December, 1982

FINDING THE TWO-CORE OF A TREE

Ronald I. Becker*[1,2]
Yehoshua Perl**

DCS-TR-121

*Department of Mathematics
University of Cape Town
Rondebosch 7700
Republic of South Africa

and

Department of Mathematics
University of Tennessee
Knoxville, TN 37996-1300

**Department of Computer Science
Rutgers University
New Brunswick, NJ 08903

and

Bar-Ilan University
Ramat Gran
Israel

---

[1] Supported in part by a grant from CSIR, Pretoria, Republic of South Africa.

[2] Work done in part while visiting Bar-Ilan University, Ramat Gran, Israel.

ABSTRACT

The 1-core of a graph is a path minimizing the sum of the distances of all vertices of the graph from the path. A linear algorithm for finding a 1-core of a tree was presented by Morgan and Slater. The problem for general graphs is NP-Hard.

A 2-core of a graph is a set of two paths minimizing the sum of the distances of all vertices of the graph from any of the two paths. We consider both cases of disjoint paths and intersecting paths for a tree.

Interesting relations between 1-core and 2-core of a tree are found. These relations imply efficient algorithms for finding the 2-core. The complexity of the algorithms is $O(|V|.d(T))$ where $d(T)$ is the number of edges in the diameter of the tree. These algorithms are applicable for routing highways in a system of roads.

A w-point core is a path minimizing the sum of the distances of all vertices of the graph from either the vertex w or the path. A linear algorithm for finding a w-point core of a tree is presented. It is applied as a procedure for the previous algorithms.

1. Introduction

The core of a graph is defined in [MS] as a path in the graph minimizing the sum of the distances of all vertices of the graph from the path. The notion of a core is generalization of the notions of median and multimedian, where the median is a vertex minimizing the sum of distances from a single vertex to all other vertices. Algorithms for finding the median in a network and the multimedian in a tree appear in [HA] and [KH] respectively. One of the applications mentioned in [MS] is routing a highway through a road network. Morgan and Slater [MS] give a linear algorithm for finding the core of a tree. This is an interesting algorithm traversing the edges of the tree in a double scanning first bottom-up and then top-down.

In this work we consider the case of routing two highways through a road network. We define a 2-core of a graph as a set of 2 paths minimizing the sum of the distances of all vertices of the graph from the two paths. We consider both cases of disjoint paths, the 2-core, and intersecting paths the I2-core. We refer to the core as 1-core.

Note that the 1-core problem is NP-Hard [GJ] by the trivial reduction from the Hamiltonian path problem. The 2-core problem is also NP-Hard and the reduction is from the problem of Partition onto k Hamiltonian subgraphs where $k = 2$ (Papadimitriou, see problem [GT 13] in [GJ]).

We consider the problem of finding a 2-core and I2-core of a tree. The relation between a 1-core and a 2-core and I2-core are investigated and applied to obtain efficient algorithms for these problems.

Formal definitions are presented at the end of the introduction. Section 2 gives a version of the Morgan-Slater [MS] algorithm for finding the 1-core which will be used in the sequel for theoretical and algorthmic purposes. An example demonstrating the algorithm is presented. A modification of the algorithm for the weighted 1-core problem

is presented. Section 3 presents a simple algorithm for finding a 2-core of a tree. Section 4 finds an interesting relation between a 1-core and a 2-core of a tree implying a more efficient algorithm for the 2-core problem. Similar relation and algorithm for the I2-core are given in Section 5.

Let $T = (V,E)$ be a tree. We choose an arbitrary vertex $r \in V$ and regard T as being noted at r. We will use the usual terminology of graph theory (see Harary [H]).

Let $d(v_1,v_2)$ denote the number of edges in the path joining vertices v_1,v_2 in T. If P is a path in T, let

$$d(v,P) = \min \{d(v,v') \mid v' \in P\} .$$

Let B be a subgraph of T. We write

$$d_B(P) = \sum_{v \in B} d(v,P) .$$

We write $d_B(\{v'\})$ as $d_B(v')$ ($\{v'\}$ is the path with vertex v' and no edges). If $B = V$ then we simply write $d(P)$, $d(v)$ in place of $d_V(P), d_V(v)$. If P,Q are paths, let

$$d(v,P,Q) = \min \{d(v,P), d(v,Q)\}$$

and

$$d(P,Q) = \sum_{v \in V} d(v,P,Q) .$$

A *1-core* of T is a path P minimizing $d(P)$. A 1-core clearly contains two leaves of T. An *I2-core (2-core)* is a pair of (disjoint) paths $\{P,Q\}$ minimizing $d(P,Q)$. (I2-core is an abbreviation of intersecting 2-core).

The removal of a vertex v from T and all its adjacent edges splits T into one or more subtrees. The union of one of these subtrees, the edge joining it to v, and v itself is called a *branch at* v.

As is noted in [MS] the algorithm for the 1-core can be easily modified for the case where the edges have lengths. The same applies to our algorithms with the same complexity.

Another generalization of the problem is where each vertex v has a weight $w(v)$ and we consider the weighted sum of the distances

$$d(P) = \sum_{v \in T} w(v)\, d(v,P) \ .$$

In Section 2 we show that it is easy to modify the algorithm for the weighted 1-core problem. Similar modifications are possible for the other algorithms.

2. A 1-core Algorithm

We will now give a version of the algorithm of Morgan and Slater [MS] for finding a 1-core of a tree in linear time. This version is suitable for the algorithmic and theoretical applications we have for it.

The algorithm attaches two labels to each edge of T, one adjacent to each endpoint. The *upper label* of an edge is the label closer to the root, and the *lower label* is the other label. We use L_i^v to denote that i^{th} $(i = 1,2, \ldots)$ label adjacent to vertex v. We will have $L_i^v = (\ell_i^v, m_i^v, n_i^v)$ where the components of L_i^v are integers. The following procedure will be used to calculate the labels inductively.

Procedure A (To calculate the label adjacent to vertex v on edge e).

Let v' be the other endpoint of e. Let $e_1, \ldots, e_p$ be the other edges adjacent to v', and let $L_i^{v'}$ be the label on e_i adjacent to v' (see Figure 1(a) or (b).)

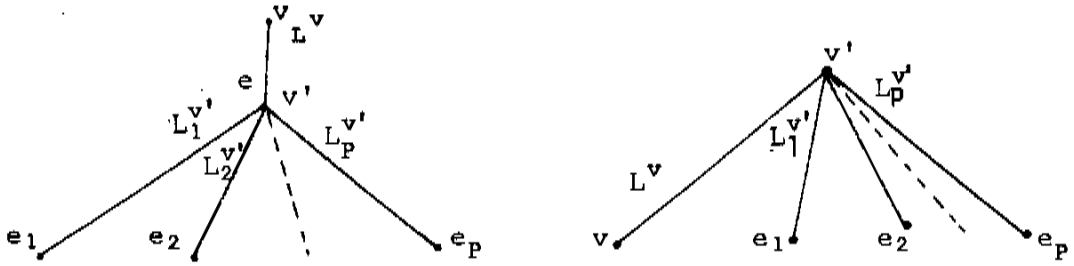


Figure 1

Then

$$n^v = \sum_{i=1}^{p} n_i^{v'} + 1$$

$$\ell^v = \sum_{i=1}^{p} \ell_i^{v'} + n^v$$

$$m^v = \max \{m_1^{v'}, \ldots, m_p^{v'}\} + n^v \quad .$$

Remark: We will show in Theorem 1 that ℓ, m, n have natural descriptions in terms of graph theory. These are :

n^v = number of vertices in the branch B at v containing e but excluding v .

$\ell^v = d_B(v)$

m^v = maximum one can save on $d_B(v)$ by putting a path P through v into B and replacing $d_B(v)$ by $d_B(P)$.

Algorithm 1

1. Let the upper labels of each leaf be $(1,1,1)$. Proceeding in bottom-up order calculate the upper labels of each edge of T using Procedure A.

2. Proceeding top-down level-by-level from the root, calculate the lower labels of each edge in T using Procedure A.
(Note that in calculating a lower label , upper labels and only one lower level are used.)

3. Traverse the vertices of T and assign to each vertex v the two quantities

$$d(v) = \sum_{\substack{L^v \text{ adjacent} \\ \text{to } v}} \ell^v$$

$c(v) = d(v) -$ sum of two largest elements of $\{m^v | L^v$ adjacent to $v\}$

(We will show that $d(v)$ coincides with our previous definition of d).

4. While traversing the tree calculate $c' = \min_{v \in V} c(v)$.
Find a maximal path P such that if $v \in P$ then $c(v) = c'$ and if e is on a path adjacent to v , its label m^v is one of the two largest adjacent to v .
P is then a 1-core

Remark: Clearly the above algorithm could be made more efficient, but the improvements we know all have the same asymptotic complexity, which is clearly linear. We now give an example and then prove correctness of algorithm 1.

Example: Operation of Algorithm 1

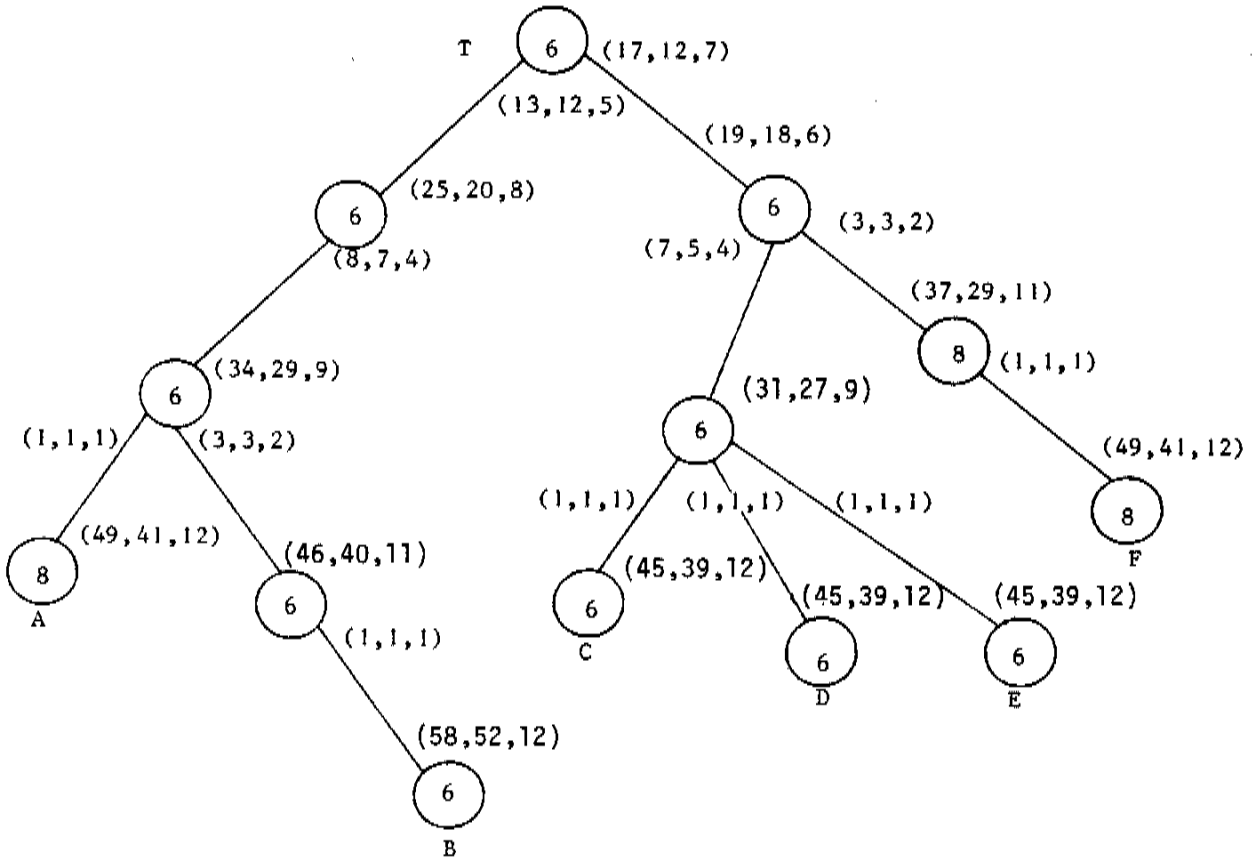


Figure 2

Figures inside vertex circles v represents $c(v)$.
Any maximal path, each vertex v having $c(v) = 6$, is a 1-core of T. A minimal cost path through vertex A has other terminal vertex C, D or E.
A minimal cost path through F has other terminal vertex B.

Note that the only change required for obtaining the weighted 1-core is labeling each leaf v by $(w(v),w(v),w(v))$

and changing the rule for $n(v)$ to $n(v) = \sum_{i=1}^{P} n_i^{v'} + w(v)$.

Theorem 1

(a) At the end of the labelling (step 2) in Algorithm 1 each edge has two labels.

(b) Let e be an edge adjacent to v and v' . Let B be the branch at v containing e and let $L^v = (\ell^v, m^v, n^v)$ be the label on e adjacent to v .
Then

$$\ell^v = d_B(v)$$

$$m^v = d_B(v) - \min_{P \text{ a path through } v \text{ in } B} d_B(P)$$

n^v = number of vertices in $B - \{v\}$.

(c) $d(v)$ (of step 3) is the same as the previous definition of $d(v)$. $c(v)$ = cost $(d(P))$ of those paths through v for which $d(P)$ is minimal.

Proof:

(a) To calculate upper labels at v of an edge (v',v) by Procedure A, we need the upper labels of all son-edges of v' . Initially, these are defined on leaves by step 1, and the level-by-level bottom-up calculation ensures that the calculation may be continued. To calculate the lower labels of son-edges of the root, we need all upper labels of these son-edges, which are available from step 1. Thereafter the same argument as before shows the labelling is successful.

(b) Initially, statement is true if e is incident with a leaf. It is easily seen that if $e = (v',v)$ and the description is true for all labels adjacent to v' (except possibly the label on e) then it is also true for the label on e adjacent to v . The result then follows inductively.

(c) The result concerning $d(v)$ follows easily from the two definitions of d and the fact that $\ell^v = d_B(v)$ proved in (b). For the description of $c(v)$, observe that a

least cost path through v is obtained by passing a path through these two branches B_1, B_2 at v for which the amount of $d(v)$ one can save by passing paths in B_1, B_2 through v is maximal over all branches at v.

Thus from the definition of $c(v)$ and the description of m^v in (b), the description in (c) is valid.

Remark: By (b) m^v is the most of $d_B(v)$ one can "save" by putting a path P into B through v and then considering $d_B(P)$ instead of $d_B(v)$.

Corollary 1

(a) Algorithm 1 finds the 1-core correctly.

(b) Let v be a vertex with adjacent labels $L_1^v, \ldots, L_p^v$. Let B_i, B_j be the branches at v for which m_i^v, m_j^v are maximal among the labels $m_1^v, \ldots, m_p^v$. Then a minimal cost path through v is given by joining a minimal cost path through v in B_i to one through B_j. Further any such minimal path can be so constructed.

(c) In Procedure A we have

$$\ell^v > \ell_i^{v'}, \quad m^v > m_i^{v'}, \quad n^v > n_i^{v'} \quad \text{(all } i\text{)}.$$

Proof:

(a) This follows from part (a) of the theorem and the description of $d(v)$ in (c) of the theorem.

(b) Follows from the definition of $c(v)$ and the description of $c(v)$ in part (c) of the theorem.

(c) Follows from the definitions of these quantities in Procedure A.

We conclude this section by giving an interesting result about 1-cores. It will serve also as an introduction to the later proofs. We found that this result appears in Slater [S1] However we include it for the sake of a complete treatment of the subject.

Theorem 2

Let $P_1, P_2, \ldots, P_k$ be 1-cores of T then
$P_1 \cap P_2 \cap \ldots \cap P_k \neq \phi$

Proof: Assume first to the contrary that P and Q are two disjoint 1-cores of tree T. Let ab be the shortest path joining P to Q. (See Figure 3). Let 1, 2, 3, 4 be endpoints of P,Q as shown.

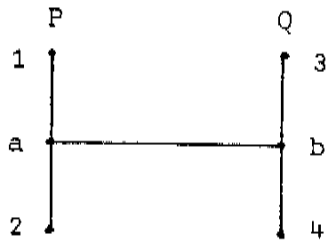


Figure 3

Denote the label at a in the branch containing vertex 1 by L_1^a etc. Then

$$m_1^a \geq m_3^a \qquad \text{(by Corollary 1(b) since } P \text{ is a 1-core)}$$
$$> m_3^b \qquad \text{(by Corollary 1(c))}$$

Similarly

$$m_3^b \geq m_1^b \qquad (Q \text{ is a 1-core})$$
$$> m_1^a$$

combining these we get $m_i^a > m_i^a$, a contradiction.
Hence $P_i \cap P_j \neq \phi \quad i \neq j$.

Suppose now $P_1 \cap P_2 \cap P_3 = \phi$. Hence there exists a vertex $u \in P_1 \cap P_2$ such that $u \notin P_3$ and a vertex $v \in P_1 \cap P_3$ such that $v \notin P_2$. The path $\{u,v\}$ in T is contained in P_1. If there is no vertex x on P_1 such that $x \in P_2 \cap P_3$ then there exists an edge e on $\{u,v\}$ such that removing e partitions T into two disjoint subtrees T^1 containing P_2 and T^N containing T_3. Hence $P_2 \cap P_3 = \phi$, a contradiction. Therefore $P_1 \cap P_2 \cap P_3 \neq \phi$.

The proof follows now by applying induction for $S_1, P_3, P_4, \ldots, P_k$ where $S_1 = P_1 \cap P_2$ using the fact that $S_1 \cap P_i \neq \phi$ for $i = 3, \ldots, k$.

Q.E.D.

Note that such a result exists for every set of paths in a tree satisfying the condition that any two paths intersect.

Theorem 2 implies that the set of vertices of T for which $c(v)$ is minimized (in Algorithm 1) consists of one (connected) subtree of T .

This theorem relates to the relation between a central vertex and a central path (for the same central measure) of a tree. In [S2] Slater shows that a path-centre of a tree contains the center of the tree and a branch weight centroid path (called a spine) contains the centroid of the tree. Hence all "central paths" (for each of the two cases) intersect at the "central vertex" of the tree.

However in [MS] it is shown that unlike the previous cases a median vertex (minimizing the sum of lengths to all vertices) of a tree is not necessarily contained in a core of the tree. Theorem 2 shows now that nevertheless all cores intersect in some central vertex although it is not necessarily the median vertex.

3. The First Algorithm for the 2-Core

An edge e is called *a bisector* of a simple path if it is a middle edge of the path (i.e. will be *the* middle edge if the number of edges in the path is odd).

Theoren 3

Let $\{P,Q\}$ be a 2-core of T and let e be a bisector of the path from P to Q. Let T_P be the component obtained by removing e which contains P and T_Q the component which contains Q. Then

(i) P is a 1-core of T_P and Q is a 1-core of T_Q.

(ii) $d(P,Q) = d_{T_P}(P) + d_{T_Q}(Q)$.

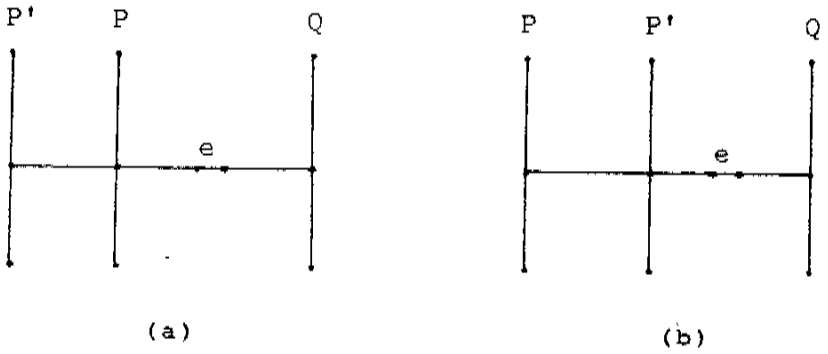
Proof: Each $v \in T_P$ satisfies $d(v,P) \leq d(v,Q)$, and each $v \in T_Q$ satisfies $d(v,Q) \leq d(v,P)$. Hence

$$d(P,Q) = \sum_{v \in V} \min \{d(v,P), d(v,Q)\} = \sum_{v \in T_P} d(v,P) + \sum_{v \in T_Q} d(v,Q)$$

$$= d_{T_P}(P) + d_{T_Q}(Q) \geq d_{T_P}(P') + d_{T_Q}(Q') \qquad (1)$$

where P', Q' are 1-cores of T_P, T_Q respectively. Suppose P is not a 1-core of T_P, so that $d_{T_P}(P') < d_{T_P}(P)$.

Assume firstly that the closest vertex of P' to Q is further or equidistant from e than the closest vertex of P to Q (see Figure 4(a)).



(a) (b)

Figure 4

Then

$$d(P',Q) = \sum_{\substack{v \text{ closer to} \\ P' \text{ than to } Q}} d(v,P') + \sum_{\substack{v \text{ closer to} \\ Q \text{ than to } P'}} d(v,Q) \qquad \text{(If } v \text{ is equidistant put it in any one sum)}$$

$$\leq \sum_{v \in T_P} d(v,P') + \sum_{v \in T_Q} d(v,Q) = d_{T_P}(P') + d_{T_Q}(Q)$$

(some v in T_P may be closer to Q than to P'),

$$< d_{T_P}(P) + d_{T_Q}(Q) = d(P,Q). \qquad \text{(by (1))}$$

Thus $d(P',Q) < d(P,Q)$, contradicting the assumption that $\{P,Q\}$ is a 2-core.

Assume next that the closest vertex of P' to Q is closer than the closest vertex of P to Q (Figure 4(b)). Then

$$d(P',Q) = \sum_{\substack{v \text{ closer to} \\ P' \text{ than to } Q}} d(v,P') + \sum_{\substack{v \text{ closer to} \\ Q \text{ than to } P'}} d(v,Q)$$

$$\leq \sum_{v \in T_P} d(v,P') + \sum_{v \in T_Q} d(v,Q) \qquad \text{(similar to above)}$$

$$= d_{T_P}(P') + d_{T_Q}(Q) < d_{T_P}(P) + d_{T_Q}(Q) = d(P,Q) \qquad \text{(by (1))}$$

and we have a contradiction as before. Hence P is a 1-core of T_P. Similarly, Q is a 1-core of T_Q.

Theorem 4

Let $\{P,Q\}$ be a 2-core of T, and let T_P, T_Q be as in Theorem 3. Let P', Q' be any 1-cores of T_P, T_Q respectively. Then $\{P',Q'\}$ is a 2-core of T.

<u>Proof:</u> $d(P',Q') \leq d_{T_P}(P') + d_{T_Q}(Q')$ (since a vertex in T_P might be closer to Q' than to P' and similarly for T_Q)

$$= d_{T_P}(P) + d_{T_Q}(Q) = d(P,Q) \quad \text{by Theorem 3.}$$

But $d(P,Q) \leq d(P',Q')$ since $\{P,Q\}$ is a 2-core.
Hence $d(P',Q') = d(P,Q)$ and $\{P',Q'\}$ is a 2-core also.

Q.E.D.

Theorem 4 implies the first algorithm for the 2-core of a tree. If e is an edge of T, let T_e^1, T_e^2 be components of T obtained by delteing e from T.

Algorithm 2 (2-core of T)

1. For each edge e of T use Algorithm 1 to calculate P_e = 1-core of T_e^1, Q_e = 1-core of T_e^2. Let $d_e = d(P_e, Q_e)$.

2. Let $d_{e'} = \min_{e \in T} d_e$. Then $\{P_{e'}, Q_{e'}\}$ is a 2-core of T.

The validity of Algorithm 2 follows from Theorem 4. For all edges of T we apply the linear Algorithm 1 for the two parts of the tree. Hence the complexity of the algorithm is $O(|V|^2)$.

4. The Second Algorithm for the 2-core

We continue to investigate further the relations between a 1-core and a 2-core a tree in order to obtain a more efficient algorithm for the 2-core.

Theorem 5

Let C be a 1-core of T. Then there exists a 2-core $\{P,Q\}$ of T satisfying one of the following :

(a) $P = C$

or

(b) One terminal vertex of C is a terminal vertex of P, while the other is a terminal vertex of Q.

Proof: Let $\{P,Q\}$ be a 2-core of T. Firstly, we show that C intersects P or Q.
Suppose that $P \cap C = Q \cap C = \phi$. Let $a \in P$ and $b \in Q$ be vertices on the path connecting P to Q. Suppose, firstly, that ab intersects C, at i say,

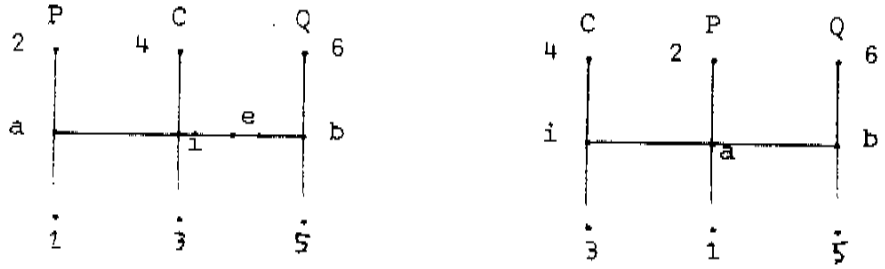


Figure 5

(See Figure 5(a)). We use the notation L_1^a etc. as in the proof of Theorem 2 to denote labels in T_P and L_1^{a*} etc. to denote labels in T. We choose e between P,Q as in Theorem 3. Suppose e lies on ib (the case e lies on ai is similar). By Theorem 3 we have P a 1-core of T_P, Q a 1-core of T_Q.

We have

$$m_2^a \geq m_4^a \qquad (P \text{ is a 1-core of } T_P)$$

$$> m_4^i \qquad (\text{Corollary 1(c)})$$

$$m_4^i = m_4^{i*} \geq m_2^{i*} = m_2^i \qquad (C \text{ is a 1-core of } T)$$

$$> m_2^a \qquad (\text{Corollary 1(c)}) .$$

Hence $m_2^a > m_2^a$, a contradiction. Hence i cannot lie between a and b .

Suppose there is a path i a b . (The case where a b i is a path is similar). (See Figure 5(b)). Then for $v \in T_Q$ we have

$$d(v,P) < d(v,C) .$$

Since P is a 1-core of T_P , we have

$$\sum_{v \in T_P} d(v,P) \leq \sum_{v \in T_P} d(v,C)$$

so

$$d(P) = \sum_{v \in T_P} d(v,P) + \sum_{v \in T_Q} d(v,P)$$

$$< \sum_{v \in T_P} d(v,C) + \sum_{v \in T_Q} d(v,C) = d(C) .$$

This contradicts the assumption that C is a 1-core. Hence C intersects P or Q .

Suppose C intersects P but not Q . Consider the two cases where the path connecting P and Q is either disjoint from C (Fig. 7(a)) or intersects C (Fig. 7(b)).

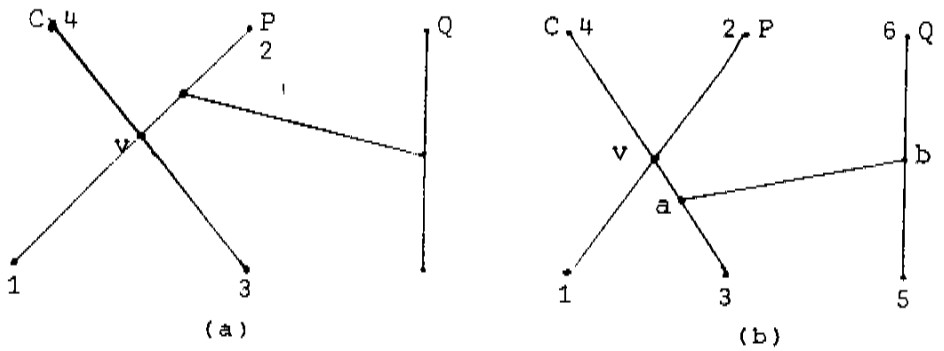


Figure 6

Consider the first case (Fig. 7(a)). Then

$m_4^v = m_4^{v*} \geq m_1^{v*} = m_1^v$ (C is a 1-core of T)

$m_1^v \geq m_4^v$ unless $4 = 2$. (P is a 1-core of T_P)

Hence $m_4^{v*} = m_1^{v*}$ or $4 = 2$. Similarly

$m_4^v = m_4^{v*} \geq m_2^{v*} \geq m_2^v$ (3)

$m_2^v \geq m_4^v$ or $4 = 1$ (4)

Hence $m_4^{v*} = m_2^{v*}$ or $4 = 1$.

In the same way, $m_3^{v*} = m_1^{v*}$ or $3 = 2$, $m_3^{v*} = m_2^{v*}$ or $3 = 1$.

So either 1v,3v and 2v,4v , or 1v,4v and 2v,3v have the same m^*-labels. Hence C is a 1-core of T_P . By Theorem 4, {C,Q} is a 2-core of T , and (a) holds.

We show that the second case (Fig. 7(b)) cannot occur. W.l.o.g. we may suppose that $m_6^{b*} \geq m_5^{b*}$. Let 6ba3 = Q' . Then {P,Q'} satisfies $d(P,Q') < d(P,Q)$. It is clear that if v is not a vertex of B_{5b} (branch at b containing 5) then $\min\{d(v,P),d(v,Q')\} \leq \min\{d(v,P),d(v,Q)\}$. On the other hand, we show that the amount we save (see remark after theorem 1) by putting a path into B_{3a} is > the amount we save by putting a path into B_{5b} . (For

$m_3^a \geq m_5^a$ (C is a 1-core of T)

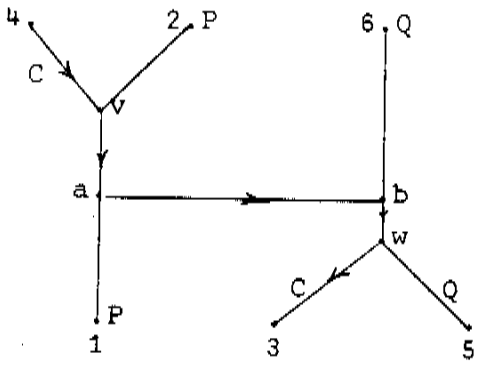
$> m_5^b$ (Corollary 1(c))

and this implies the result.) So

$$d(P,Q') = \sum_{v \in V} \min\{d(v,P), d(v,Q')\} < \sum_{v \in V} \min\{d(v,P),d(v,Q)\} = d(P,Q).$$

This contradicts the assertion that {P,Q} is a 2-core. So we see that if $P \cap C \neq \phi$, $Q \cap C = \phi$ then P is a 1-core of T i.e. (a) of the theorem holds.

Finally suppose $P \cap C \neq \phi$, $Q \cap C \neq \phi$. The situation is as in Figure 8.



Arrows show C .

Figure 7

Since the dividing edge e of {P,Q} lies on ab , it follows by a similar argument that {4va1, 6bw3} is a 2-core and condition (b) holds.

Corollary 2 : Let C be a 1-core and {P,Q} a 2-core of T . If C ≠ P and C ≠ Q then C contains the bisector of {P,Q} .

Proof: Follows from case (b) of Theorem 5.

We give now a more efficient algorithm for the 2-core implied by Theorem 5 and Corollary 2.
For the algorithm we need a new definition. Let w be a vertex of T . A path P in T not containing w is a *w-point core* if P minimizes d(w,Q} over all paths Q in T not containing w .

Let T be a tree rooted at v . In the next algorithm we apply a procedure to find the v-point core of T . Such a linear procedure is described later in this section.

In Algorithm 3 steps 2 and 3 correspond to cases (a) and (b) in Theorem 5. For each of these cases the 2-core is found and the best of both 2-cores is the 2-core of T .

Algorithm 3 (2-core of T).

1. Apply Algorithm 1 to find a 1-core C of T.
2. For each $r \in C$ and each branch B_i rooted at r apply the procedure to find a v-point core P_v^i of B_i and calculate $d_v^i = d(C,P_v^i)$. Let $d_{v_0}^{i_0} = d(C,P_{v_0}^{i_0}) = \min_{v,i} d(C,P_v^i)$
3. For each edge f of C let T_f^1 and T_f^2 be the subtrees obtained by deleting f from T. Let P_f be a 1-core of T_f and let Q_f be a 1-core of T_f. Let $d_f = d(P_f,Q_f)$ and let $d_{f_0} = \min_{f \in C} d_f$.
4. If $d_{v_0}^{i_0} < d_{f_0}$ then $\{C,P_{v_0}\}$ is a 2-core
   else $\{P_{f_0},Q_{f_0}\}$ is a 2-core.

The validity of Algorithm 3 follows from Theorem 5 Corollary 2 Theorem 4 and the validity of the v-point core procedure discussed later.

Complexity Analysis: Step 1 is linear. Step 2 can be implemented in linear time assuming linearity of the v-point core procedure. The complexity of Step 3 is bounded by (diameter $(T)).|V|$ and thus this is also the complexity of Algorithm 3. The average diameter of a tree depends on the probability model. However it is bounded by twice the average height of a rooted tree which is $O(\sqrt{n})$ (see e.g. [Flajolet] [RS] and [K]).

Consider now the r-point core of a tree T rooted at r. The procedure to find a r-point core is first presented on general lines and later we discuss a precise efficient implementation of the complex steps.

Procedure 4 (r-point core)

1. Apply Algorithm 1 to label all edges of T.

2. Scan T in Depth First Search order maintaining the path from the root r to the current vertex v in a vector A such that $r = A(1)$ and $v = A(i)$. The bisector $e_v = (s_v, t_v)$ of the path $r\,v$ is $(A(\lfloor i/2 \rfloor), A(\lfloor i/2 \rfloor + 1))$. Let R_v, E_v be the two subtrees obtained by deleting e_v from T such that $r \in R_v$, $v \in E_v$. Let E_v^u be the subtree obtained from E_v by deleting v its father edge and all its descendents.

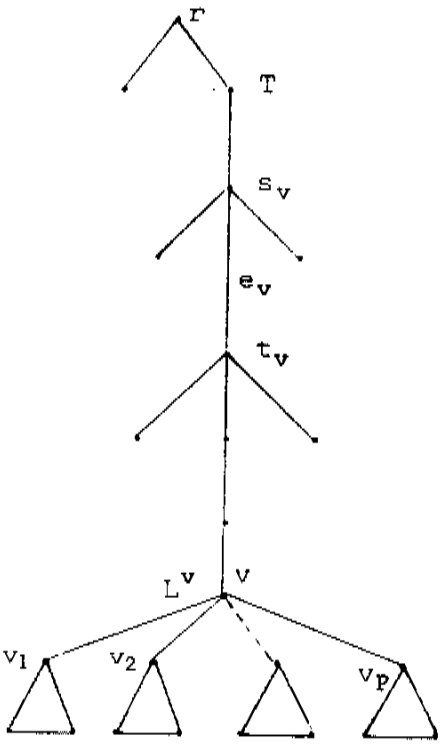


Figure 8

3. Find $d_{R_v}(r)$ and $d_{E_v^u}(v)$.
4. From the k sons of v find two sons i and j maximizing m_i^v and m_j^v.
5. Calculate
   (i) $c_r(v) = d_{R_v}(r) + d_{E_v^u}(v) + \sum_k \ell_k^v - m_i^v - m_j^v$.

(ii) A path P_v through v in $B' = B_{(v,v_i)} \cup B_{(v,v_j)}$ minimizing $d_{B'}(P_v)$.

6. Find a vertex v_0 minimizing $c_r(v)$ over $v \neq r$. Then P_{v_0} is an r-point core of T .

We give now precise implementations of step 3 and 5(ii).

Implementation of Step 3

Let r have sons $x_1, \ldots, x_q$ with upper labels L_i^r of (r,x_i) .
Let t_v have sons $w_1, \ldots, w_{q'}$ with upper labels $L_i^{t_v}$ of (t_v,w_i) . Then

$$d_{R_v}(r) = \sum_{i=1}^{q} \ell_i^r - \sum_{i=1}^{q'} \ell_i^{t_v} - \left(\sum_{i=1}^{q'} n_i^{t_v} \right). \quad \text{(depth } (t_v)).$$

Let L^{t_v} be the lower label of the father edge of t_v and let L^v be the lower label of the father edge of v .

$$d_{E_v^u}(v) = \ell^v - \ell^{t_v} - n^{t_v}(\lceil \text{depth } (v)/2 \rceil)$$

Implementation of Step 5(ii) :

Proceed from v into $B_{(v,v_i)}$ visiting vertices, passing along those edges with maximal m-labels among brother edges not yet passed along. When a terminal vertex is reached do the same from v into $B_{(v,v_j)}$. The path traversed is P_v .

The validity and linearity of the r-point core procedure can be shown now. The details are left to the reader.

5. Algorithm for the intersecting two-core

Theorem 6

(a) If $\{P,Q\}$ is a I2-core, then $P \cap Q \neq \phi$.

(b) Let C be a 1-core of T . Then there exists an I2-core of T containing C .

Proof:

(a) If $P \cap Q = \phi$, we can clearly choose two paths P',Q' in $P \cup Q \cup \{\text{path joining } P \text{ to } Q\}$ so that $d(P',Q') < d(P,Q)$, a contradiction.

(b) We outline the proof since it is much the same as previous proofs. Let C be a 1-core and let $\{P,Q\}$ be a I2-core of T . If C does not intersect $P \cup Q$, the situation is as in Figure 9(a).

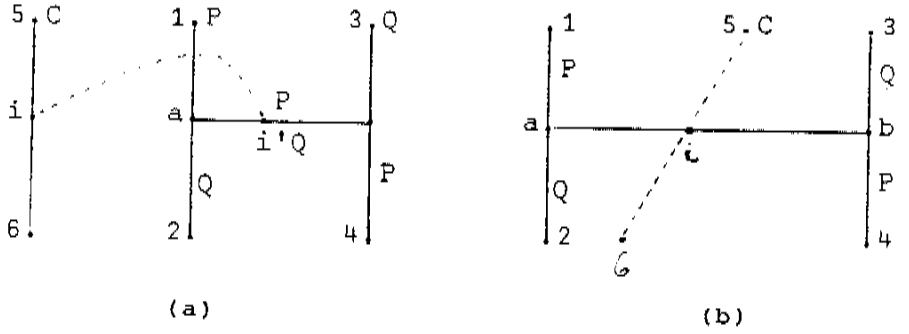


Figure 9

But substituting $5i\,i'$ for $1a\,i'$ would lead to a better I2-core. Hence any 1-core intersects any I2-core. A similar argument shows that C must intersect the subpath $P \cap Q$ (See Fig. 9(b)) since otherwise a better I2-core is obtained.

We show now that there exists an I2-core containing C . Suppose the situation is as in Fig. 9(b). As in the argument of Theorem 5 we must have the $5i$ and $6i$ belonging

to some other I2-core containing those two out of the 4 paths ia1, ia2, ib3, ib4 which save most. Thus the result follows.

Remark: There are three possibilities for an I2-core and they are illustrated in Figure 10.

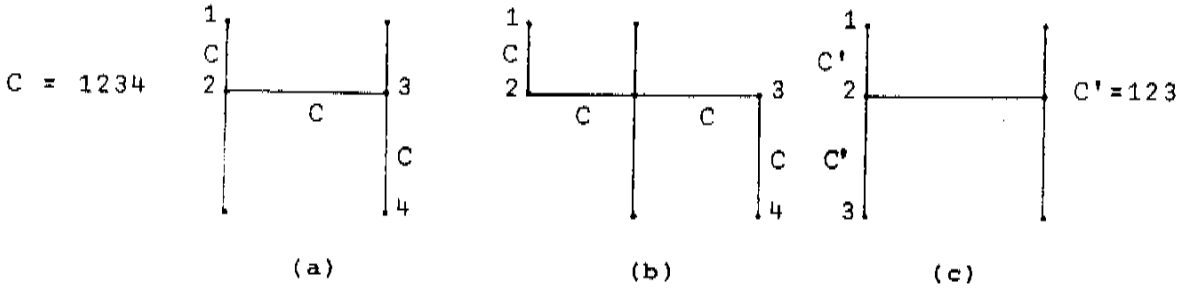


Figure 10 **Possible I2-core shapes**

Algorithm 4 (To find an I2-core)

1. Find a 1-core C of T.

2. (Accounts for the situation in Figure 10(a), (b).)
Find the vertices v, v' on C which have branches with maximal labels m^v, $m^{v'}$. Draw minimizing paths (i.e. maximizing the reduction of $d(C)$) in these branches from v, v' and let IC be the union of C and these paths. Calculate the cost of IC.

3. (Accounts for situations in Figure 10(c)).
For each vertex v of C and each branch at v not containing other vertices of C find v_1 v_2 v_3 such that that

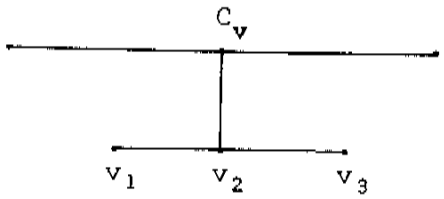


Figure 11

$d_{B_{v_1}}(vv_2 \cup v_1v_2v_3)$ is minimal over all such unions of two paths. Find the v with a path of this form which is

minimal and let IC be the union of C and this path.

4. The I2-core is the path of 2 or 3 with least cost.

Theorem 6 implies the validity of the algorithm.

Complexity Analysis: Steps 1 and 2 can be implemented in linear time. In step 3 finding the best vv_2 U $v_1v_2v_3$ for a given v_2 is linear and thus the complexity of this step for the whole tree is $|V| \cdot$diameter (T). Thus the complexity of algorithm 1 is $|V| \cdot$diameter (T) .

[FO] Flajolet, P and Odlyzko, A, The average height of binary trees and other simple trees. Rapports de Recherche No.56, INRIA, Rocquencourt, France (1981).

[GJ] Garey, M R and Johnson, D S, Computers and Intractibility, A Guide to the Theory of NP Completeness, Freeman, 1979.

[H] Harary, F, Graph Theory. Addison Wesley, Reading, Mass., 1969.

[HA] Hakimi, S L, Optimal location of switching centers and the absolute centers and medians of a graph, Operations Research 12 (1964), 450-459.

[K] Knuth, D E, The Art of Computer Programming, Vol. 1 Addison Wesley, Reading, Mass., 1968. Ex 2.3.1-11 and Ex 2.3.7.5-5.

[KH] Kariv, O and Hakimi, S L, An algorithmic approach to network location problems. II: The p-medians, Siam J Appl. Math. 37 (1979), 539-560.

[MS] Morgan, C A and Slater, P J, A linear algorithm for the core of a tree, J Algorithms 1 (1980), 247-258.

[RS] Renyi, A and Szekeres, S, On the height of trees, J Australian Math. Soc. 7 (1967), 497-507.

[S1] Slater, P.J., "Centrality of paths and vertices in a graph: Cores and pits", in the fourth International Conference on the Theory and Applications of Graphs, Western Michigan University (G. Chartrand et al ed.), John Wiley New York, 1980, 529 - 542.

[S2] Slater, P.J., "Locating central paths in a graph" Transportation Science 16 (1982) 57 -