limb weigths in $O(|V|^2)$ time. The rest of this paper assumes that this algorithm is used to compute either the weighted 1-center and the limb weights of the tree, or the weighted centroid and associated limb weights.

It should be noted that Megiddo [3] gives a linear-time algorithm to compute the weighted 1-center of a tree, but does not provide all limb eccentricities. The same remark applies to a previous algorithm by Kariv and Hakimi [2] which has O(|V| log|V|) time complexity. Although the algorithm by Hedetniemi et al. [1] finds a weighted 1-center of a tree in linear time, it refers to a simpler, additive weights case.

LEMMA 1. For any path $P(u,v) = u, w_1, \ldots, w_k, v$, the following inequality holds:

$$wt(u,w_1) > wt(w_1,w_2) > \ldots > wt(w_k,v).$$

Proof. For the centroid problem, from [5]:

$$wt(u,w_1) = \text{total}\,(OTHER\_WTS\,(w_1,u)) + c(w_1)$$

where **total** is a function which yields the sum of the argument values. Since every value of $OTHER\_WTS(w_1, u)$ is a positive real number, one of them is $wt(w_1,w_2)$. Thus,

$$\text{total}(OTHER\_WTS(w_1, u)) > wt(w_1,w_2)$$

and since $c(w_1) > 0$,

$$wt(u,w_1) > wt\,(w_1,w_2)$$

An inductive argument proves the thesis. For the weighted 1-center problem,

$$wt(u,w_1) = \max\,\{\, c(x)\, d(u,x) \;/\; x \in V_{(u,w_1)} \,\} > \max\,\{\, c(z)\, d(w_1,z) \;/\; z \in V_{(w_1,w_2)} \,\} = wt(w_1,w_2)$$

since $L_{(u,w_1)} \supset L_{(w_1,w_2)}$ and $d(u, w_1) > 0$.

□

COROLLARY 1. Let v be a central vertex in T such that

$e(v) = \max \{wt(v,w) / w \in Adj(v)\} \leq \max_{x \in V} \{wt(x,y) / y \in Adj(x)\}$

Then, for each path $v, w_1, \cdots, w_k, z$ with $v \neq z$, the eccentricity of z is $e(z) = wt(z, w_k)$.

□

COROLLARY 2. For any pair u,v of adjacent vertices, the next inequality holds:

$wt(u,v) > \max \{ wt(x,w) / \forall x \in V_{(u,v)}, w \notin P(u,x) \}$

□

Hedetniemi et al. [1] proved the uniqueness of the unweighted path center of a tree. With few changes, it can be used to prove:

THEOREM 1. The weighted path center and the weighted spine of a tree are unique.

□

THEOREM 2. The weighted path center P*, or the weighted spine P*, of a tree T contains the weighted 1-center vertex v, or the weighted centroid v, respectively.

Proof. P* satisfies, by definition, the following for any path P in T,

$$\max_{u \in P^*} \{ wt(u,x) / x \in Adj(u), x \notin P^* \} \leq \max_{u \in P} \{ wt(u,x) / x \in Adj(u), x \notin P \} \quad (4)$$

and v is such that,

$$e(v) = \max_{z \in Adj(v)} \{ wt(v,z) \} \leq e(y) = \max_{z \in Adj(y)} \{ wt(y,z) \} \quad \text{for all } y \in V. \quad (5)$$

Assume $v \notin P^*$ and let Q be a path built only with v, i.e., $Q = Q(v,v)$. Then, for Q:

$$e(v) = \max \{ wt(v,z) \} = \max \{ wt(v,z) / z \notin Q \} \quad (6)$$

Let $w \in P^*$ such that $d(v,w) = d(v,P^*)$, and $R=R(v,w)$ be the path $v, w_1, \ldots, w_k, w$. By corollary 1 to lemma 1,

$$e(w) = wt(w,w_k) = \max_{z \in Adj(w)} \{wt(w,z)\} \leq \max_{u \in P^*} \{wt(u,z) / z \notin P^* \} \quad (7)$$

Using (6) and (7) in (5), we have

$$\max_{u \in Q} \{wt(u,z) / z \notin Q\} \leq \max_{u \in P^*} \{wt(u,x) / x \in Adj(u), x \notin P^*\}$$

but this violates the uniqueness of the central path (Theorem 1), therefore $v \in P^*$.

□

Theorem 2 suggests a method to find the central path of a tree starting with the central vertex (The approach is similar to the one described by Slater [6].) The next algorithm builds a path P^+, which after some pruning, will be proved to be the central path.

ALGORITHM CPC (Central Path Construction)

/* It builds a path $P^+ = u_r, \cdots, u_1, v, v_1, \cdots, v_s$ */

1. Let v be a central vertex and v_1 be a vertex such that

   $e(v) = wt(v,v_1) = \max \{wt(v,w) / w \in Adj(v)\}$ /* from the CENTRAL VERTEX algorithm */

   make $P^+ \leftarrow v // v_1$

2.1 $i \leftarrow 2$

2.2 Find vertex $v_i \in Adj(v_{i-1})$ and $v_i \notin P^+$, such that

$wt(v_{i-1}, v_i) = \max \{ wt(v_{i-1}, w) / w \notin P^+ \}$

If more than one vertex satisfies this condition, then $s \leftarrow i - 1$ and go to step 3.

Else, $P^+ \leftarrow P^+ // v_i$.

2.3 If there are unvisited vertices which are adjacent to v_i, $i \leftarrow i + 1$ and go to step 2.2

Else, $s \leftarrow i$.

3. Find vertex $u_1 \in$ Adj(v), $u_1 \notin P^+$, such that

$wt(v, u_1) = \max \{ wt(v,w) / w \notin P^+ \}$

and $P^+ \leftarrow u_1 // P^+$

4.1 $i \leftarrow 2$

4.2 Find vertex vertex $u_i \in$ Adj(u_{i-1}), $u_i \notin P^+$, such that

$wt(u_{i-1}, u_i) = \max \{ wt(u_{i-1}, w) / w \notin P^+ \}$.

If more than one vertex satisfy this condition, $r \leftarrow i - 1$ and stop.

Else, $P^+ \leftarrow u_i // P^+$.

4.3 If there are unvisited vertices which are adjacent to u_i, $i \leftarrow i + 1$ and go to step 4.2.

Else, $r \leftarrow i$ and stop.

□

LEMMA 2. Algorithm CPC takes O(|V|) time to build path P^+.

Proof. Step 1 takes time O(degree(v)) = O(|V|). Loop 2 (steps 2.2 and 2.3) assembles a path from v to at most a leaf of T, visiting each vertex once; thus the loop is bounded by

$$O(\sum_{i=1}^{|V|} degree(v_i)) = 2|V| - 1 = O(|V|).$$

Step 3 is analogous to step 1, and loop 4 is similar to loop 2. Therefore, algorithm CPC takes O(|V|) time.

□

LEMMA 3. Any path P^+ built by algorithm CPC satisfies $e(P^+) \leq e(Q)$, for any path Q in T.

Proof. Suppose the contrary, i.e., there is a path Q such that

$$\max_{u \in P} \{wt(u,w) / w \notin P^+\} > \max_{u \in Q} \{wt(u,w) / w \notin Q\} \quad (8)$$

By step 1 of CPC, the central vertex $v \in P^+$.

1) We first prove $v \in Q$. Suppose the contrary, i.e., $v \notin Q$ and let $w \in Q$ be the closest vertex to v (Figure 3a). Let $w, w_1, \ldots, w_k, v$ be the path P(w,v).

By definition,

$e(Q) \geq wt(w, w_1)$.

By corollary 2 to lemma 1,

$wt(w, w_1) > wt(w_k, v)$.

By corollary 1 to the same lemma,

$wt(w_k, v) > e(v)$.

By algorithm CPC,

$e(v) > \max_{x \in P^+}\{wt(x,y) / y \notin P^+\}$.

Combining the previous relations,

$e(Q) \geq wt(w, w_1) > wt(w_k, v) > e(v) > \max_{x \in P^+}\{wt(x,y) / y \notin P^+\} = e(P^+)$.

But this contradicts (8), thus, $v \in Q$.

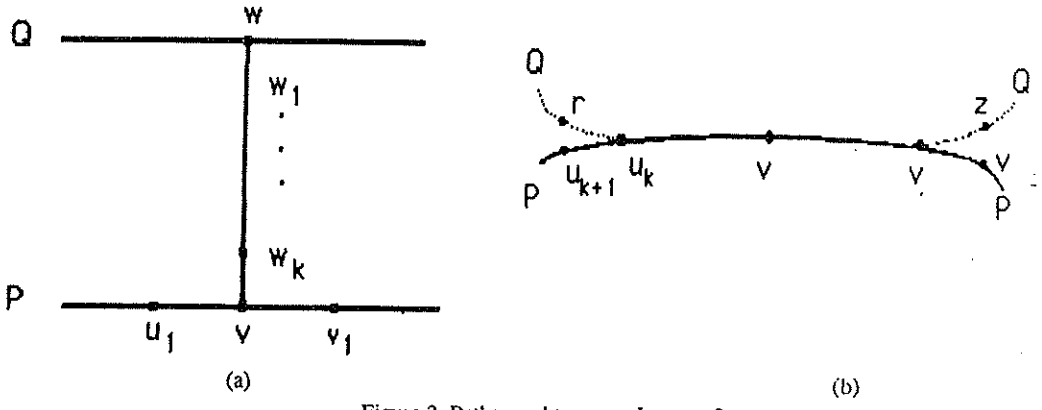

Figure 3. Paths used to prove Lemma 3

2) We now prove that (8) cannot hold. Suppose P^+ and Q have vertices $u_k, \ldots, u_1, v, v_1, \ldots, v_l$ in common (Figure 3b).

If $\max_{u \in P^+}\{wt(u,w) / w \notin P^+\}$ occurs in $P(u_k, v_l)$, then $e(P^+) \leq e(Q)$, contradicting (8). Then let

$e(P^+) = wt(x,y)$, for $x \in P^+$, $y \notin P^+$, $(x,y) \in E_{v_l, v_{l+1}}$ (the same argument applies to $(x,y) \in E_{u_k, u_{k+1}}$)

Then, by corollary 2 to lemma 1 and by algorithm CPC,

$wt(v_l, v_{l+1}) > wt(x,y) = e(P^+)$

By definition (see Figure 3b),

$e(Q) \geq wt(v_l, v_{l+1})$

Combining the last two relations, $e(Q) > e(P^+)$, contradicting (8).

□

It might be noted that P^+ spans from one leaf to another leaf of the tree, unless in the process of going to a leaf, two edges have identical limb weight (steps 2.2 and 4.2 of CPC). The rationale to stop path building in the latter case is that adding a new vertex (and edge) to the path will not decrease the maximum path weight, already bounded by the limb weight on the other edge.

It will be proved that a path P^*, which is path P^+ after some pruning, is the weighted central path as defined previously. To do the pruning, we need to compute $e(P^+)$. Expression (3) above

gives the precise definition of path eccentricity for the weighted path center problem. For the weighted spine problem, we first notice that:

wt(u,v) = total(OTHER_WTS(v,u)) + c(v)

and

e(v) = max(WTS(v)).

Then,

$$e(P(u,v)) = \max_{x \in P(u,v)} \{\max \{ wt(x,y) / y \notin P(u,v)\}\} \quad (11)$$

It is therefore easy to verify the next lemma for both versions of path centrality.

LEMMA 4. Computation of $e(P^+)$ can be done on O(|V|) time if all limb weights are already computed.

□

The pruning operation can be explained as follows. Let P^+ be the path $u_r,...,u_1,v,v_1,\ldots,v_s$. If $wt(u_{r-1},u_r) < e(P^+)$, then it is clear by Lemma 1 that extracting u_r from P^+ still leaves P^+ with the same eccentricity. This idea can be applied iteratively, as in the next procedure:

ALGORITHM PP (Path Pruning)

/* The output is path $P^* = u_p,...,u_1,v,v_1,...,v_q$, starting with path $P^+ = u_r,\ldots,u_1,v,v_1,...,v_s$ */

1. i ← r .
2. If $wt(u_{i-1},u_i) < e(P^+)$ then i ← i+1 and repeat step 2.
   Else, if $wt(u_{i-1},u_i)$ then p ← i-1.
   Else, if $wt(u_{i-1},u_i)$ then p ← i.
3. i ← s
4. If $wt(v_{i-1},v_i) < e(P^+)$ then i ← i+1 and repeat step 4.
   Else, if $wt(v_{i-1},v_i)$ then q ← i-1.
   Else, if $wt(v_{i-1},v_i)$ then q ← i.
5. Stop

□

THEOREM 3. Path P^* produced by algorithm PP is the weighted path center or the weighted spine, depending on the eccentricity and limb weight definition.

Proof. P^* has minimum eccentricity, since the pruning operation in algorithm PP preserves this eccentricity. By Theorem 1, there cannot be two (or more) paths with minimum eccentricity having the same number of vertices.

There remains to be proved that there is no path with minimum eccentricity having less number of vertices than P^*.

Let then Q be a path with minimum eccentricity such that $|Q| < |P^*|$. If v is the central vertex, then $v \in P^*$ by construction, and $v \in Q$ by Theorem 1. Thus, either $Q \subseteq P^*$ or $Q \cap P^* \neq \phi$ with Q having at least one vertex not in P^*.

The first case, $Q \subseteq P^*$, is impossible by steps 2 and 4 of algorithm PP (and steps 2.2 and 4.2 of algorithm CPC). The second case, $Q \cap P^* \neq \phi$ with at least one distinct vertex, is also impossible: let $v_l \in Q \cap P^*$, with $v_{l+1} \in P^*$, $v_{l+1} \notin Q$, $z \in Q$ and $z \in \text{Adj}(v_l)$. Since v_{l+1} is the vertex chosen by algorithm CPC to expand P^* (step 2.2),

$$wt(v_l, v_{l+1}) > wt(v_l, z)$$

By step 2 of algorithm PP, v_{l+1} cannot be pruned without increasing the eccentricity of P^*, therefore, $e(Q) > e(P^*)$ and this contradicts the assumption of Q having minimum eccentricity.

□

THEOREM 4. The time complexity of finding the weighted central path of a tree is bounded by the computation of limb weights.

Proof. P^* is computed by:

i) Perform algorithm CENTRAL VERTEX on the tree,

ii) Perform algorithm CPC to yield P^+,

iii) Compute $e(P^+)$ and

iv) Perform algorithm PP to yield P^*.

Step i) takes O(|V|) time for the weighted centroid and $O(|V|^2)$ for the weighted 1-center (and corresponding limb weights). Step ii) takes O(|V|) time (lemma 2). Step iii) also takes O(|V|) time (lemma 4). Finally, and since PP at most visits all the vertices on path P^+, step iv) takes O(|V|) time.

□

COROLLARY 1. The weighted spine of a tree can be found in O(|V|) time.

□

COROLLARY 2. The weighted path center of a tree can be found in $O(|V|^2)$ time.

□

4. Concluding remarks

The procedures developed in the previous section do not require the use of the CENTRAL VERTEX algorithm; any algorithm which computes the central vertex and limb weights of the tree can replace it. The approach can be applied to any definition of central path built from a central vertex which is part of it (see [5] for some definitions of central vertices). As noted in [6], the path median (core) of a tree does not necessarily contain the median vertex, and is thus an example of a case where the approach presented in this paper cannot be taken.

References

1. Hedetniemi, S.M., Cockayne, E.J., Hedetniemi, S.T., Linear Algorithms for Finding the Jordan Center and Path Center of a Tree. **Trans. Sci. 15**, 2 (May 1981), 98-114.

2. Kariv, O. and Hakimi, S.L., An Algorithmic Approach to Network Location Problems, I: the p-centers. **SIAM J. on Applied Math. 37**, 3 (1979), 513-538.

3. Megiddo, N., Linear-Time Algorithms for Linear Programming in R^3 and related problems. **SIAM J. Comput. 12**, 4 (Nov. 1983), 759-776.

4. Morgan, C.A. and Slater, P.J., A Linear Algorithm for a Core of a Tree, J. of Algorithms 1 (1980), 247-258.

5. Rosenthal, A. and Pino, J.A., A Generalized Algorithm for Centrality problems on Trees. JACM 36, 2 (Apr. 1989), 349-361.

6. Slater, P.J., Locating Central Paths in a Graph, Trans. Sci. 16, 1 (Feb. 1982), 1-18.

7. Tansel, B.C., Francis, R.L. and Lowe, T.J., Location on Networks: A survey. Part I: the p-center and p-median problems. Management Sci. 29, 4 (Apr. 1983), 482-511.