# A Generalized Algorithm for Centrality Problems on Trees

ARNON ROSENTHAL

*Xerox Advanced Information Technology, Cambridge, Massachusetts*

AND

JOSÉ A. PINO

*Universidad de Chile, Santiago, Chile*

Abstract. A general framework is presented for rapidly analyzing tree networks to compute a measure of the *centrality* or *eccentricity* of all vertices in the tree. Several problems, which have been previously described in the literature, fit this framework. Some of these problems have no published solution better than performing a separate traversal for each vertex whose eccentricity is calculated. The method presented in this paper performs just two traversals and yields the eccentricities of all vertices in the tree. Natural sufficient conditions for the algorithm to work in linear time on any given problem are stated.

Categories and Subject Descriptors: F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Numerical Algorithms and Problems—*computation on discrete structures, routing and layout, searching*

General Terms: Algorithms, Verification

Additional Key Words and Phrases: Center, centralized network, centroid, eccentricity, facility location.

## 1. *Introduction*

Finding a vertex in a graph that is most central with respect to a given cost criterion can be useful to solve many optimization problems arising in applications (see [14], for example). The cost criterion defines costs associated to vertices, edges, traffic through edges, etc. For each of these criteria there is a corresponding definition of "eccentricity" of a vertex such that a "center" is a vertex with minimal "eccentricity." A large number of algorithms for finding centers, each designed for a certain cost structure, have appeared in the literature. The problems include finding 1-medians [6, 15, 18], placing telecommunication amplifiers ($r$-domination problem) [4, 17, 22], finding weighted centroids (weight-branch problem) [18], minimizing maximum distance from center (finding 1-center) [5, 7, 8, 10, 11, 17, 20], minimizing expected communication blockage [19], and minimizing power needed in a lossy network [21]. More references to these and related problems are found in [3], [12], and [24].

In most of the applications given below, each vertex $v$ is to receive a volume $c(v)$ of traffic (e.g., messages) from a central facility in a network. For simplicity, we restrict ourselves to a network T, which is a tree and to the case where the central facility consists of exactly one vertex. Each message has only one recipient. The traffic (denoted $t_{(u,v)}$) carried by an edge $(u, v)$ depends on the placement of the central facility; all vertices from the connected component of T—$\{(u, v)\}$ *not* containing the center have their traffic routed over $(u, v)$. Given the cost structure of a tree, we define the *eccentricity of a vertex v in that tree* (denoted simply ecc($v$)) to be the cost of distributing to all the vertices of the tree from a center located at $v$.

1.1 THE GENERAL PROBLEM. Given a tree and its cost structure, we want to compute simultaneously the "eccentricities" for all vertices in the tree. (In practice, this approach allows later choice of a center based on eccentricity and unquantifiable factors as well.) To compute eccentrity or "total" cost for all vertices, we introduce a generalized "schematic" algorithm. The method for expressing the algorithm's generality merits attention. Formally, the local cost structure is specified by subroutines (i.e., uninterpreted function symbols). Conditions are given explicitly such that for any admissible local cost functions, the algorithm will be valid and will run in linear time. Admissible cost structures must satisfy the following rather weak condition: If one roots the tree at an arbitrary vertex $r_o$, it must be possible to traverse the tree bottom-up, recursively computing information (called limb weights) such that the eccentricity of vertex $r_o$ may easily be computed from the weights of limbs incident to $r_o$.

The user supplies subroutines that specify the cost structure by calculations to be performed at each vertex. Our algorithm handles the global tree traversal and the cost calculations for the limb weights.

The contribution of this paper lies in four points: First, it incorporates under one framework many algorithms that were previously treated separately. Second, we abstract the cost manipulation and handle our algorithm independently of the details of the cost structure. Third, we observe that once we know the eccentricity of an arbitrarily chosen root, then a *single* downward traversal can provide *all* vertices' eccentricities (this property was already used in some previous algorithms [14]). Finally, we provide some conditions that ensure the linearity of an algorithm fitting the framework.

1.2 PREVIOUS WORK. In many of the applications below, attention has centered on computing ecc(root) in linear time for a complicated cost structure. To find the eccentricities of all vertices, one must repeat the computation for each vertex, for a total time of $O(|V|^2)$. For some of the applications, there is no previous algorithm that computes all the eccentricities for certain cost structures in linear time, although many applications possess algorithms that find the center in linear time.

For graphs formed by composition rules with limited attachment points, Bern et al. [1] and Takamizawa et al. [23] show how to find min/max weight subgraphs satisfying some *regular* property P. Weight is the sum of weights in the selected subgraph. Their algorithms perform a postorder traversal of the composition tree that produced the graphs; during that traversal they execute a dynamic program based on an assumption that each subgraph is in one of a small number of equivalence classes with respect to P.

The algorithm we present also begins with a postorder traversal of a tree, and makes assumptions about the cost function similar to the *homomorphism* assumption of Bern et al. [1]. But our algorithm handles more general computations of

weight (i.e., eccentricity) and performs *mass production* to find the eccentricity of every vertex. There is no dynamic program either, and the second pass is different.

## 2. *Terminology*

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

A *path* $P(v_i, v_j)$ is a sequence $v_i, \ldots, v_j$ of *distinct* elements of V such that there exists an edge $(v_k, v_l) \in E$ for any pair $v_k, v_l$ of consecutive vertices in the sequence.

We define a rooted tree $T(V, E, r_o)$ as 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 $u, v \in V$.

Let $v, w, \ldots, r_o$ be the path $P(v, r_o)$. Then $w$ is called father($v$) ($r_o$ does not have a father).

$$\begin{aligned} \text{Adj}(v) &\equiv \{u/u \in V \text{ and } (u, v) \in E\}, \\ \text{Sons}(v) &\equiv \text{Adj}(v) - \{\text{father}(v)\} \quad (\text{Sons}(v) \text{ is empty if } v \text{ is a leaf}), \\ \text{degree}(v) &\equiv |\,\text{Adj}(v)\,|, \end{aligned}$$

$l_{(u,v)}$ is the length of the edge $(u, v)$ (a nonnegative real number).

The edges satisfy $l_{(u,v)} = l_{(v,u)}$.

$d(u, v) \equiv \sum_{x_i, x_j \in P(u,v)} l_{(x_i, x_j)}$ is the distance from $u$ to $v$.

$\{c_j\colon V \to \text{Reals}/j \in J = \{1, \ldots, m\}\}$ is a set of vertex cost functions defined for some applications. $\{g_{(u,v)}\colon \text{Reals}^r \to \text{Reals}^s/(u, v) \in E, r, s \in N\}$ is a set of edge functions defined for some applications.

For a tree $T(V, E, r_o)$, we define $T_{(u,v)}(V_{(u,v)}, E_{(u,v)}, v)$ (see Figure 1) as a triple where $(u, v) \in E$, and $V_{(u,v)} = \{w \in V/u \notin P(v, w)\} \cup \{v\}$ and $E_{(u,v)} = \{(x, y)/(x, y) \in E; x, y \in V_{(u,v)}\}$.

Similarly, the *limb* $L_{(u,v)}(V_{(u,v)}, E_{(u,v)}, (u, v))$ is a triple where $(u, v) \in E$ and $V_{(u,v)} = \{w \in V/u \notin P(v, w)\} \cup \{v\}$ and $E_{(u,v)} = \{(x, y)/(x, y) \in E; x, y \in V_{(u,v)}\} \cup \{(u, v), (v, u)\}$.

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

In each example application below, each limb $L_{(u,v)}$ is given an associated quantity called "weight" denoted wt($u$, $v$) from which the eccentricities are computed. The weights are obtained from the input data and previously computed weights. The input data may be the edge lengths, vertex cost functions, and/or edge functions and depend on the specific application.

The computation of the eccentricities is also application-defined. The real-valued quantity ecc($u$) will be computed to mean the cost of the tree if the center is located at vertex $u$.

The weight wt($u$, $v$) of a limb $L_{(u,v)}$ 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 ecc($u$), and
(ii) if $L_{(w,y)}$ and $L_{(u,v)}$ are limbs such that $V_{(w,y)} \supseteq V_{(u,v)}$, then wt($u$, $v$) contains all the information about $L_{(u,v)}$ relevant to computing wt($w$, $y$).

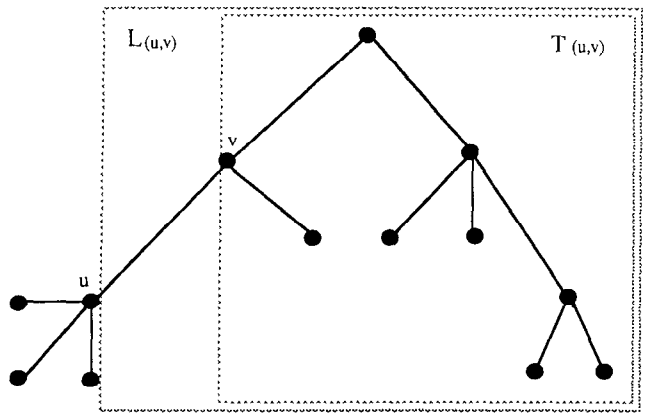


FIG. 1. $T_{(u,v)}$ and $L_{(u,v)}$.

The crucial idea is that for "reasonable" cost functions, it is possible to obtain such a compact representation of any limb. Intuitively, this representation expresses the contribution of the limb to the eccentricity of any vertex outside it.

The next two expressions define useful sets of weights.

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

## 3. *Applications*

Before presenting the algorithm, we list below some optimization problems, together with the corresponding definitions. The value of ecc($v$) will be determined by these (center-dependent) edge traffics.

(i) *Minimizing a Sum of Traffic-Dependent Edge Costs.* We wish to minimize the total message costs. Suppose it costs $g_{(x,y)}(t)$ to send $t$ units of traffic across edge($x$, $y$). Let ecc($v$) be the sum of all these (traffic-dependent) edge costs when $v$ is the center:

$$\text{ecc}(v) = \sum_{x \in \text{Adj}(v)} g_{(v,x)}(t).$$

References [6], [15], and [18] consider this problem in the special case where all $g(\cdot)$ are linear (this reduces to the problem to finding the 1-median of the tree). Goldman [6] and Hua [15] describe how to find the eccentricity of the 1-median in linear time. Kariv and Hakimi [18] propose methods to find $p$-medians and compute many eccentricities in a tree, but their algorithms are not linear, since they address a more complicated problem. Hedetniemi et al. propose a linear time algorithm to find the center [14].

(ii) *Telecommunications Amplifier Problem* (r-*Domination Problem*). In some telecommunication systems (e.g., cable television) a signal can travel only a limited distance without being amplified. We wish to locate the central signal distribution point so as to minimize the number of amplifiers required. The problem has many variants. Kariv and Hakimi [17] solve some of these versions in linear time. Other references are [4] and [22]. The Appendix contains a detailed specification of one of the variants.

(iii) *Minimize Maximum Congestion* (*Weighted Centroid*). We wish to minimize the maximum traffic carried by any single edge. If $v$ is the center, the maximum traffic will be on one of the edges $(v, x)$ out of $v$.

Define $\text{ecc}(v) = \max\{t_{(v,x)} \mid x \in \text{Adj}(v)\}$.

When each vertex receives 1 unit of traffic, the example reduces to the problem of finding the "centroid" of a tree [13]. Linear time algorithms exist for finding the weighted centroid [16]. The general problem is actually equivalent to finding the 1-median [18].

(iv) *Minimax Distance Location Problem* (1-*Center Problem*). Consider the problem of locating an emergency facility or pizza parlor for communities connected by a tree of roads. We wish to minimize the longest distance a patient or pizza must travel.

Define $\text{ecc}(v) = \max_{x \in V}\{\text{d}(v, x)\}$.

The simple case in which all the edge distances are the same is the problem of finding the "center" of the tree as defined in graph theory [13].

Handler [11] proposed a linear-time algorithm to find the 1-center for the general problem of edges with arbitrary nonnegative lengths; all eccentricities can also be found in linear time.

Halfin [10] introduced an additional vertex-dependent time delay at each vertex. Hakimi [8] and Hakimi et al. [9] proposed an alternative problem with vertex-dependent weights added in instead of time delays. Linear time solutions exist for all these problems [20]. See also [5], [7], and [17].

(v) *Lossy Network.* Let each vertex $v$ have a "demand" $c(v)$ and consider a distribution network (e.g., power or water) in which to supply $b$ units at end $v$ of edge $(u, v)$ one must supply $g_{(u,v)}(b)$ units at vertex $u$. $\text{ecc}(v)$ is the number of units required at vertex $v$ to supply the entire tree from $v$. Shekel [21] proposed an $O(|\text{V}|)$ algorithm to find the eccentricity of one vertex, restricted to linear $g(\cdot)$. By performing a binary search on the tree, that algorithm finds a vertex of minimum eccentricity in $O(|\text{V}| \log |\text{V}|)$ time.

(vi) *Minimum Expected Traffic Blockage.* Consider a network whose vertices and edges fail independently with known probabilities. Failed vertices block communication. In one version, all traffic comes from the vertices to the center, and $\text{ecc}(v)$ = expected amount of blocked traffic. A second version assumes that vertices communicate pairwise, but with all traffic routed via the center (e.g., for billing). In this case, $\text{ecc}(v)$ = expected number of vertex pairs disconnected by failures. Kershenbaum and Van Slyke [19] gave an algorithm for each version of the problem, in which $O(|\text{V}|)$ time is required for each eccentricity computed.

## 4. *Generalized algorithms*

We want to separate the main algorithm from the details of the cost structure. For each application, "weights" for limbs will be defined to serve as summaries which are sufficient for eccentricity calculations. For instance, in the centroid example (iii), the weight of a limb is simply the amount of traffic routed toward the limb. (The Appendix contains weight definitions for the other applications.) The algorithm to be proposed will work as long as the weights satisfy certain conditions shown below.

Although traversing the tree, the generalized algorithm will call subroutines (i.e., uninterpreted functions) COMBINE and EXTRACT to compute weights and eccentricities. EXTRACT takes the weights of all limbs incident to a vertex $v$, and computes $\text{ccc}(v)$. COMBINE takes a given information about the edge$(u, v)$ and

the vertex $v$ and combines it with the weights of the limbs $\{L_{(v,z)} \mid z \neq u\}$ (i.e., OTHER_WTS($v$, $u$)) and yields the weight wt($u$, $v$) of the limb $L_{(u,v)}$. The task of the tree traversal algorithm is to call COMBINE and EXTRACT in the proper order with the proper arguments when the relevant weights are known.

The manipulations performed by COMBINE and EXTRACT are not complicated. In all the applications mentioned, there are very simple algorithms that traverse a rooted tree bottom-up to compute ecc(root). At each $v \neq$ root, these algorithms use a function analogous to COMBINE to compute wt(father($v$), $v$) from OTHER_WTS($v$, father($v$)). An analogue of EXTRACT is used to compute ecc(root). Our contribution at this point is the observation that a single downward traversal from the root can provide *all* vertices' eccentricities.

Conditions 1 and 2 below apply to the cost structure. They give a rigorous meaning to the statement that weights must effectively summarize limbs.

*Conditions.* There exist functions COMBINE and EXTRACT, and limb weights such that

(1) ecc($v$) = EXTRACT($v$, WTS($v$)).
(2) wt($u$, $v$) = COMBINE($u$, $v$, OTHER_WTS($v$, $u$)).

Note that EXTRACT($v$, WTS($v$)) [respectively, COMBINE($u$, $v$, OTHER_WTS($v$, $u$))] may be involved only after WTS($v$) [OTHER_WTS($v$, $u$))] has been computed. Also note that when $v$ is a leaf, OTHER_WTS($v$, $u$) is empty. Thus, a bottom-up tree traversal is needed to compute COMBINE for all vertices.

The mechanics of storing and passing the vector arguments to the subroutines for EXTRACT and COMBINE are uninteresting, so we shall assume that the vectors WTS($v$) and OTHER_WTS($u$, $v$) are available as soon as their component weights have been computed.

In order to perform meaningful analysis of running time, it will be necessary to specify running times for COMBINE and EXTRACT. It will not be sufficient to count function calls, since some invocations take much more time than others.

*Condition* 1T. EXTRACT($v$, WTS($v$)) may be computed in time $O$(degree($v$)).

*Condition* 2T. COMBINE($u$, $v$, OTHER_WTS($v$, $u$)) may be computed in time $O$(degree($v$)).

Conditions 1T and 2T are designed so that the overall algorithm will take linear time on trees without high-degree vertices (the "1T" and "2T" denomination emphasizes the *time* complexity). These conditions also rule out trivially defining the weight of a limb as (some appropriate encoding of) the limb itself. To satisfy 1T and 2T, weights will need to be *succinct* summaries.

## 5. *A Simple (Nonlinear) Algorithm*

After arbitrarily choosing a root $r_0$, it will be necessary to traverse the tree twice, once upward (i.e., from the leaves to the root), and once downward. In the first traversal, before a vertex is processed, all of its sons must be processed. This will be called a "bottom-up traversal." In the second traversal, a vertex is always processed before any of its sons. This will be called a "top-down traversal." The conventional postorder [preorder] traversal is a satisfactory bottom-up [top-down] traversal.

We now present the first algorithm for computing eccentricities. During the bottom-up traversal, the weights wt(father($v$), $v$) are computed using COMBINE.

Then, the top-down traversal computes wt($w$, $v$), for $w \in$ Sons($v$) using COMBINE, and also the eccentricities using EXTRACT.

```
      PROCEDURE central_vertex
      BEGIN
(1)     v := first_vertex_of_bottom_up_traversal
(2)     WHILE v ≠ r_o
          BEGIN
(3)       wt(father(v), v) := COMBINE(father(v), v, OTHER_WTS(v, father(v)));
            // Note that entries in OTHER_WTS(v, father(v)) were computed //
            // when Sons(v) were processed //
(4)       v := bottom_up_successor(v)
          END;
(5)     v := r_o;  // SECOND TRAVERSAL //
(6)     ecc(r_o) := EXTRACT(r_o, WTS(r_o));
        // All the entries in WTS(r_o) have been computed in the first phase //
        REPEAT
(7)     FOR each son w of v
        // This loop computes {wt(w, v) | w ∈ Sons(v)} //
        BEGIN
(8)       wt(w, v) := COMBINE(w, v, OTHER_WTS(v, w));
          // All but one entry in OTHER_WTS(v, w) was computed in the first
            phase. //
          // The exception [wt(v, father(v))] was computed earlier in the second traversal,
            when //
          // father(v) was processed //
(9)       ecc(w) := EXTRACT(w, WTS(w));
          // Step (8) just supplied the last entry in WTS(w) //
        END;
(10)    v := top_down_successor(v);  // (non-leaves) top down //
(11)  UNTIL all _non_leaf_vertices_have_been_visited
      END;
```

Figure 2 illustrates the limb weight computations in the weighted centroid problem (iii). Recall that the weight of a limb is the traffic destined for vertices in the limb. Each value wt($u$, $v$) is represented on the edge ($u$, $v$).

The COMBINE function in this case is

$$\mathrm{wt}(u, v) := c(v) + \sum_{\substack{x \in \mathrm{Adj}(v) \\ x \neq u}} \mathrm{wt}(v, x).$$

Thus, wt($r_o$, $v_4$) is computed as 4 + (5 + 2 + 3).

*Complexity.* The tree traversal "successor" routines (lines (4) and (10)) take a total time $Q(|\,V\,|)$. We note that in a tree, as previously defined, $\sum_v \mathrm{degree}(v) = |\,E\,| = 2(|\,V\,| - 1) = O(|\,V\,|)$. Appealing to Condition 1T [respectively, 2T], we see that the total time over all interactions at lines (6) and (9) [respectively, (3)] is $O(|\,V\,|)$. A careful check of Condition 2T shows that line (8) requires time

$$O\left\{\sum_{\text{nonleaves}}\left[\sum_{w \in \mathrm{Sons}(v)} \mathrm{degree}(v)\right]\right\} = O\left\{\sum_{\text{nonleaves } v} [\mathrm{degree}(v)]^2\right\}.$$

Thus, the time required for Algorithm 1 is $O(|\,V\,| + \sum [\mathrm{degree}(v)]^2)$.

## 6. *A Linear-Time Algorithm*

For the algorithm to run in linear time on all trees, steps (7)–(9) must be altered to mass-produce {wt($w$, $v$) | $w \in$ Sons($v$)} in total time $O$(degree($v$)).

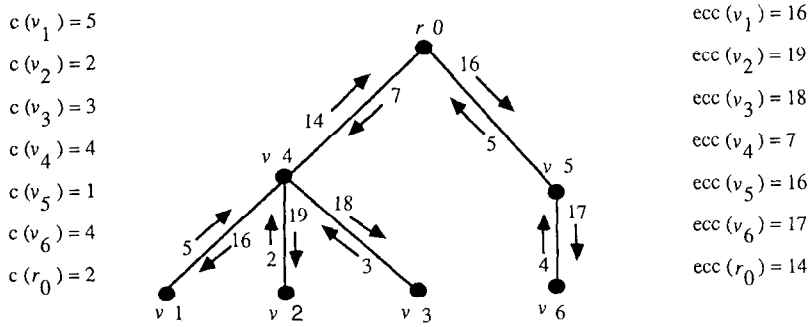


FIG. 2. Tree rooted at $r_0$. Weights computed after the first (↑) and second (↓) loops of Algorithm 1 for the weighted centroid problem.

A scheme to make those changes is based on these expressions for the vertex and edge sets of the limbs:

$$V_{(w,v)} = \left[\bigcup_{x \in \mathrm{Adj}(v)-\{w\}} V_{(v,x)}\right] \cup \{v\}, \qquad E_{(w,v)} = \left[\bigcup_{x \in \mathrm{Adj}(v)-\{w\}} E_{(v,x)}\right] \cup \{(w, v), (v, w)\}.$$

The obvious approach then is to compute a summary (called SMR($v$)) of the union of the vertex and edge sets of *all* limbs incident to $v$, and then find a way to subtract off limb($v$, $w$) and add in pertinent information on vertex $v$ and edges ($w$, $v$) and ($v$, $w$). Rather surprisingly, in every application considered, it is possible to define such a "subtraction" operation (although occasionally SMR must be defined for a 2 or 3-element vector).

A very simple example of a SMR function is found in the centroid problem (iii). In this case the relevant information to be added is the cost $c(v)$. Recall $\mathrm{wt}(w, v) = \sum_{x \in V(w,v)} c(x)$. If we define SMR by $\mathrm{SMR}(v) = \sum_{x \in \mathrm{Adj}(v)} \mathrm{wt}(v, x)$, then $\mathrm{wt}(w, v) = \mathrm{SMR}(v) - \mathrm{wt}(v, w) + c(v)$.

The following are the conditions needed to modify the algorithm:

*Condition* 3. There exists a function MAKESUM such that

$$\mathrm{SMR}(v) = \mathrm{MAKESUM}(v, \mathrm{WTS}(v)).$$

*Condition* 3T. MAKESUM($v$, WTS($v$)) can be computed in time $O(\mathrm{degree}(v))$.

*Condition* 4. There exists a function SUBTRACT such that

$$\mathrm{wt}(w, v) = \mathrm{SUBTRACT}(v, w, \mathrm{SMR}(v), \mathrm{wt}(v, w)).$$

*Condition* 4T. SUBTRACT($v$, $w$, SMR($v$), wt($v$, $w$)) can be computed in time $O(1)$.

The MAKESUM and SUBTRACT, together, form a COMBINE operation. Time is saved because MAKESUM need only be performed once at each vertex.

Assuming Conditions 3 and 4 are satisfied, we replace the second loop (statements (7)–(9), leaving (9) unchanged) by

```
(7.0) SMR(v) = MAKESUM(v, WTS(v))
(7.1) FOR each son w of v
        BEGIN
(8)       wt(w, v) = SUBTRACT(v, w, SMR(v), wt(v, w))
(9)       ecc(w) = EXTRACT (wm, WTS(w))
        END
```

Conditions 3T and 4T are satisfied by all the applications listed in Section 3. Under these assumptions, it is easy to see that new steps (7.0)–(8) take time $O(\text{degree}(v))$, and hence the entire algorithm is linear in time.

For the numerical example of the centroid problem shown in Figure 2, SMR($v$) is simply the total demand of the tree from vertices other than $v$, for example, $\text{SMR}(v_4) = 5 + 2 + 3 + 7 = 17$, and SUBTRACT is defined by

$$\text{SUBTRACT}(v, w, \text{SMR}(v), \text{wt}(v, w)) = \text{SMR}(v) - \text{wt}(v, w) + c(v).$$

Thus, $\text{wt}(v_2, v_4) = 17 - 2 + 4 = 19$.

## 7. *Final Remarks*

The Appendix contains definitions of weights and summaries and the main components of the MAKESUM, SUBTRACT, and EXTRACT functions for the previously described applications. They show that the linear algorithm can indeed be applied to obtain all eccentricities and the center for all these problems.

Future work to try to extend the current approach to more general classes of graphs can be done. Chen et al. [2] have shown that if the 1-median problem is solved on the blocking graph (which is a tree if the original graph is connected), then the solution obtained either solves the 1-median problem on the original graph or else identifies a single block that contains every 1-median. Thus, an interesting problem should be to extend the algorithm to the blocking graph to see if similar results can be obtained for any of the applications. Also, research can be done to handle sensitivity analysis problems for the more general classes of graphs considered by Bern et al. [1], using their composition tree, but the second phase of our algorithm.

## *Appendix. Algorithm CENTRAL_VERTEX Applied to the Problems Mentioned in Section 3*

The following gives a description of weights that can be used in the Algorithms 1 and 2 to compute the eccentricities in the problems mentioned under "Applications." Equations are then given for the functions COMBINE and EXTRACT in Algorithm 1. For Algorithm 2, a description of the SMR is given and then equations for MAKESUM and SUBTRACT. Since the weight description for Algorithm 1 is the same as for Algorithm 2, COMBINE and EXTRACT for Algorithm 2 are the same as COMBINE and EXTRACT for Algorithm 1. Application 1 is more fully annotated than the others.

The short notation $f(v)$ will be used to represent father($v$). The functions MAX and TOTAL with vector arguments will yield the largest component of the vector and the sum of the components of the vector, respectively. For example,

$$\text{MAX}(\text{WTS}_1(v)) = \text{Max}\{\text{wt}_1(v, z) \mid z \in \text{Adj}(v)\}.$$

(i) *Minimizing a Sum of Traffic-Dependent Edge Costs.* Inputs to this problem are the edge functions $g_{(u,v)}$: Reals $\rightarrow$ Reals (for each edge $(u, v)$), and the demands $c$: V $\rightarrow$ Reals.

*Algorithm* 1. The weight of a limb $L_{(u,v)}$ is a two-component vector. $\text{wt}_1(u, v)$ is the total traffic in the subtree $T_{(u,v)}$. $\text{wt}_2(u, v)$ is the cost of transporting the traffic from $u$ to the limb's other vertices. The following formulas define COMBINE, which is a vector-valued function,

$$\text{wt}_1(u, v) := \text{total}(\text{OTHER\_WTS}_1(v, u)) + c(v),$$
$$\text{wt}_2(u, v) := \text{total}(\text{OTHER\_WTS}_2(v, u)) + g_{(u,v)}(\text{wt}_1(u, v)).$$

The next formula defines EXTRACT:

$$\text{ecc}(v) := \text{total}(\text{WTS}_2(v)).$$

*Algorithm* 2. The SMR of a vertex is a two component vector. $\text{SMR}_1(v)$ is the total traffic in the limb except for $v$. $\text{SMR}_2(v)$ is the total cost of transporting the traffic to vertices other than $v$ from $v$.

These formulas define MAKESUM:

$$\text{SMR}_1(v) := \text{total}(\text{WTS}_1(v)),$$
$$\text{SMR}_2(v) := \text{total}(\text{WTS}_2(v)).$$

These formulas define SUBTRACT:

$$\text{wt}_1(v, \text{f}(v)) := \text{SMR}_1(\text{f}(v)) - \text{wt}_1(\text{f}(v), v) + c(\text{f}(v)),$$
$$\text{wt}_2(v, \text{f}(v)) := \text{SMR}_2(\text{f}(v)) - \text{wt}_2(\text{f}(v), v) + g_{(v,\text{f}(v))}(\text{wt}_1(v, \text{f}(v))).$$

(ii) *Minimizing Amplifiers in a Telecommunications System.* The telecommunications problem was not fully specified in Section 3 because it has many variants (all variants we have considered seem to satisfy the conditions). For the sake of example, suppose:

(1) amplifiers may be placed anywhere on an edge,
(2) the distance downstream from an amplifier (or the distribution point) to the next amplifier (or end of line) cannot exceed a fixed length **L**,
(3) there is no penalty for splitting a signal from an amplifier when it is received by more than one place.

Inputs to the problem are then the real constant **L** and the edge lengths $l_{(u,v)}$.

*Algorithm* 1. The weight of a limb $L_{(u,v)}$ will be a vector with two components. $\text{wt}_1(u, v)$ is the longest distance from $u$ to an amplifier in the limb. $\text{wt}_2(u, v)$ is the number of amplifiers in the limb. $\delta$ is a small positive real number (smaller than **L** or any distance).

Let

$$h(u, v) := \begin{cases} d(u, v) - \delta & \text{if } v \text{ is leaf} \\ d(u, v) & \text{otherwise} \end{cases}$$
$$\text{wt}_1(\text{f}(v), v) := [\max(\text{OTHER\_WTS}_1(v, \text{f}(v)) + h(\text{f}(v), v)] \bmod \mathbf{L},$$
$$\text{wt}_2(\text{f}(v), v) := \text{total}(\text{OTHER\_WTS}_2(v, \text{f}(v)) + \text{Truncate}[[\max(\text{OTHER\_WTS}_1(v, \text{f}(v)) + h(\text{f}(v), v)]/\mathbf{L}],$$
$$\text{ecc}(v) := \text{total}(\text{WTS}_2(v)).$$

*Algorithm* 2. The SMR of a vertex $v$ is a three element vector. $\text{SMR}_1(v)$ is the largest distance from $v$ to a needed amplifier (assuming $v$ is the center). $\text{SMR}_2(v)$ is the second largest distance from $v$ to a needed amplifier. $\text{SMR}_3(v)$ is the total number of amplifiers needed if $v$ is the center (i.e., $\text{ecc}(v)$).

$$\text{SMR}_1(v) := \max(\text{WTS}_1(v)).$$

Pick $z$ such that $\text{wt}_1(v, z) = \text{SMR}_1(v)$ (any such $z$ will do if there is more than one). Then,

$$\text{SMR}_2(v) := \max(\text{OTHER\_WTS}_1(v, z)),$$
$$\text{SMR}_3(v) := \text{total}(\text{WTS}_2(v)).$$

Define

$$S := \begin{cases} SMR_1(v) & \text{if } wt_1(f(v), v) \neq SMR_1(v) \\ SMR_2(v) & \text{otherwise} \end{cases}$$

$$wt_1(v, f(v)) := [S + h(v, f(v))] \bmod \mathbf{L},$$

$$wt_2(v, f(v)) := SMR_3(f(v)) - wt_2(f(v), v) + \text{Truncate}\left(\frac{S + h(v, f(v))}{\mathbf{L}}\right).$$

(iii) *Centroid Problem.* Inputs to this problem are the demands at vertices, $c$: $V \rightarrow$ Reals.

*Algorithm* 1. The weight of a limb is the total demand of the vertices in the limb.

$$wt(f(v), v) := \text{total}(\text{OTHER\_WTS}(v, f(v))) + c(v),$$
$$ecc(v) := \max(\text{WTS}(v)).$$

*Algorithm* 2. The SMR of a vertex $v$ is the total demand of the tree from vertices other than $v$.

$$SMR(v) := \text{total}(\text{WTS}(v)),$$
$$wt(v, f(v)) := SMR(f(v)) - wt(f(v), v) + c(f(v)).$$

(iv) *Minimax Distance Location Problem* (*1-Center Problem*). Inputs to this problem are the edge lengths $l_{(u,v)}$.

*Algorithm* 1. The weight of a limb $L_{(f(v),v)}$ is the maximum distance from $f(v)$ to any vertex $z$ in the limb. Note that the path from $f(v)$ to $z$ must have $v$ as the first vertex after $f(v)$.

$$wt(f(v), v) := \max(\text{OTHER\_WTS}(v, f(v))) + l_{(f(v),v)},$$
$$ecc(v) := \max(\text{WTS}(v)).$$

*Algorithm* 2. The SMR of a vertex $v$ is a two-component vector. $SMR_1(v)$ is the maximum distance from $v$ to any point in the tree (so it is also the eccentricity of $v$!). If $u$ is the first vertex on a path from $v$ to a vertex where $SMR_1(v)$ is achieved, then $SMR_2(v)$ is the longest distance from $v$ to any point where the path does not go through $u$.

$$SMR_1(v) := \max(\text{WTS}(v)).$$

Pick $u$ such that $wt(v, u) = SMR_1(v)$ (any such $u$ will do, if there is more than one). Then,

$$SMR_2(f(v)) := \max(\text{OTHER\_WTS}(v, u))$$

$$wt(v, f(v)) := \begin{cases} SMR_1(f(v)) + l_{(v,f(v))} & \text{if } wt(f(v), v) \neq SMR_1(f(v)), \\ SMR_2(f(v)) + l_{(v,f(v))} & \text{otherwise.} \end{cases}$$

(v) *Lossy Network.* The inputs are the demands $c$: $V \rightarrow$ Reals and the edge functions $g_{(u,v)}$: Reals $\rightarrow$ Reals.

*Algorithm* 1. The weight of a limb $L_{(f(v),v)}$ is the number of units needed at $f(v)$ to supply the vertices in the limb.

$$wt(f(v), v) := g_{(f(v),v)}(\text{total}(\text{OTHER\_WTS}(f(v), v)) + c(v)),$$
$$ecc(v) := \text{total}(\text{WTS}(v)).$$

*Algorithm* 2. The SMR of a vertex $v$ is the total power needed to supply the vertices other than $v$ from $v$.

$$\mathrm{SMR}(v) := \mathrm{total}(\mathrm{WTS}(v)),$$
$$\mathrm{wt}(v, \mathrm{f}(v)) := g_{(v,\mathrm{f}(v))}(\mathrm{SMR}(\mathrm{f}(v)) - \mathrm{wt}(\mathrm{f}(v), v) + c(\mathrm{f}(v)).$$

(vi) *Minimizing Expected Blocked Traffic.* The version shown here assumes traffic coming from the vertices to the center. The inputs are the functions $c_i$: V $\rightarrow$ Reals, representing the traffic generated at each vertex, and P: V $\rightarrow$ Reals, the probability of failure of each vertex (the range is, more precisely in this case, the real interval [0, 1]).

*Algorithm* 1. The weight of a limb $L_{(\mathrm{f}(v),v)}$ has two parts. $\mathrm{wt}_2(\mathrm{f}(v), v)$ is the expected traffic within the limb. $\mathrm{wt}_1(\mathrm{f}(v), v)$ is the normal traffic from the limb to $\mathrm{f}(v)$.

$$\mathrm{wt}_1(\mathrm{f}(v), v) := \mathrm{total}(\mathrm{OTHER\_WTS}_1(v, \mathrm{f}(v))) + \mathbf{c}(v),$$
$$\mathrm{wt}_2(\mathrm{f}(v), v) := [1 - p(v)]^*\mathrm{total}(\mathrm{OTHER\_WTS}_2(v, \mathrm{f}(v))) + p(v)^*\mathrm{wt}_1(\mathrm{f}(v), v),$$
$$\mathrm{ecc}(v) := [1 - p(v)]^*\mathrm{total}(\mathrm{WTS}_2(v)) + p(v)^*[\mathrm{total}(\mathrm{WTS}_1(v)) + c(v)].$$

*Algorithm* 2. The SMR of a vertex $v$ is a two-component vector. $\mathrm{SMR}_1(v)$ is the normal traffic from the rest of the tree to $v$. $\mathrm{SMR}_2(v)$ is the expected amount of traffic that will not reach $v$ from the rest of the tree.

$$\mathrm{SMR}_1(v) := \mathrm{total}(\mathrm{WTS}_1(v)),$$
$$\mathrm{SMR}_2(v) := \mathrm{total}(\mathrm{WTS}_2(v)),$$
$$\mathrm{wt}_1(v, \mathrm{f}(v)) := \mathrm{SMR}_1(\mathrm{f}(v)) - \mathrm{wt}_1(\mathrm{f}(v), v) + c(\mathrm{f}(v)),$$
$$\mathrm{wt}_2(v, \mathrm{f}(v)) := [1 - p(\mathrm{f}(v))]^*[\mathrm{SMR}_2(\mathrm{f}(v)) - \mathrm{wt}_2(\mathrm{f}(v), v)] + p(\mathrm{f}(v))^*\mathrm{wt}_1(v, \mathrm{f}(v)).$$

ACKNOWLEDGMENTS. We would like to thank Mark Hersey and Michael Coulter for their help in writing this paper. We also thank the anonymous referees for their valuable contribution.

REFERENCES

1. BERN, M. W., LAWLER, E. L., AND WONG, A. L. Why certain subgraph computations require only linear time. In *Proceedings of the 26th Annual Symposium on Foundations of Computer Science* (Portland, Ore., Oct.). IEEE, New York, 1985, pp. 117–125.
2. CHEN, M. L., FRANCIS, R. L., LAWRENCE, J. F., LOWE, T. J., AND TUFEKCI, S. Block-vertex duality and the one-median problem. *Networks 15* (1985), 395–412.
3. CHRISTOFIDES, N. *Graph Theory, An Algorithmic Approach.* Academic Press, Orlando, Fla., 1975, Ch. 5–6.
4. COCKAYNE, E. J., GOODMAN, S. E., AND HEDETNIEMI, S. T. A linear algorithm for the domination number of a tree. *Inf. Proc. Lett. 4* (1975), 41–44.
5. FARLEY, A. M. Vertex centers of trees. *Trans. Sci. 16*, 3 (Aug. 1982), 265–280.
6. GOLDMAN, A. J. Optimal center location in simple networks. *Trans. Sci. 5*, 2 (1971), 212–221.
7. GOLDMAN, A. J. Minimax location of a facility in a network. *Trans. Sci. 6*, 4 (Nov. 1972), 407–418.
8. HAKIMI, S. L. Optimum locations of switching centers and the absolute centers and medians of a graph. *Op. Res. 12*, 3 (May–June 1964), 450–459.
9. HAKIMI, S. L., SCHMEICHEL, E. F., AND PIERCE, J. G. On $p$-centers in Networks. *Trans. Sci. 12*, 1 (Feb. 1978), 1–15.
10. HALFIN, S. On finding the absolute and vertex centers of a tree with distances. *Trans. Sci. 8*, 1 (Feb. 1974), 75–77.
11. HANDLER, G. Y. Minimax location of a facility in an undirected tree graph. *Trans. Sci. 7*, 3 (Aug. 1973), 287–293.
12. HANSEN, P., LABBE, M., PEETERS, D., AND THISSE, J. F. Single facility location on networks. *Ann. Discrete Math. 31* (1987), 113–145.

13. HARARY, F. *Graph Theory.* Addison-Wesley, Reading, Mass., 1969, Ch. 2–4.
14. HEDETNIEMI, S. M., COCKAYNE, E. J., AND HEDETNIEMI, S. T. Linear algorithms for finding the Jordan center and path center of a tree. *Trans. Sci. 15*, 2 (May 1981), 98–114.
15. HUA, L. K., ET AL. Application of mathematical methods to wheat harvesting. *Math. Sin. 11*, 1 (1961), 77–91.
16. KANG, A., AND AULT, D. Some properties of a centroid of a free tree. *Inf. Proc. Lett. 4* (1975), 18–20.
17. KARIV, O., AND HAKIMI, S. L. An algorithmic approach to network location problems, I: The $p$-centers. *SIAM J. Appl. Math. 37*, 3 (1979), 513–538.
18. KARIV, O., AND HAKIMI, S. L. An algorithmic approach to network location problems, II: The $p$-medians. *SIAM J. Appl. Math. 37*, 3 (1979), 539–560.
19. KERSHENBAUM, A., AND VAN SLYKE, R. Recursive analysis of network reliability. *Networks 3* (1973), 81–94.
20. MEGIDDO, N. Linear-time algorithms for linear programming in $R^3$ and related problems. *SIAM J. Comput. 12*, 4 (Nov. 1983), 759–776.
21. SHEKEL, J. Optimal source location to feed a lossy distribution tree. *IEEE Trans. Circ. Theory 20*, 3 (May 1973), 246–250.
22. SLATER, P. J. R-Domination in Graphs. *J. ACM 23* (1976), 446–450.
23. TAKAMIZAWA, K., NISHIZEKI, T., AND SAITO, N. Linear-time computability of combinatorial problems on series-parallel graphs. *J. ACM 29*, 3 (July 1982), 623–641.
24. 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.