Theoretical Computer Science 19 (1982) 1–16
North-Holland Publishing Company

ON THE COMPLEXITY OF EDGE LABELINGS FOR TREES*

Y. PERL

Department of Mathematics and Computer Science, Bar-Ilan University, Ramat-Gan, Israel

S. ZAKS

Department of Computer Science, Technion, Haifa, Israel

Communicated by R. Karp
Received February 1979
Revised February 1980

Abstract. Given a tree T with n edges and a set W of n weights, we deal with labelings of the edges of T with weights from W, optimizing certain objective functions. For some of these functions the optimization problem is shown to be NP-complete (e.g., finding a labeling with minimal diameter), and for others we find polynomial-time algorithms (e.g., finding a labeling with minimal average distance).

1. Introduction

1.1. *Definitions*

Let $T(V, E)$ be a tree with vertices $V=\{1, 2, \ldots, n+1\}$ and edges $E=\{e_1, e_2, \ldots, e_n\}$. A tree is considered to be unrooted, unless otherwise stated. A vertex of degree one is a *leaf*, and an edge incident with a leaf is a *terminal edge*. P_n denotes a path with n vertices. $W=\{w_1, w_2, \ldots, w_n\}$ is a set of *weights* such that $0 \leqslant w_{\min} = w_1 \leqslant w_2 \leqslant \cdots \leqslant w_n = w_{\max}$. A *labeling* of the edges of T with weights from W is a bijection $f: E \to W$. In this paper we consider labelings that optimize certain objective functions.

Let $p(i, j)$ denote the (unique) path connecting vertices i and j in T. Given a labeling f of T, the *distance* $d_f(i, j)$ between i and j is

$$d_f(i, j) = \sum_{e \in p(i,j)} f(e).$$

The *diameter* $D_f(T)$ of the tree T, labeled with f, is

$$\mathcal{D}_f(T) = \max_{i<j} \{d_f(i, j)\}.$$

* This work was done while both authors were at the Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL 61801, and was supported in part by the National Science Foundation under grant NSF MCS 77-22830.

Note. We omit the references to f and T whenever possible; i.e., we use the notations $d(i,j)$ and $\mathcal{D}$ instead of $d_f(i,j)$ and $\mathcal{D}_f(T)$, respectively, and similarly for the functions defined in the sequel.

A *center* of a labeled tree is a vertex i for which $\max_{j\neq i}\{d(i,j)\}$ is minimal, and this value is called the *radius* $\mathcal{R}$ of the tree; i.e.,

$$\mathcal{R} = \min_i\{\max_j \{d(i,j)\}\}.$$

The terms diameter, radius and center are similarly defined for labeled undirected graphs. In a rooted tree the center is located at the root.

The *maximal weight* and *minimal weight* on $p(i,j)$ are

$$\max(i,j) = \max_{e\in p(i,j)} \{f(e)\} \quad \text{and} \quad \min(i,j) = \min_{e\in p(i,j)} \{f(e)\},$$

respectively.

The number of edges in $p(i,j)$ is $|p(i,j)|$, and

$$\operatorname{avr}(i,j) = \frac{d(i,j)}{|p(i,j)|}$$

is the *average weight* on $p(i,j)$.

1.2. The objective functions

The quantities which we optimize, for a given tree T, over all possible labelings f, are the following:

(1) $\mathcal{D}$,
(2) $\mathcal{R}$,
(3) $\text{SUMDIS}(T) = \sum_{i<j} d(i,j)$,
(4) $\text{SUMMAX}(T) = \sum_{i<j} \max(i,j)$,
(5) $\text{SUMMIN}(T) = \sum_{i<j} \min(i,j)$, and
(6) $\text{SUMAVR}(T) = \sum_{i<j} \operatorname{avr}(i,j)$.

The functions (3)–(6) are average measurements, since the summations are over all pairs of vertices.

1.3. Motivation

This work is applicable in the design of communication networks. Given communication lines of different properties, such as communication cost, capacity, vulnerability and reliability, we want to assign these communication lines to the direct connections ('edges') of a given communication network (of a tree structure), such that certain objective cost functions will be optimized.

For example, the function SUMDIS is measuring the average communication cost of the network (see [4] and [8]). The diameter is measuring the maximal communication cost in the network, and the radius is measuring the maximal communication cost from a directory optimally located in the network. Let $c(e)$ and $v(e)$ be the capacity and vulnerability of a communication line assigned to the edge e, respectively. Define the capacity and vulnerability between the vertices i and j as

$$c(i,j) = \min_{e \in p(i,j)} \{c(e)\} \quad \text{and} \quad v(i,j) = \max_{e \in p(i,j)} \{v(e)\},$$

respectively. Then the functions SUMMIN and SUMMAX measure the average capacity and vulnerability of the network, respectively.

1.4. In Section 2 we present several NP-complete problems, concerning optimizing the radius and the diameter of a tree and of a binary weighted (0, 1 weights) general graph. Polynomial-time algorithms for special cases of minimizing the radius of a tree are shown in Section 3. In Section 4 we present polynomial-time algorithms for optimizing average measurement functions of the tree, such as SUMDIS and SUMMAX. Open problems are found in Section 5.

2. Radius and diameter: NP-complete results

2.1. In this section the following problems are shown to be NP-complete (see [6]):

Problem 1. Given a tree T with n edges, a set W of n non-negative weights, and an integer $k > 0$, to determine whether there exists a labeling f with diameter $\leqslant k$.

Problem 2. Like PROBLEM 1, for radius $\leqslant k$.

Note. (1) This result holds also for rooted trees.

(2) PROBLEMS 1 and 2 remain NP-complete when the weights are integral and bounded by a polynomial in n.

Problem 3. Like PROBLEM 1, for radius $\geqslant k$.

Note. (1) Maximizing the diameter of a tree is trivial.

(2) PROBLEMS 1, 2 and 3 are NP-complete also for the corresponding vertex labeling problems, e.g.,

Problem 1′. Given a tree T with $n+1$ vertices, a set W of $n+1$ non-negative weights and an integer $k > 0$, to determine whether there exists a labeling of the vertices of T with the weights of W such that the sum of the weights of the vertices along any path in the tree is not greater than k.

Problem 4. Given a connected graph G with m edges and a set of 0, 1 weights W, $|W| = m$, to determine whether there exists a labeling with radius ≤ 1.

Problem 5. Given a connected graph G with $n+1$ vertices and a set of 0, 1 weights W, $|W| = n$, to determine whether G contains a spanning tree which can be labeled such that its diameter ≤ 2.

2.2. The reductions are from the PARTITION problem and the MAXIMUM TERMINAL SPANNING TREE problem, both known to be NP-complete (see [6] and [2]):

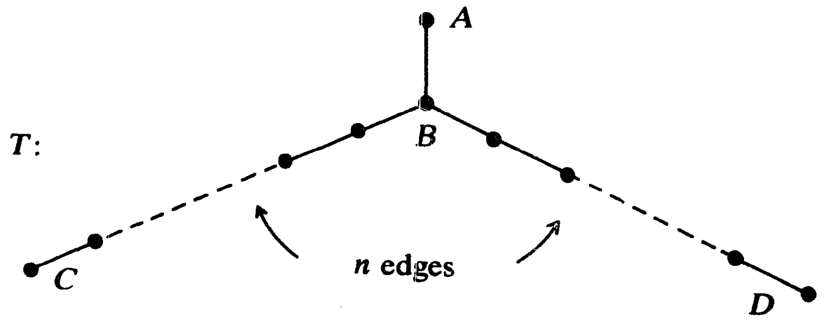
Partition. Given positive integers a_i, $i = 1, 2, \ldots, n$, to determine whether there exists a set $I \subseteq \{1, 2, \ldots, n\}$ such that

$$\sum_{i\in I} a_i = \sum_{i\notin I} a_i.$$

Maximum Terminal Spanning Tree. Given a graph G and an integer $k > 0$, to determine whether G has a spanning tree with at least k terminals (=leaves).

2.3. Proof for PROBLEM *1*

We show that PARTITION is reduced to PROBLEM 1. Given a_i, $i = 1, 2, \ldots, n$, (an instance of PARTITION), we define the following instance of PROBLEM 1:



$$W = \{a_1, a_2, \ldots, a_n, \underbrace{0, 0, \ldots, 0}_{n\,0\text{'s}}, \sum_i a_i\}.$$

$$k = \tfrac{3}{2} \sum_i a_i.$$

We prove that there exists a solution to the PARTITION problem iff there exists a labeling of T with diameter $\leq k$.

Suppose there exists a solution to the PARTITION problem, namely $\sum_{i\in I} a_i = \sum_{i\notin I} a_i$ for some $I \subseteq \{1, 2, \ldots, n\}$. The following labeling has a diameter k: label AB with $\sum_i a_i$, spread the a_i's for $i \in I$ and $i \notin I$ on BC and BD, respectively, and label the rest of the edges with 0's.

Suppose there exists a labeling with diameter $\leq k$. One can find such a labeling in which $\sum a_i$ labels the edge AB (if in the given labeling AB is labeled with a_t, for some t, and $\sum a_i$ labels another edge e, then by interchanging $\sum a_i$ and a_t the diameter is not increased). This means that there exists a labeling such that the a_i's on BC and on BD sum up to $\frac{1}{2}\sum a_i$ each, hence we have a solution to the given PARTITION instance.

2.4. *Proof for* PROBLEM *2*

We prove that PARTITION is reduced to PROBLEM 2. Given a_i (as above), we define the following instance of PROBLEM 2:

$$T = P_{2n+1}, \qquad W = \{a_1, a_2, \ldots, a_n, \underbrace{0, 0, \ldots, 0}_{n\,0\text{'s}}\} \quad \text{and} \quad k = \tfrac{1}{2}\sum a_i.$$

The rest follows immediately.

Note. The same reduction holds in the case when this path is rooted in its center, which proves that PROBLEM 2 is NP-complete also for rooted trees.

Note. As was pointed out by Johnson [5], PROBLEM 1 and PROBLEM 2 remain NP-complete ('in the strong sense' [2]) when the weights are integral and bounded by a polynomial in n. This is shown by similar reductions from the 3-PARTITION problem (see [2]) of $3m$ integers to corresponding instances of PROBLEM 1 and PROBLEM 2, where the labeled tree consists of m paths of three edges each, all emanating from a common vertex (which is the root in the rooted tree case of PROBLEM 2).

2.5. *Proof for* PROBLEM *3*

We prove that PARTITION is reduced to PROBLEM 3. Given a_i (as above), we define the following instance of PROBLEM 3:

$$T = P_{n+2},$$

$$W = \{a_1, a_2, \ldots, a_n, x\} \quad \text{where } x \geq \max_i \{a_i\}$$

and

$$k = x + \tfrac{1}{2}\sum a_i.$$

If there exists a solution to PARTITION, then the labeling shown in Fig. 1 has a radius $\mathcal{R} = k$.

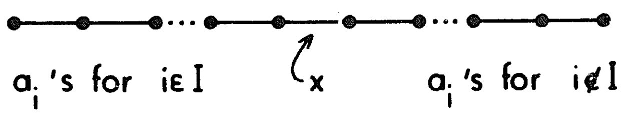


Fig. 1.

If there exists a labeling with radius $\mathcal{R} \geq x + \frac{1}{2}\sum a_i$, then let D be a center and DB a radius (see Fig. 2). Then

$$d(D,B) \geq x + \tfrac{1}{2}\sum a_i, \quad \text{and hence} \quad d(A,D) \leq \tfrac{1}{2}\sum a_i.$$

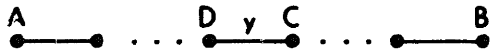


Fig. 2.

Also $d(D,B) \leq d(A,C)$, otherwise D is not a center. Therefore

$$x + \tfrac{1}{2}\sum a_i \leq d(D,B) \leq d(A,C) \leq y + d(A,D) \leq y + \tfrac{1}{2}\sum a_i,$$

hence $x \leq y$. But $y \leq x$, hence $y = x$, and

$$d(A,D) = d(B,C) = \tfrac{1}{2}\sum a_i,$$

which solves the PARTITION problem.

2.6. *Note*

PROBLEM 1, PROBLEM 2 and PROBLEM 3 are also NP-complete for vertex labeling. The reductions are from the corresponding edge labeling problems. Adding a weight 0 to the given weights, a solution for the vertex labeling problem induces a solution for the corresponding edge labeling problem. This is done by regarding the vertex labeled 0 as a root of the tree and using a label of any other vertex to label the edge directed to this vertex.

2.7. *Proof for* PROBLEM 4

We prove that MAXIMUM TERMINAL SPANNING TREE is reduced to PROBLEM 4. Given a graph G with $n+1$ vertices and $k>0$ (an instance of MAXIMUM TERMINAL SPANNING TREE), we define the following instance of PROBLEM 4: G is the same given graph and W contains $n-k$ 0's and all the rest 1's. We prove that G has a spanning tree with at least k leaves iff G has a labeling with radius $\mathcal{R} \leq 1$.

Suppose G has a spanning tree with at least k leaves. This yields a labeling of G with radius $\mathcal{R} \leq 1$, by labeling the internal edges (of the tree) with 0's, and thus every internal vertex (of the tree) is a center for the graph G with radius $\mathcal{R} \leq 1$.

Suppose there exists a labeling of G with radius $\mathcal{R} \leq 1$. Let A be a center of the labeled graph, and apply a shortest-path algorithm from A to get a 'shortest-path spanning tree' with radius $\mathcal{R} \leq 1$. The tree is labeled with at least k 1's. In this tree there exists at most one edge labeled 1 on each path from A to any leaf, since the radius is ≤ 1. Therefore the number of leaves in the spanning tree is at least as the number of 1's in the tree, i.e. $\geq k$.

2.8. *Proof for* PROBLEM 5

The reduction is similar to that of PROBLEM 4.

3. Radius: polynomial results

3.1. In the previous section we proved that minimizing the diameter or the radius in a tree are NP-complete problems. Here we present polynomial-time algorithms for certain instances of these problems.

3.2. **Problem 6.** Given a tree T and a set of 0, 1 weights W, to find a labeling with minimal radius.

The following algorithm solves this problem in $O(n)$ running-time:

Algorithm A

```
1. while (number of leaves in $T$) $\leq$ (number of 1's in $W$)
     do begin
        label all the terminal edges with 1;
        delete the terminal edges from $T$;
        delete the used 1's from $W$;
        end;
2. Put all the remaining 1's on terminal edges;
   label all other edges with 0's.
```

Proof. Left to the reader.

Note. Algorithm A is applicable also for rooted trees.

3.3. **Problem 7.** Like Problem 6, for a set of a, b weights, $a < b$.

The following algorithm solves this problem in $O(n \log n)$ running-time in the case when T is a rooted tree. An $O(n^2 \log n)$ algorithm for unrooted trees is obtained by applying this algorithm from every vertex in the tree.

Algorithm B

```
1. Label all edges of $T$ with $a$'s;
   insert all terminal edges into a 'candidate list';
2. while $W$ contains at least one $b$
     do begin
        find an edge $(x, y)$ in the 'candidate list', which lies on a path of minimal
          length from the root to a leaf;
        label $(x, y)$ with $b$;
        delete this $b$ from $W$;
        delete $(x, y)$ from the 'candidate list';
```

```
if $x$ has no sons labeled with $a$
  then begin
       find the unique $w$ such that $(w, x) \in T$;
       insert $(w, x)$ into the 'candidate list';
       end;
end;
```

Proof. Let T be the tree labeled by the algorithm while T' is an optimally labeled tree minimizing the radius and satisfying the condition that if a (directed) edge (x, y) is labeled with b, then all edges in the subtree T'_y rooted at y are labeled with b (all the b-weights are pushed towards the leaves). Note that T satisfies this condition by the algorithm.

First we show that there exists such an optimal labeled tree T'. Let T'' be an optimal labeled tree not satisfying this property. Denote by d, d' and d'' distances in T, T' and T'', respectively (note that all these three trees differ only by their labelings). There exists an edge (x, y) labeled b in T'' while some edge (w, z) in T_y is labeled with a. Exchanging the weights of the edges (x, y) and (w, z) yields a labeling where $d''(r, u)$ is decreased by $b-a$ for every $u \in T_y - T_z$ and is not changed for any other vertex in the tree. Repeating this process yields a labeled tree T', satisfying the desired property, without increasing the radius, and therefore T' is also optimal.

Assume now that the tree T labeled by the algorithm is not optimal, i.e., there exists a vertex $x \in T$ such that $d(r, x) > \mathcal{R}$, where R is the radius of T'. Then the path from r to x contains an edge labeled with b in T and labeled with a in T'. Let (y, z) be the first such edge from the root r. On the other hand there exists an edge (u, v) labeled with b in T' and labeled with a in T. By the property of T' all the edges in the subtree T_v are labeled with b. Let w be a vertex in T_v with maximal distance $d'(r, w)$. $d'(r, w) \leq \mathcal{R}$ since $\mathcal{R}$ is the radius of T' (see Fig. 3).

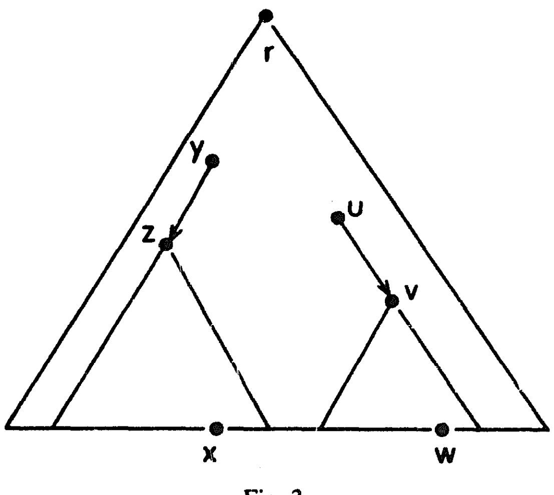


Fig. 3.

Note that the edge (u, v) is labeled with a in T and with b in T', and thus

$$d(r, w)+(b-a)\leq d'(r, w)\leq R.$$

Hence

$$d(r, w)\leq R-(b-a).$$

By our assumption $d(r, x)>R$. Thus before the algorithm labeled the edge (y, z) with b we had

$$d(r, x)+(b-a)>R \quad \text{or} \quad d(r, x)>R-(b-a)$$

and therefore

$$d(r, w)<d(r, x).$$

Hence the algorithm should have labeled the edge (u, v) with b before labeling the edge (y, z), a contradiction.

Therefore the labeling of T is optimal.

3.4. **Problem 8.** Like PROBLEM 7, for a maximal radius.

In a rooted tree this is done by finding a longest path from the root, labeling its edges with as much b's as possible, and labeling the rest of the edges arbitrarily. In an unrooted tree this is done by finding a longest path, labeling it with $\lceil x/2\rceil$ b's on one end and $\lfloor x/2\rfloor$ b's on the other (where x is the number of b's), and labeling the rest of the edges arbitrarily. Both algorithms are of $O(n)$ running-time.

3.5. **Problem 9.** Given a tree T and a set of 0, 1 weights W, to find a labeling with minimal diameter.

This is done in $O(n)$ running-time algorithm, identical to that of PROBLEM 6.

4. Average measurements: polynomial results

4.1. In this section we present polynomial-time algorithms for optimization problems concerning the functions SUMDIS, SUMAVR, SUMMIN and SUMMAX.

4.2. **Problem 10.** Given T and W, to find a labeling that maximizes or minimizes

$$\text{SUMDIS}(T)=\sum_{i<j} d(i, j).$$

Following Hu [4] we observe that instead of summing the $d(i, j)$'s we can count the number of occurrences of each edge in the summation. Suppose that an edge e partitions the tree T into two subtrees with k and $n+1-k$ vertices, respectively.

Then e is counted $k(n+1-k)$ times in SUMDIS(T). We call this value $c(e)=k(n+1-k)$ the *connection* of e. Now

$$\mathrm{SUMDIS}(T)=\sum_{e} c(e)\cdot f(e)$$

and the next theorem follows:

Theorem. *Given the connections of the edges of the tree T, $c(e_1)\leqslant c(e_2)\leqslant\cdots\leqslant c(e_n)$, the labeling of e_i with $w_i(w_{n+1-i})$, for $i=1,2,\ldots,n$, maximizes (minimizes) the function* SUMDIS(T).

The optimal labeling is unique up to permuting edges with equal connections. Assuming that all weights are distinct, the number N of optimal labelings satisfies

$$2^{\lfloor n/2\rfloor}\leqslant N\leqslant n!.$$

The bounds are achieved for the graphs P_{n+1} and $K_{1,n}$.

The connections can be calculated in O(n) time by regarding T as rooted and traversing it in end-order (see [7]). Therefore, and because of the sorting step involved, the algorithm for optimal labeling (for either maximizing or minimizing SUMDIS(T)) has O(n log n) running-time.

4.3. **Problem 11.** Given T and W, to find a labeling that maximizes or minimizes

$$\mathrm{SUMAVR}(T)=\sum_{i<j}\frac{d(i,j)}{|p(i,j)|}.$$

The solution is essentially similar to the solution of PROBLEM 10. The only change is that the connections $c(e)$ are replaced by the *average connection* $\tilde{c}(e)$ defined by

$$\tilde{c}(e)=\sum_{i}\frac{c_j(e)}{j},$$

where $c_j(e)$ is the number of paths of length j through the edge e.

Calculating the average connections is done by a breadth-first search (see [1]) from every edge e, getting the number of paths of length i on each side of e (for every i), and then 'merging' the information from both sides of e by convolution double in O(n log n) time (see [1]). Therefore the algorithm in this case is of O(n^2 log n) time.

4.4. **Problem 12.** Given T and W, to find a labeling that minimizes

$$\mathrm{SUMMAX}(T)=\sum_{i<j}\max(i,j).$$

The problem of maximizing SUMMAX(T) is still open. Similarly, our algorithm for solving PROBLEM 12 solves also the problem of maximizing the function SUMMIN(T), while the problem of minimizing SUMMIN(T) is still open.

Conjecture. The problem of maximizing (minimizing) SUMMAX(T) SUMMIN(T)) is NP-hard.

The algorithm to solve PROBLEM 12 is based on the following theorem:

Theorem. *Given T and W as above, and assuming $w_i < w_{i+1}$ for every i, we have:*

(1) *In any labeling that minimizes* SUMMAX(T) *the largest weight w_n labels a terminal edge.*

(2) *For any terminal edge there exists an optimal labeling in which w_n labels this edge.*

(3) $\min_f\{\text{SUMMAX}_f(T)\} = \sum_i i \cdot w_i$ *(which means that this optimal value is independent on the tree T).*

Note. The result holds also for the case $0 \leqslant w_1 \leqslant \cdots \leqslant w_n$, and the modification of the proof is left to the reader.

Proof. We prove the theorem by induction on n. The theorem holds for $n = 1, 2$ and 3. Assume it holds for a tree with less than n edges. Let T be a tree with n edges. Assume f is a labeling that minimizes SUMMAX(T), and that w_n labels a non-terminal edge e, thus partitioning T into two subtrees T_1 and T_2 containing k and $n+1-k$ vertices, respectively, for some $2 \leqslant k \leqslant n-1$. Wlog we assume that w_{n-1} labels an edge in T_1. Applying the inductive hypothesis to T_1 we know that there exists an optimal labeling for T_1 in which a specified terminal edge is labeled with w_{n-1}. We choose this edge to be such that by removing it with its corresponding leaf we won't disconnect T (see Fig. 4). Note that all the paths from vertices in T_1, containing w_n, are not effected by this new labeling of T_1 (since on them w_n is maximal), and, as for T_1, the value of SUMMAX is not changed by the inductive hypothesis. In the current labeling the contribution of w_n and w_{n-1} to SUMMAX(T) is

$$k(n+1-k)w_n + (k-1)w_{n-1}. \tag{1}$$

By interchanging w_{n-1} and w_n we get a labeling f' in which their contribution is

$$nw_n + (k-1)(n+1-k)w_{n-1}. \tag{2}$$

But subtracting (1) from (2) gives

$$(2)-(1) = (k-1)(n-k)(w_{n-1} - w_n) < 0$$

because $1 < k < n$ and $w_{n-1} < w_n$, a contradiction.

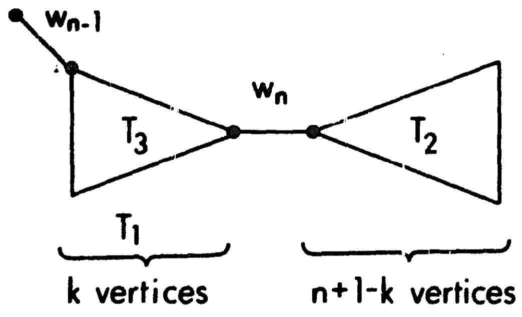


Fig. 4.

We know that in an optimal labeling of T w_n labels a terminal edge. It is clear that the contribution of w_n to SUMMAX(T) is

$$nw_n. \tag{3}$$

It is also clear that we can label any terminal edge with w_n, and, after erasing it together with the corresponding leaf, we apply the inductive hypothesis to the remaining tree, getting optimal labeling with a value of

$$\sum_{i=1}^{n-1} iw_i$$

hence for T we get the optimal value of $\sum_{i=1}^{n} iw_i$.

Following the above proof, it is clear how to get an O(n) labeling algorithm that minimizes SUMMAX(T).

4.5. As for the number of optimal labelings, it can be shown that, when all the weights are distinct, the number N of such labelings satisfies

$$2^{n-1} \leqslant N \leqslant n!,$$

when the bounds are achieved for the graphs P_{n+1} and $K_{1,n}$.

4.6. **Problem 13.** Given T and W, to find a labeling that maximizes SUMMAX(T).

We stated our conjecture (above) that this problem is NP-hard. As a matter of fact, we don't even know how to solve this problem in the case when the tree T is a path (P_{n+1}). Note that the 'natural' approach of labeling the edge with maximal connection with the largest weight w_n is not optimal; for example, consider the case when $T = P_6$ and $W = \{1, 2, 3, 8, 9\}$.

4.7. We consider now the case when T is a path and W contains a bounded number of distinct weights. We present an exact description of the solution when W contains only two kinds of weights, and give an optimal labeling algorithm for the case when W contains a bounded number of distinct weights.

4.8. A path, 2 weights

Here $T = P_{n+1}$, and W contains k weights b and $n-k$ weights a, $a < b$. The structure of the solution is characterized in the following theorem:

Theorem. *Let T and W be as above, and denote a labeling f as a sequence*

$$a^{x_0}ba^{x_1}b \cdots ba^{x_{t-1}}ba^{x_t} \tag{4}$$

where $x_i \geqslant 0$ and $\Sigma\, x_i = n-k$. If the labeling f is optimal, then

$$|x_i - x_j| \leqslant 1 \quad \text{for every } i \text{ and } j.$$

Proof. For a labeling function f as described in (4), let $N_f(a)$ ($N_f(b)$) denote the number of paths $p(i,j)$ in which the largest label is a (b). It is clear that by minimizing $N_f(a)$ over all possible labelings f we will also maximize SUMMAX(T) over all possible labelings f, since $a < b$ and

$$N_f(a) + N_f(b) = \binom{n+1}{2}.$$

It is clear that

$$N_f(a) = \binom{x_0+1}{2} + \binom{x_1+1}{2} + \cdots + \binom{x_t+1}{2}. \tag{5}$$

Assume that f is an optimal labeling and that the theorem doesn't hold. Wlog we may assume that

$$x_0 > x_1 + 1. \tag{6}$$

We define a labeling f' as follows:

$$f'\colon\ a^{x_0-1}ba^{x_1+1}ba^{x_2}b \cdots ba^{x_{t-1}}ba^{x_t}. \tag{7}$$

By (5) we have

$$N_{f'}(a) = \binom{x_0}{2} + \binom{x_1+2}{2} + \cdots + \binom{x_t+1}{2}. \tag{8}$$

Using (5), (8) and (6) it can be shown that

$$N_{f'}(a) < N_f(a),$$

hence $\text{SUMMAX}_{f'}(T) > \text{SUMMAX}_f(T)$, a contradiction.

Using this theorem it is clear how one should 'spread' the a's among the b's in order to get a labeling that maximizes the function SUMMAX.

4.9. *A path, k weights*

Here $T = P_{n+1}$ and W contains m_i copies of w_i for $i = 1, 2, \dots, k$, where $m_1 + m_2 + \cdots + m_k = n$ and $w_1 > w_2 > \cdots > w_k \geq 0$. The following dynamic programming algorithm, proposed by Megiddo [9], solves the problem in time $O(n^{2k+1})$. Let $s = (s_1, s_2, \dots, s_k)$ and $p = (p_1, p_2, \dots, p_k)$ be integral vectors where $0 \leq s_i \leq m_i$, $0 \leq p_i \leq n$ for $i = 1, 2, \dots, k$. Consider the following problem: Given a path of length $m = \sum s_i$ (denote its vertices by $0, 1, \dots, m$), label the edges with s_i copies of w_i ($i = 1, 2, \dots, k$), maximizing SUMMAX, where p_i is the lowest occurrence of w_i (if $s_i = 0$, then we assume $p_i = 0$). (There may be no feasible solution.) Let $\phi(s, p)$ be this maximal value (if not feasible let $\phi(s, p) = 0$). For a feasible solution (s, p) let $\psi(s, p)$ denote the contribution of the vertex 0 to SUMMAX. This contribution, $\sum_{j=1}^{m} \max(0, j)$, depends only on s and p (and not on the particular labeling), and can be computed in $O(m)$ time. Now, $\phi(s, p)$ can be computed recursively, as follows: Given a feasible (s, p), there is a unique t, $1 \leq t \leq k$, such that $s_t > 0$ and $p_t = 1$. Then

$$\phi(s, p) = \psi(s, p) + \max_{i \leq i \leq m-1} \phi(s_1, \dots, s_{t-1}, s_t - 1, s_{t+1}, \dots, s_k, p_1, \dots, p_{t-1}, i, p_{t+1}, \dots, p_k).$$

Thus, the function $\phi(s, p)$ is evaluated at no more than n^{2k} points, and each evaluation does not require more than $O(n)$ time, hence the algorithm runs in time $O(n^{2k+1})$. This bound is a non-trivial one, since a complete enumeration requires $O(k^n)$ time.

5. Open problems

The first two open problems are related to PROBLEM 5:

Problem A. Given a connected graph G with m edges and a set W of 0, 1 weights, $|W| = m$, to determine whether there exists a labeling with diameter ≤ 2.

This problem seems to be NP-complete. However, it is not clear that it can be reduced from the MAXIMUM TERMINAL SPANNING TREE problem; new ideas are needed. Furthermore, we conjecture that the following problem is also NP-complete:

Problem B. Given a connected graph G with m edges and a set W of 0, 1 weights, $|W| = m$, to determine whether there exists a labeling with diameter ≤ 1.

We have the following observations about this problem. Suppose the graph G is labeled with the given weights such that its diameter $\mathcal{D} \leq 1$. It is clear that $\mathcal{D} = 0$

iff G contains a spanning tree with all edges labeled 0. Thus the interesting case is when $\mathscr{D} = 1$. In this case we may assume that the edges labeled with 0 generate a forest in G, since if they generate any cycle we can replace one 0-edge with a 1-edge without increasing the diameter (putting this 0-edge in another arbitrary place not closing a cycle of 0's). Let $T_i = (V_i, E_i)$, $i = 1, 2, \ldots, j$ be the trees in this forest. A graph $G' = (V', E')$ is constructed from G by contracting (see [3]) all the vertices of each tree T_i, $i = 1, 2, \ldots, j$, into one vertex u_i. It can be shown that G is labeled with a diameter $\mathscr{D} = 1$ iff G' is a complete graph. This observation leads us to the above conjecture.

It can also be observed that this conjecture is equivalent to the following contractability problem:

Problem C. Given a graph G and an integer $t > 0$, to determine whether one can obtain the complete graph K_t from G by a series of edge contractions, i.e., a series in which each step replaces two adjacent vertices u and v by a single vertex w adjacent to exactly those vertices that were previously adjacent to at least one of u and v.

The next two problems are related to the conjecture of Section 4 that the problem of maximizing SUMMAX(T) is NP-hard.

Problem D (see 4.4 and 4.6). Given a tree $T = P_{n+1}$ and W, to find a labeling that maximizes SUMMAX(T).

Problem E. Given a tree T and a set W of a, b weights, to find a labeling which maximizes SUMMAX(T) (this is an interesting special case of the open problem of maximizing SUMMAX(T) for general T and W).

Acknowledgment

We would like to thank C. L. Liu for the inspiring conversations we had while working on this paper, and the referees for their suggestions.

References

[1] A.V. Aho, J.E. Hopcroft and J.D. Ullman, *The Design and Analysis of Computer Algorithms* (Addison-Wesley, Reading, MA, 1974).
[2] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness* (Freeman, San Francisco, 1979).
[3] F. Harary, *Graph Theory* (Addison-Wesley, Reading, MA, 1969).
[4] T. C. Hu, Optimal communication spanning trees, *SIAM J. Comput.* **3** (3) (1974) 188–195.
[5] D.S. Johnson, Private communication.

[6] R.M. Karp, Reducibility among combinatorial problems, in: R.E. Miller and J.W. Thatcher, Eds., *Complexity of Computer Computations* (Plenum Press, New York, 1972) 85–104.

[7] D.E. Knuth, *The Art of Computer Programming, Vol. 1: Fundamental Algorithms* (Addison-Wesley, Reading, MA, 1968).

[8] J.K. Lenstra, A.H.G. Rinnooy Kan and D.S. Johnson, The complexity of the network design problem, Mathematisch Centrum, Amsterdam (1976).

[9] N. Megiddo, Private communication.