

# Upgrading Trees Under Diameter and Budget Constraints

**Victor Chepoi**

Laboratoire d'Informatique Fondamentale, Université de la Méditerranée, Faculté des Sciences de Luminy, F-13288 Marseille Cedex 9, France

**Hartmut Noltemeier**

Lehrstuhl für Informatik I, Universität Würzburg Am Hubland, D-97074 Würzburg, Germany

**Yann Vaxès**

Laboratoire d'Informatique Fondamentale, Université de la Méditerranée, Faculté des Sciences de Luminy, F-13288 Marseille Cedex 9, France

Given a tree  $T = (V, E)$  endowed with a length function  $l$  and a cost function  $c$ , the *diameter lowering problem* consists in finding the reals  $0 \leq x(e) \leq l(e)$ ,  $e \in E$  such that the tree obtained from  $T$  by decreasing the length of every edge  $e$  by  $x(e)$  units has a minimal diameter subject to the constraint  $\sum_{e \in E} c(e)x(e) \leq B$ , where  $B$  is the available budget (analogously, one can minimize the cost of lowering subject to a diameter constraint). We present an  $O(|V|^2)$  algorithm for solving this problem by developing and using algorithms of similar complexity for related eccentricity lowering problems. © 2002 Wiley Periodicals, Inc.

**Keywords:** network design; network upgrade; diameter; trees; cuts; facility location problems

## 1. DIAMETER AND ECCENTRICITY LOWERING PROBLEMS

The paper addresses the *diameter lowering problem*, where we are given a tree  $T = (V, E)$ , positive lengths  $l(e)$ , and costs  $c(e)$  for all edges  $e \in E$  and a nonnegative parameter  $D$ . Let the diameter  $D_l := D_l(T)$  of  $T$  be the length of a longest path in  $T$ . The objective is to shrink edge lengths in a cost-efficient way, such that the diameter under the shrunken length is at most  $D$ . More formally, we need

to find  $0 \leq x(e) \leq l(e)$ ,  $e \in E$ , such that  $\sum_{e \in E} c(e)x(e)$  is minimized and  $D_{l-x}$  is at most  $D$ . Another version of the problem is to maximize the diameter reduction under a given cost constraint. We also consider the *eccentricity lowering problem* in which, given a vertex  $s \in T$  and a nonnegative parameter  $R$ , we need to shrink edge lengths in a cost-efficient way, such that the eccentricity of  $s$  under the resulting length is at most  $R$  (the *eccentricity* of a vertex is the largest distance from this vertex to a leaf of  $T$ ).

The solution of the diameter lowering problem can be used to reduce the maximum communication delay in communication networks to guarantee certain access or query times (gossiping problems). The eccentricity lowering problem can find similar applications in broadcasting problems but also in a variety of real-world-situations, where we have to invest a budget to shorten the longest paths in a shortest path tree to or from a facility (school, hospital, firestation, supermarket, etc.) located in a given region. The diameter lowering problem subject to 0/1 reductions (an edge is either contracted to a given minimal length or not at all) was considered in [1], where it was shown to be NP-hard even for trees. The paper [1] also provided bicriteria approximation algorithms for this and some other related problems. Notice that a slight modification of the proof of Proposition 10 of [1] shows that, in general networks, diameter and eccentricity lowering problems considered in our paper are NP-hard and are at least as hard to approximate as SET COVER.

In this paper, we present strongly polynomial algorithms for diameter and eccentricity lowering problems on trees. We consider two closely related diameter lowering problems:

---

Received December 1, 2000; accepted September 1, 2002

Correspondence to: Y. Vaxès

Contract grant sponsor: Bayerisch-Französisches Hochschulzentrum/Centre de Coopération Universitaire Franco-Bavariens

Contract grant sponsor: France Telecom R&D; contract grant number: 0018090

© 2002 Wiley Periodicals, Inc.

PROBLEM DLP( $T, l, c, D$ ): Given  $0 \leq D \leq D_l$ , find the reals  $0 \leq x(e) \leq l(e)$ ,  $e \in E$ , such that the tree obtained from  $T$  by decreasing the length of every edge  $e \in E$  by  $x(e)$  units has diameter at most  $D$  and the cost  $\sum_{e \in E} c(e) \cdot x(e)$  of this lowering is minimized.

PROBLEM DLP\*( $T, l, c, B$ ): Given a budget  $B \geq 0$ , find the reals  $0 \leq x(e) \leq l(e)$ ,  $e \in E$ , such that  $\sum_{e \in E} c(e) \cdot x(e) \leq B$  and the diameter  $D_{l-x}(T)$  of the tree obtained from  $T$  by decreasing the length of every edge  $e \in E$  by  $x(e)$  units is minimized.

Both problems can be formulated as linear programs; therefore, they can be solved in polynomial time using the ellipsoid method:

Problem DLP( $T, l, c, D$ )

$$\begin{aligned} \text{Minimize} \quad & \sum_{e \in E} c(e) \cdot x(e) \\ \text{subject to} \quad & \begin{cases} \sum_{e \in P} [l(e) - x(e)] \leq D & \text{for } P \in \mathcal{P} \\ 0 \leq x(e) \leq l(e), e \in E, \end{cases} \end{aligned}$$

and

Problem DLP\*( $T, l, c, B$ )

$$\begin{aligned} \text{Minimize} \quad & D \\ \text{subject to} \quad & \begin{cases} \sum_{e \in P} [l(e) - x(e)] \leq D & \text{for } P \in \mathcal{P} \\ \sum_{e \in E} c(e) \cdot x(e) \leq B \\ 0 \leq x(e) \leq l(e), e \in E, \end{cases} \end{aligned}$$

where  $\mathcal{P}$  is the set of all paths between the leaves of  $T$ . By  $P(v, w)$ , we denote the path of  $T$  between two vertices  $v$  and  $w$ .

The strongly polynomial algorithm for solving these problems resides on related eccentricity lowering problems. To formulate them, first we introduce the tree networks as they are defined in location theory [2] and a few other concepts. For a tree  $T = (V, E)$  endowed with a length function  $l$ , consider each edge  $e$  of  $T$  as a contiguous segment of length  $l(e)$ . Denote by  $\mathcal{T}$  the union of these segments in the assumption that they pairwise intersect only in common end-vertices. Call  $\mathcal{T}$  a *tree network* with support  $T$ . Then,  $v \in V$  are the *vertices* of  $\mathcal{T}$  and  $p \in \mathcal{T} - V$  are the (*inner*) *points* of  $\mathcal{T}$ . The length function  $l$  extends in a canonical way to a distance function  $d_l : \mathcal{T} \times \mathcal{T} \rightarrow \mathbb{R}^+ \cup \{0\}$ , where  $d_l(p, q)$  is the sum of lengths of the segments constituting the unique path of  $\mathcal{T}$  between the points  $p$  and  $q$ . The *eccentricity*  $R_l(s)$  of a point  $s \in \mathcal{T}$  is

$$\begin{aligned} R_l(s) &:= \sup\{d_l(s, p) : p \in \mathcal{T}\} \\ &= \max\{d_l(s, q) : q \text{ is a leaf of } \mathcal{T}\}. \end{aligned}$$

Denote by  $F_l(s)$  the set of leaves which are furthest from  $s$ , that is:

$$F_l(s) := \{q \in V : d_l(q, s) = R_l(s)\}.$$

The *radius*  $R_l = \min_{s \in \mathcal{T}} R_l(s)$  is the minimum eccentricity of a point in  $\mathcal{T}$ , while the *diameter*  $D_l$  is the maximum eccentricity. It is well known that  $D_l = 2R_l$  always holds for tree networks and that there exists exactly one point  $s^*$  (the *absolute center*) of  $\mathcal{T}$  whose eccentricity is  $R_l$  (cf. [2]). The point  $s^*$  lies in the middle of each diametral path of  $\mathcal{T}$ .

These observations indicate that, in place of solving the diameter lowering problems, one can solve the analogous radius or, more generally, eccentricity lowering problems. Similarly to the DLP formulations, the problem ELP( $T, l, c, s, R$ ) is defined as follows: Given a point  $s \in \mathcal{T}$  and a positive number  $R < R_l(s)$ , we are searching for a minimum-cost lowering of the edges of  $T$  such that in the updated tree network the point  $s$  has eccentricity at most  $R$ . Finally, in the problem ELP\*( $T, l, c, s, B$ ), we are searching for a lowering  $x$  of the tree  $T$  whose total cost does not exceed an available budget  $B$  and such that the eccentricity  $R_{l-x}(s)$  of a given point  $s$  in the updated tree network is as small as possible. The ELP problems can be written as linear programs by replacing in the respective DLP formulations  $D$  by  $R$  and  $\mathcal{P}$  by the set  $\mathcal{P}_s$  of all paths between  $s$  and the leaves of  $T$ . A feasible solution of the problem ELP that reduces the eccentricity of  $s$  by  $\beta := R_l(s) - R$  units will be also called a  $\beta$ -*contraction*.

For a point  $s \in \mathcal{T}$ , let  $\text{Rad}_B(s)$  be the least eccentricity of  $s$  which can be achieved under a fixed budget  $B$  and let  $\text{Bud}_R(s)$  be the minimum cost needed to lower the eccentricity of  $s$  to  $R$ . In the sequel, we consider  $\text{Rad}_B$  and  $\text{Bud}_R$  as functions of variable  $s \in \mathcal{T}$ .

If for a fixed budget  $B$ , among all points of  $\mathcal{T}$ , we could select a point  $s^*$  for which  $\text{Rad}_B(s^*)$  is minimum; then,  $2\text{Rad}_B(s^*)$  is the least diameter that one can achieve under budget  $B$  (a similar reasoning applies to  $\text{DLP}(T, l, c, R)$  and the function  $\text{Bud}_R(s)$ ). Therefore, to solve the DLP problems efficiently, (i) we must be able to solve the ELP problems in strongly polynomial time and (ii) to discretize the DLP problems by inferring a compact list (in the size of  $T$ ) of ELP problems which have to be solved in order to find the point  $s^*$ . Thus, the paper is organized as follows. In Section 2, we describe an algorithm with complexity  $O(|V|^2)$  for solving the ELP problems. In Section 3, we establish that the functions  $\text{Bud}_R$  and  $\text{Rad}_B$  are convex on every edge  $uv$  of  $\mathcal{T}$ . Using this, we derive an  $O(|V|^2)$  algorithm for minimizing these functions on a given edge  $uv$ . The idea here is to start from an optimal solution at point  $s = u$  (derived by the algorithm developed in Section 2) and move  $s$  carefully toward  $v$  as long as  $\text{Bud}_R(s)$

decreases. We show how the solution for  $u$  can be updated to obtain an optimal solution for interior points  $s$  on  $uv$ . Applying the algorithm to each edge, we derive an  $O(|V|^3)$  algorithm for the DLP problems. Finally, in Section 4, we show that the functions  $\text{Bud}_R$  and  $\text{Rad}_B$  are quasiconvex on the whole network  $\mathcal{T}$ . This allows to avoid an exhaustive search of all edges of  $\mathcal{T}$  and to reduce the time complexity to  $O(|V|^2)$ .

## 2. ALGORITHMS FOR SOLVING THE ELP PROBLEMS

This section shows how to solve the eccentricity lowering problem for a given point  $s \in \mathcal{T}$  and a parameter  $R$ . To simplify the presentation, we assume that  $s$  is a vertex of  $T$ ; otherwise, if  $s$  is an inner point of the edge  $uv$ , we replace  $T$  by the tree  $T + s := (V \cup \{s\}, (E - \{uv\}) \cup \{us, vs\})$  in which the new edges  $us$  and  $sv$  have cost  $c(us) = c(sv) = c(uv)$  and length  $d_l(u, s)$  and  $d_l(s, v)$ , respectively, and solve the corresponding problem for  $T'$ .

To give some intuition, let us first present the algorithm as a continuous process (for the sake of time efficiency, the algorithm itself will be implemented as a discrete process). We decrease continuously and uniformly the length of all edges in a cut of minimum cost separating the root  $s$  from the set  $F_l(s)$  of vertices that are furthest away from  $s$  under the current length. A new minimum cut is computed each time an edge is fully contracted or a new vertex enters  $F_l(s)$ . In this case, we say that an *event* occurs (from the definition immediately follows that the total number  $n$  of events is at most  $|V| + |E|$ ). The process stops when the eccentricity of  $s$  in the shrunken tree becomes  $R$ .

The discrete process handles each event by finding in the current tree a new cut of minimum cost and computing the date of the next event (this also shows how much the edges of this cut have to be shrunken). In other words, we search for an optimal solution  $x_s$  of the problem  $\text{ELP}(T, l, c, s, R)$  which has the form  $x_s = \sum_{i=1}^n x_i$ , where each  $x_i$  is the *elementary contraction* of the  $i$ th minimum cut. The solution  $x_s$  is computed applying the recursive algorithm  $\text{UPGRADE}(T, l, c, s, R)$ . If  $R_l(s) = R$ , then it returns the solution  $x_s(e) = 0$  for each  $e \in E$ . Otherwise, let  $T_0$  be the tree obtained from  $T$  by contracting all edges  $e$  of  $T$  such that  $l(e) = 0$ . Among all cuts of  $T_0$  which separate  $s$  from the vertices of the set  $F_l(s)$ , the algorithm computes a cut  $C \subseteq E$  of minimum cost  $c(C) := \sum_{e \in C} c(e)$ . The cut  $C$  can be found in linear time using a postorder of the tree  $T$  rooted at  $s$ . To compute the value  $\alpha$  of the new contraction (and the date of the next event), we set

$$\alpha' := \min\{R_l(s) - d_l(s, v) : v \text{ is a leaf of } T \text{ such that } P(s, v) \cap C = \emptyset\}.$$

Note that  $\alpha'$  is the largest value of a contraction so that no new vertices enter  $F_l(s)$ . If after some event all leaves become equidistant to  $s$ , then the parameter  $\alpha'$  is not further

defined or used. Denote by  $\alpha''$  the minimum taken over the (residual) lengths of the edges from  $C$ . Now, set

$$\alpha := \min\{\alpha', \alpha'', R_l(s) - R\}, \quad (1)$$

and define the elementary reduction  $x'$  by letting  $x'(e) = \alpha$  if  $e \in C$  and  $x'(e) = 0$  otherwise. Clearly,  $x'$  is an optimal solution of the  $\alpha$ -contraction problem. Then, let  $x_s := x' + x''$ , where  $x'' := \text{UPGRADE}(T, l - x', c, s, R)$ . The formal description of the algorithm is given below. The minimum-cost cuts computed by the algorithm are pushed onto a stack  $\mathcal{C}$  which will be used in Algorithm 2.

**Algorithm 1**  $\text{UPGRADE}(T, l, c, s, R)$

```

if  $R_l(s) = R$  then
  for all  $e \in E$  do  $x'(e) := 0$ 
  return  $x'$ 
else
  let  $T_0$  be the tree obtained from  $T$  by contracting all
  edges  $e$  such that  $l(e) = 0$ 
  in  $T_0$  compute a cut  $C$  of minimum cost separating  $s$ 
  from  $F_l(s)$ 
  push  $C$  onto  $\mathcal{C}$ 
  compute  $\alpha$  using (1)
  for all  $e \in E$  do  $x'(e) := \alpha$  if  $e \in C$  and  $x'(e) = 0$ 
  otherwise
   $x'' := \text{UPGRADE}(T, l - x', c, s, R)$ 
  return  $(x' + x'')$ 

```

Since the number of recursive calls in the algorithm coincides with the number  $n \leq |V| + |E|$  of events and at each stage we solve in linear time a minimum-cut problem, the complexity of this algorithm is  $O(|V|^2)$ . Now, we turn to the proof of its correctness.

By rooting the tree  $T$  at the vertex  $s$ , we implicitly define partial orders on vertices and on edges of  $T$ : For two distinct edges  $e', e''$ , we set  $e' \prec_E e''$  if and only if  $e'$  belongs to the path between  $s$  and  $e''$  (the ordering  $\prec_V$  of vertices is similar). For a vertex  $v \neq s$ , by  $T_v$ , we denote the subtree of  $T$  induced by all vertices  $w$  such that  $v \prec_V w$ . Let  $F_v := T_v \cap F_l(s)$ . Finally, for two feasible contractions  $z', z'' : E \rightarrow \mathbb{R}^+ \cup \{0\}$ , we define  $z' \prec z''$  if  $z'(e) \leq z''(e)$  for all edges  $e$  of  $T$ . The *support* of a function  $z : E \rightarrow \mathbb{R}^+ \cup \{0\}$  is the set  $\text{supp}(z) := \{e \in E : z(e) > 0\}$ . With some abuse of notation, we will write  $v \in T$  and  $e \in T$  to mention that the vertex  $v$  and the edge  $e$  belong to tree  $T$ .

To prove the correctness, first we establish the following result:

**Lemma 1.** *For the optimal  $\alpha$ -contraction  $x'$  produced by the first recursive call of  $\text{UPGRADE}$  and for each  $\alpha < \beta \leq R_l(s)$ , there exists an optimal  $\beta$ -contraction  $x$  such that  $x' \prec x$ .*

**Proof.** Among the optimal solutions of the problem  $\text{ELP}(T, l, c, s, R)$ , we select a solution  $x$  which overlaps  $x'$

as much as possible, in the sense that  $x$  is an optimal solution of the following linear program:

$$\begin{aligned}
& \text{Maximize} && \sum_{e \in \text{supp}(x')} z(e) \\
& \text{subject to} && \begin{cases} \sum_{e \in E} c(e) \cdot z(e) \leq c_\beta \\ \sum_{e \in P} [l(e) - z(e)] \leq R_l(s) - \beta \quad \text{for } P \in \mathcal{P}_s \\ 0 \leq z(e) \leq l(e), e \in E, \end{cases}
\end{aligned} \tag{2}$$

where  $c_\beta$  is the minimum cost of a  $\beta$ -contraction (even if  $c_\beta$  is unknown, its existence suffices for the proof). We assert that  $x' \prec x$ . For this, we first establish two auxiliary facts:

**Fact 1.** *If  $z$  is a  $\gamma$ -contraction of minimum cost, then for each path  $P \in \mathcal{P}_s$ , we have  $\sum_{e \in P} z(e) \leq \gamma$ . In particular, if  $w \in F_l(s)$ , then  $\sum_{e \in P(s,w)} z(e) = \gamma$ .*

**Proof.** Suppose, by way of contradiction, that  $\mathcal{P}_s$  contains paths violating our assertion, and among them select a path  $P$  such that  $P \cap \text{supp}(z)$  is as large as possible. Let  $e' = uv$  be the edge that is furthest from  $s$  in the set  $P \cap \text{supp}(z)$ , and assume without loss of generality that  $u \prec_v v$ . By the choice of  $e'$ , the subtree  $T_v$  does not contain edges from the support of  $z$ . Therefore, by decreasing  $z(e')$  by  $\min\{z(e'), \sum_{e \in P} z(e) - \gamma\}$  units without changing  $z$  elsewhere, we obtain a  $\gamma$ -contraction of smaller cost, contrary to the optimality of  $z$ . ■

The conclusions of Fact 1 hold for the optimal solutions  $x'$  and  $x$ .

**Fact 2.** *If an edge  $e = uv$ ,  $u \prec_v v$ , belongs to  $\text{supp}(x')$  (i.e.,  $x'(e) = \alpha$ ) and  $x(e) < l(e)$ , then  $x(e') = 0$  for every  $e' \in T_v$ .*

**Proof.** The set  $F_v$  is nonempty because  $x'(e) = \alpha$ . Suppose, by way of contradiction, that the set  $M = T_v \cap \text{supp}(x)$  is nonempty. Let  $M'$  be the set of minimal elements of  $M$  with respect to the partial order  $\prec_E$ , (i.e.,  $e' \in M'$  and  $e' \prec_E e''$  for  $e'' \in M$  implies  $e' = e''$ ). Since  $M' \neq \emptyset$ , from Fact 1, we conclude that  $\sum_{e' \in P(v,s)} x(e') < \beta$ . Therefore, for each vertex  $w \in F_v$ , the path  $P(w, s)$  shares an edge with  $M'$ , whence  $M'$  is a simple cut separating the root  $s$  from  $F_v$ . The choice of  $x'$  yields  $\sum_{e' \in M'} c(e') \geq c(e)$ . Since  $x(e) < l(e)$ , if we decrease  $x$  by  $\epsilon := \min\{l(e) - x(e), \beta - x(e), x(e') : e' \in M'\}$  units on the edges of  $M'$  and increase  $x(e)$  by  $\epsilon$  units, we will obtain a better feasible solution  $y$  of (2), contrary to the choice of  $x$ . This establishes that  $x(e') = 0$  for all  $e' \in T_v$ . ■

Returning to the proof of Lemma 1, assume by way of contradiction that there exists an edge  $e' = u'v'$ ,  $u' \prec_v v'$ , such that  $\alpha = x'(e') > x(e')$ . The equality  $x'(e') = \alpha$

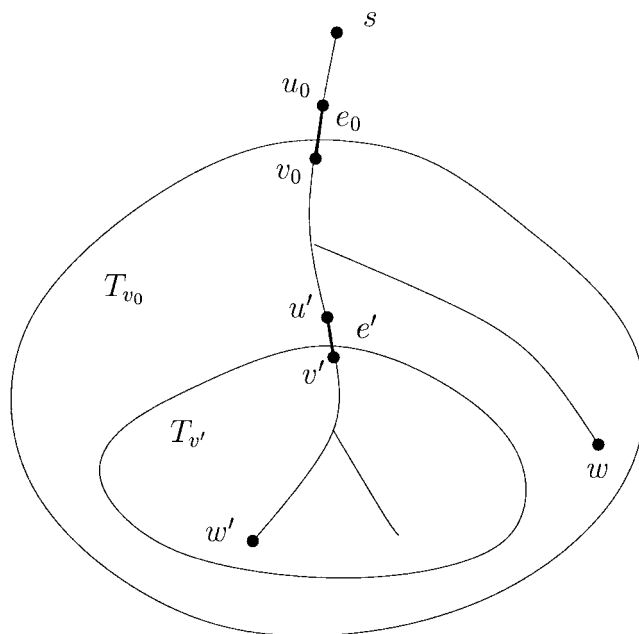


FIG. 1.

shows that the set  $F_{v'}$  is nonempty, say  $w' \in F_{v'}$ . Fact 1 implies that  $\sum_{e \in P(w',s)} x(e) = \beta > \alpha$ , while from Fact 2 we deduce that  $x(e) = 0$  for every edge of the subtree  $T_{v'}$ . Therefore,  $\sum_{e \in P(u',s)} x(e) > \beta - \alpha > 0$ . Suppose that  $e_0 = u_0v_0$  ( $u_0 \prec_v v_0$ ) is the edge from the set  $P(u', s) \cap \text{supp}(x)$  that is closest to  $e'$  (see Fig. 1). Hence,  $\sum_{e \in P(v_0,s)} x(e) > \beta - \alpha$ . Let  $\epsilon := \sum_{e \in P(v_0,s)} x(e) - \beta + \alpha > 0$ . Set  $C := \text{supp}(x') \cap T_{v_0}$ . From the definition of  $x'$ ,  $C$  is a cut of minimum cost separating the root  $s$  from the set  $F_{v_0} \supseteq F_{v'}$ . For a leaf  $w \in F_{v_0}$ , denote by  $e_w$  the unique edge of  $C$  lying on the path  $P(w, s)$ . Note that for every edge  $e \in C$  there exists a vertex  $w \in F_{v_0}$  such that  $e := e_w$ , that is,  $C = \{e_w : w \in F_{v_0}\}$ , and that  $e' = e_w$  for each  $w \in F_{v'}$ . By Fact 1, we have  $\sum_{e \in P(w,s)} x(e) = \beta$  for every  $w \in F_{v_0}$ . This equality and the definition of  $\epsilon$  imply that  $\sum_{e \in P(w,v_0)} x(e) = \alpha - \epsilon$ , whence  $x(e_w) < \alpha$  for each  $e_w \in C$ .

Now define  $y$  by setting  $y(e) := x(e) + \epsilon$  for  $e \in C$ ,  $y(e_0) := x(e_0) - \epsilon$ , and  $y(e'') = x(e'')$  for remaining edges  $e''$  of  $T$ . Clearly,  $c \cdot y \leq c \cdot x$ , while  $\sum_{e \in \text{supp}(x')} y(e) > \sum_{e' \in \text{supp}(x')} x(e)$ . We will get a contradiction with the choice of the optimal solution  $x$  provided we can establish that  $y$  is a feasible  $\beta$ -contraction. From (2), we deduce that  $y$  verifies the capacity constraint: The inequality  $0 \leq y(e) \leq l(e)$  holds for every edge  $e$ . It remains to verify that the eccentricity of  $s$  in the tree  $T$  endowed with the length function  $l - y$  is at most  $R_l(s) - \beta$ , that is, that  $d_{l-y}(s, w) \leq R$  for every leaf  $w$  of  $T$ . This is obviously true if  $w$  does not belong to  $T_{v_0}$ . From the previous analysis follows that the required inequality is also verified for all  $w \in F_{v_0}$ . Consider the remaining case  $w \in T_{v_0} - F_{v_0}$ , where the path  $P(w, s)$  does not share edges with the cut  $C$ . From the definition of  $\alpha$ , we conclude that  $d_l(s, w) \leq R_l(s) - \alpha$ . Hence,

$$\begin{aligned}
d_{l-y}(s, w) &= d_{l-x}(s, w) + \epsilon \leq d_l(s, w) - \sum_{e \in P(s, w)} x(e) + \epsilon \\
&= d_l(s, w) - \beta + \alpha \leq R_l(s) - \beta.
\end{aligned}$$

This concludes the proof of Lemma 1.  $\blacksquare$

**Proposition 1.** *The solution  $x_s = x' + x''$  returned by  $\text{UPGRADE}(T, l, c, s, R)$  is an optimum for  $\text{ELP}(T, l, c, s, R)$ .*

**Proof.** We proceed by induction on the number of recursive calls in the algorithm. Suppose by the induction hypothesis that  $x''$  is an optimal solution for  $\text{ELP}(T, l - x', c, s, R)$ . By Lemma 1, there exists an optimal solution  $x$  for  $\text{ELP}(T, l, c, s, R)$  such that  $x' \prec x$ . Then,  $x - x'$  is a feasible solution for  $\text{ELP}(T, l - x', c, s, R)$ . By optimality of  $x''$ , one concludes that  $cx'' \leq c(x - x')$ , therefore,  $c(x' + x'') \leq cx$ , thus yielding the optimality of  $x_s = x' + x''$ .  $\blacksquare$

This establishes that Algorithm 1 provides an optimal solution for  $\text{ELP}(T, l, c, s, R)$ . To solve the problem  $\text{ELP}^*(T, l, c, s, B)$ , we proceed in essentially the same way, only modifying the halting rule: The algorithm stops when the budget  $B$  is used up entirely (for further references, we call this version ‘‘Algorithm 1’’). We assert that the resulting eccentricity  $R$  of  $s$  is the minimum eccentricity which can be achieved under budget  $B$ . Suppose not, and let  $R' < R$  be an optimal eccentricity in problem  $\text{ELP}^*(T, l, c, s, B)$ . Denote by  $x'$  an optimal  $\beta'$ -contraction of  $T$ , where  $\beta' := R_l(s) - R'$ . From Proposition 1, we conclude that  $B$  is an optimal budget for the problem  $\text{ELP}(T, l, c, s, R)$ . Therefore,  $\sum_{e \in E} c(e) \cdot x'(e) = B$ . Now, one can find a small  $\epsilon > 0$  such that the solution obtained by decreasing  $x'$  by  $\epsilon$  on all edges of  $\text{supp}(x')$  is a feasible solution of the problem  $\text{ELP}(T, l, c, s, R)$ . Since its cost is strictly smaller than  $B$ , this leads us to a contradiction.

### 3. MINIMIZING THE BUDGET AND THE ECCENTRICITY FUNCTIONS

A real-valued function  $f$  defined on a tree-network  $\mathcal{T}$  is said to be *locally convex* if for every three points  $p, q, r$  lying on a common edge of  $\mathcal{T}$  and a real number  $\lambda$  between 0 and 1 such that

$$\begin{aligned}
d(p, q) &= d(p, r) + d(r, q), \\
d(p, r) &= \lambda d(p, q), \text{ and } d(r, q) = (1 - \lambda)d(p, q),
\end{aligned}$$

one has

$$f(r) \leq (1 - \lambda)f(p) + \lambda f(q). \quad (3)$$

**Proposition 2.** *The functions  $\text{Bud}_R$  and  $\text{Rad}_B$  are locally convex.*

**Proof.** Pick three points  $p, r, q$  on a common edge such that  $r$  lies between  $p$  and  $q$ . Clearly, if we subdivide the corresponding edge by adding these points to the vertex set of  $T$ , we do not change the set of feasible solutions of both problems. Thus, one may assume, without loss of generality, that  $p, r, q$  are vertices of  $T$  and  $e' := pr$  and  $e'' := rq$  are edges of length  $l(e') := d_l(p, r)$  and  $l(e'') := d_l(r, q)$  and of equal unit cost.

First, we establish the inequality (3) for the function  $\text{Bud}_R$ . Denote by  $x_p$  and  $x_q$  the optimal solutions of the problems  $\text{ELP}(T, l, c, p, R)$  and  $\text{ELP}(T, l, c, q, R)$ , that is, solutions of costs  $\text{Bud}_R(p)$  and  $\text{Bud}_R(q)$ , respectively. For an edge  $e \neq e', e''$ , define  $x_r(e) := (1 - \lambda)x_p(e) + \lambda x_q(e)$ . Set also  $x_r(e') := \lambda[x_q(e') + x_p(e'')]$  and  $x_r(e'') := (1 - \lambda)[x_p(e') + x_p(e'')]$ . We assert that  $x_r$  is an admissible solution of the problem  $\text{ELP}(T, l, c, r, R)$  (since the cost of  $x_r$  is  $(1 - \lambda)\text{Bud}_R(p) + \lambda\text{Bud}_R(q)$ , this establishes (3) for  $\text{Bud}_R$ ). Indeed, if  $e \neq e', e''$ , then

$$\begin{aligned}
x_r(e) &:= (1 - \lambda)x_p(e) + \lambda x_q(e) \\
&\leq (1 - \lambda)l(e) + \lambda l(e) = l(e).
\end{aligned}$$

Also,

$$\begin{aligned}
x_r(e') &= \lambda[x_q(e') + x_p(e'')] \leq [d(p, r)/d(p, q)] \\
&\quad [d(p, r) + d(r, q)] = d(p, r)
\end{aligned}$$

(the proof that  $x_r(e'') \leq l(e'')$  is identical). It remains to show that  $R_{l-x_r}(r) \leq R$ . Pick a leaf  $w$  of  $T$ . Suppose, without loss of generality, that  $d_l(w, q) = d_l(w, p) + d_l(p, q)$ . Then,

$$\begin{aligned}
d_{l-x_r}(w, r) &= d_{l-x_r}(w, p) + d_{l-x_r}(p, r) \\
&= \sum_{e \in P(w, p)} [l(e) - x_r(e)] + [l(e') - x_r(e')] \\
&= (1 - \lambda) \sum_{e \in P(w, p)} [l(e) - x_p(e)] \\
&\quad + \lambda \sum_{e \in P(w, p)} [l(e) - x_q(e)] + \lambda[l(e') \\
&\quad + l(e'') - x_q(e') - x_q(e'')] \\
&= (1 - \lambda)d_{l-x_p}(w, p) + \lambda d_{l-x_q}(w, q) \\
&\leq (1 - \lambda)R + \lambda R = R.
\end{aligned}$$

Analogously, to establish (3) for  $\text{Rad}_B$ , assume that  $x_p$  and  $x_q$  are optimal solutions of the problems  $\text{ELP}^*(T, l, c, p, B)$  and  $\text{ELP}^*(T, l, c, q, B)$  and define  $x_r$  as above. From that proof, one concludes immediately that the eccentricity of the vertex  $r$  in the tree  $T_{l-x_r}$  is at most  $(1 - \lambda)\text{Rad}_B(p) + \lambda\text{Rad}_B(q)$ ; thus, (3) will be verified if we will show that

the cost of the solution  $x_r$  is at most  $B$ . Indeed, since  $c(e') = c(e'')$ , we obtain

$$\begin{aligned}
\sum_{e \in E} c(e)x_r(e) &= \sum_{e \in E - \{e', e''\}} c(e)x_r(e) + \lambda c(e')[x_q(e') \\
&\quad + x_q(e'')] + (1 - \lambda)c(e'')[x_p(e') + x_p(e'')] \\
&= (1 - \lambda) \sum_{e \in E - \{e', e''\}} c(e)x_p(e) \\
&\quad + (1 - \lambda)[c(e')x_p(e') + c(e'')x_p(e'')] \\
&\quad + \lambda \sum_{e \in E - \{e', e''\}} c(e)x_q(e) \\
&\quad + \lambda[c(e')x_q(e') + c(e'')x_q(e'')] \\
&\leq \lambda B + (1 - \lambda)B = B.
\end{aligned}$$

■

We continue with an algorithm for minimizing the function  $\text{Bud}_R$  on an edge  $e^* = uv$  of  $\mathcal{T}$ . Given a point  $s$  on  $uv$ , it will be convenient to consider  $s$  as a vertex of  $T$ , that is, to denote by  $T$  by tree  $T + s$  in which the edge  $uv$  is subdivided into two edges  $us$  and  $sv$  of length  $l(us) = d_l(u, s)$ ,  $l(sv) = d_l(s, v)$  and cost  $c(us) = c(sv) = c(uv)$ . Denote by  $T'$  and  $T''$  the connected components obtained after removing the edge  $sv$  from  $T$ ; suppose that  $u, s \in T'$  and  $v \in T''$ . Let  $x_u = \sum_{i=1}^n x_i$  be an optimal solution of the problem  $\text{ELP}(T, l, c, u, R)$  returned by Algorithm 1. Let  $C_i$  be the support of the elementary contraction  $x_i$ ,  $i = 1, \dots, n$ . The cuts  $C_1, \dots, C_n$  are kept by Algorithm 1 in a stack  $\mathcal{C}$ . We assume that two standard functions on  $\mathcal{C}$  are available:  $\text{HEAD}(\mathcal{C})$  returns the cut in head of  $\mathcal{C}$  and  $\text{POP}(\mathcal{C})$  deletes this cut from  $\mathcal{C}$ . Finally, notice that the edge  $e^*$  also may belong to the support of  $x_u$ . If this happens, Algorithm 1 implies that either  $x_u(e^*) = l(e^*)$  or the inequality  $c(e^*) \geq c(C_i \cap T'')$  holds for all  $C_i \in \mathcal{C}$ .

The rough idea behind the discrete process described below is very intuitive. If we move a point  $s$  from  $u$  toward  $v$  by a small  $\alpha$ , we increase by  $\alpha$  the distance from  $s$  to the points of  $T'$  and decrease by  $\alpha$  the distance to the points of  $T''$ . Therefore, we can diminish by  $\alpha$  the contraction along each of the edges from a cut  $C''$  separating  $s$  from  $F_{l-x_u}(u) \cap T''$  (i.e., we dilate back each edge of this cut by  $\alpha$  units). As with  $C''$ , it is more convenient to select the shore from  $T''$  of the cut in head of  $\mathcal{C}$  (i.e.,  $\text{HEAD}(\mathcal{C} \cap T'')$ ); however, if  $x_u(e^*) = l(e^*)$ , we should set  $C'' := \{e^*\}$  in order to preserve the feasibility of the reduction. Now, we can dispense a part of  $\alpha \cdot c(C'')$  units of free budget to update the distances from  $s$  to the leaves of  $F_{l-x_u}(u) \cap T'$ . For this, we must find a minimum cut  $C'$  which separates  $s$  from  $F_{l-x_u}(u) \cap T'$  in the tree in which all edges  $e$  such that  $l(e) = x_u(e)$  have been contracted. If  $c(C') < c(C'')$ , then, obviously,  $\text{Bud}_R(s) < \text{Bud}_R(u)$ , otherwise,  $\text{Bud}_R(s) \geq \text{Bud}_R(u)$ . In the first case, we set  $s_1 := s$  and continue the

movement toward  $v$ . As a consequence, we will construct a sequence of points  $u = s_0, s_1, s_2, \dots, s_m$  such that  $\text{Bud}_R(s_0) > \text{Bud}_R(s_1) > \text{Bud}_R(s_2) > \dots > \text{Bud}_R(s_m)$ . The halting rule is that either  $s_i = v$  or  $\text{Bud}_R(s_{i+1}) \geq \text{Bud}_R(s_i)$  holds. If each time  $s_{i+1}$  is chosen so that the selected cuts for any point between  $s_i$  and  $s_{i+1}$  coincide with those for  $s_{i+1}$ , then the final point  $s_m$  is a local minimum of the function  $\text{Bud}_R$  on  $e^*$ , whence  $s_m$  is a global minimum by Proposition 2.

We implement this discrete process as a recursive algorithm  $\text{MOVE}(T, l, c, s, u, v, x)$ .  $\text{MOVE}$  takes a current point  $s$  on the edge  $e^* = uv$  and its optimal upgrading  $x := x_s$  as an input and either returns  $s$  and  $x$  or finds a new point (we use the same name  $s$  for this point and only change its distance to  $u$ ) with a better upgrading  $x'$  and makes a recursive call with the new parameters. To find the global minimum of  $\text{Bud}_R$  on the edge  $uv$ , we call  $\text{MOVE}$  with  $x = x_u$  and the length function  $l$  modified so that  $l(us) = 0$ ,  $l(sv) = l(uv)$  (i.e.,  $s$  coincides with  $u$ ). As we noticed already,  $s$  is considered as a vertex of  $T$ . Using this agreement, moving  $s$  toward  $v$  by  $\alpha$  units is nothing else but increasing  $l(us)$  by  $\alpha$  and decreasing  $l(sv)$  by  $\alpha$ . Hence, a recursive call will have the form  $\text{MOVE}(T, l', c, s, u, v, x')$ . It remains to precise how to compute  $l'$ ,  $\alpha$ , and  $x'$ .

Let  $T_0$  be the tree obtained from  $T$  by contracting all edges  $e$  such that  $l(e) = x(e)$ . In  $T_0$ , find a cut  $C'$  of minimum cost separating the vertex  $s$  from  $F_{l-x}(s) \cap T'$  (set  $C' = \emptyset$  if this intersection is empty). Let  $C'' = \text{HEAD}(\mathcal{C}) \cap T''$  if  $\mathcal{C}$  is nonempty and  $x(sv) < l(sv)$ . Otherwise, set  $C'' := \{sv\}$  if  $x(sv) = l(sv)$ , and  $C'' := \emptyset$  if not. Now, if  $c(C') \geq c(C'')$ , then  $\text{MOVE}$  returns  $s$  and  $x$ . Otherwise, if  $c(C') < c(C'')$ , the algorithm computes the value  $\alpha$  which indicates how much the edges of  $C'$  have to be contracted and how much the edges of  $C''$  have to be dilated:

$$\alpha = \min\{\alpha', \alpha'', \alpha''', d_l(s, v)\}, \quad (4)$$

where

$$\alpha' = \min\{l(e) : e \in C'\},$$

$$\alpha'' = \min\{x(e) : e \in C''\},$$

and

$$\alpha''' = \min\{R - d_l(s, w) :$$

$$w \text{ is a leaf of } T' \text{ such that } P(s, w) \cap C' = \emptyset\}.$$

Now, define  $x'$  by setting  $x'(e) := x(e) + \alpha$  for all  $e \in C'$ ,  $x'(e) := x(e) - \alpha$  for all  $e \in C''$ , and  $x'(e) := x(e)$  elsewhere. Update also the length function by setting  $l'(us) := l(us) + \alpha$ ,  $l'(sv) := l(sv) - \alpha$ , and  $l'(e) := l(e)$  for all edges  $e \neq us, sv$ . Finally, pop the cut in head of  $\mathcal{C}$  provided it coincides with  $C''$  and  $\alpha = \min\{x(e) : e \in C''\}$ . Since at each recursive call (alias event) either  $C''$

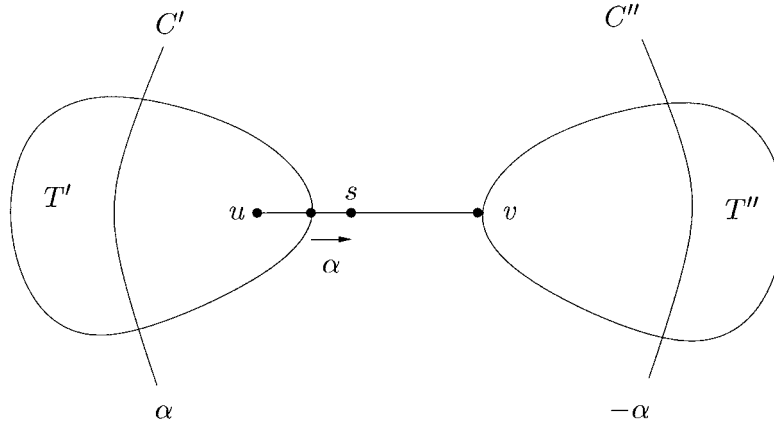


FIG. 2.

is popped from  $\mathcal{C}$  or a new minimum cut  $C'$  separating  $s$  from  $F_{l-x}(s) \cap T'$  is computed (because either a new vertex enters  $F_{l-x}(s)$  or a new edge of  $T$  becomes fully contracted), the number of recursive calls is linear in  $|V|$ ; hence, the whole execution of the algorithm **MOVE** takes  $O(|V|^2)$  time. A formal description of this procedure is given below (for an illustration, see Fig. 2).

**Algorithm 2**  $\text{MOVE}(T, l, c, s, u, v, x)$

**if**  $l(sv) = 0$  **then**

**return**  $(l(us), x)$

**else**

    let  $T_0$  be the tree obtained from  $T$  by contracting all edges  $e$  such that  $l(e) = x(e)$

    let  $C' := \begin{cases} \emptyset, & \text{if } F_{l-x}(s) \cap T' = \emptyset \\ \text{a cut of minimum cost of } T_0 \text{ separating} \\ \text{ } s \text{ from } F_{l-x} \cap T', & \text{otherwise} \end{cases}$

    let  $C'' := \begin{cases} \{sv\}, & \text{if } l(sv) = x(sv) \\ \text{HEAD}(\mathcal{C}) \cap T'', & \text{if } x(sv) < l(sv) \text{ and } \mathcal{C} \neq \emptyset \\ \emptyset, & \text{otherwise} \end{cases}$

**if**  $c(C') \geq c(C'')$  **then**

**return**  $(l(us), x)$

**else**

        compute  $\alpha$  by (4)

**if**  $\alpha = \min\{x(e) : e \in C''\}$  and  $C'' = \text{HEAD}(\mathcal{C}) \cap T''$ , **then**  $\text{POP}(\mathcal{C})$

$x'(e) := \begin{cases} x(e) + \alpha & \text{if } e \in C' \\ x(e) - \alpha & \text{if } e \in C'' \\ x(e) & \text{otherwise} \end{cases}$

$l'(e) := \begin{cases} l(e) + \alpha & \text{if } e = us \\ l(e) - \alpha & \text{if } e = sv \\ l(e) & \text{otherwise} \end{cases}$

**return**  $\text{MOVE}(T, l', c, s, u, v, x')$

Denote by  $\text{MOVE}_1$  the same algorithm as **MOVE** except that in the last line  $\text{MOVE}_1$  returns the pair  $(l'(su), x')$  instead of the recursive call of **MOVE**.

**Proposition 3.** *The point  $s$  and the upgrading  $x$  returned by algorithm **MOVE** minimizes the function  $\text{Bud}_R$  on the edge  $e^* = uv$ .*

**Proof.** It suffices to show that if  $s$  is the point computed by  $\text{MOVE}_1$ , then (i)  $x_s := x'$  is an optimal solution of the problem  $\text{ELP}(T, l, c, s, R)$  and (ii) between  $u$  and  $s$  the function  $\text{Bud}_R$  is linearly decreasing. A case analysis using the definition of  $\alpha$  implies that  $x_s$  is a feasible solution. Notice that  $x_s$  is optimal if and only if its restrictions  $x'_s$  and  $x''_s$  on  $T'$  and  $T'' + sv$ , respectively, are optimal. To prove the optimality of  $x'_s$ , notice that  $x'_s$  can be viewed as a solution returned by **UPGRADE**, except the case when  $C' = \{us\}$  and the cost  $c(us)$  is located between  $c(C_i \cap T')$  and  $c(C_{i+1} \cap T')$  for some  $i < n$ . Then, again,  $x'_s$  can be considered as a solution returned by **UPGRADE**. First, this algorithm recovers the elementary contractions  $x'_1, \dots, x'_i$  along the cuts  $C_1 \cap T', \dots, C_i \cap T'$ ; next, it contracts the edge  $us$  (this does not change the list of furthest from  $s$  leaves of  $T'$ ) and then it returns the contractions  $x'_{i+1}, \dots, x'_n$  along the cuts  $C_{i+1} \cap T', \dots, C_n \cap T'$ . Applying the same reasoning, one can show that if  $C''$  is the most expensive cut from the family  $\{C_j \cap T'' : C_j \in \mathcal{C}\}$  then  $x''_s$  is optimal. Finally, suppose that  $C'' = \{sv\}$ , that is, the equality  $x_u(uv) = l(uv)$  holds. Then,

$$\sum_{e \in T'' + e^*} c(e)x_u(e) - \sum_{e \in T'' + sv} c(e)x_s(e) = c(e^*)d_l(u, s).$$

Now, if one assumes that  $y''$  is a solution of  $\text{ELP}(T'', l, c, s, R)$  of cost smaller than  $x''_s$ , then setting  $y''_u(e) = y''(e)$  if  $e \in T''$ ,  $e \neq e^*$ , and  $y''_u(e^*) = y''(sv) + d_l(u, s)$ , we will get a better solution at  $u$ , contrary to the optimality of  $x_u$ . Finally, notice that from  $u$  to  $s$  the function  $\text{Bud}_R$  is linearly decreasing: Its graph is a line segment with slope  $c(C') - c(C'')$ . ■

Algorithm 2 can be modified to compute, within the same time bounds, a point of  $e^*$  which minimizes the

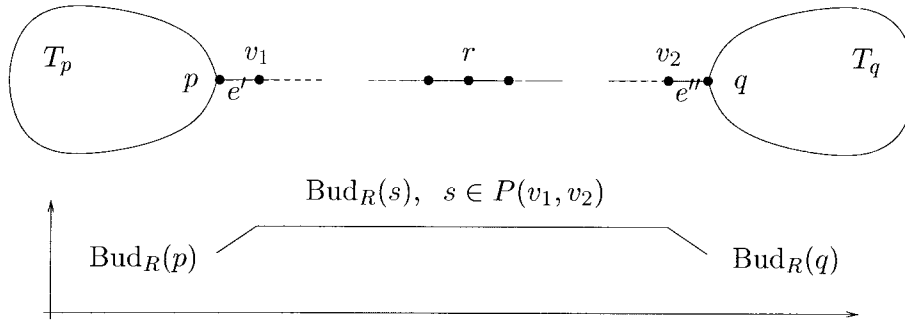


FIG. 3.

eccentricity function  $\text{Rad}_B$ , provided that  $B > 0$ , namely, we show that the point  $s$  minimizing the function  $\text{Bud}_R$  minimizes also  $\text{Rad}_B$ . Then, it will suffice to solve the problem  $\text{ELP}(T, l, c, s, R)$  employing Algorithm 1'. Our assertion is a consequence of the following more general property:

**Lemma 2.** *Let  $R > 0$  and  $B := \text{Bud}_R(p) > 0$  for some point  $p \in \mathcal{T}$ . Then,  $\text{Bud}_R(p) > \text{Bud}_R(q)$  if and only if  $\text{Rad}_B(p) > \text{Rad}_B(q)$ .*

**Proof.** First, we establish the following claim:  $\text{Rad}_B(p) = R$ .

Let  $x_p$  be an optimal solution of the problem  $\text{ELP}(T, l, c, p, R)$ . Since  $c(x_p) = \text{Bud}_R(p) = B$ ,  $x_p$  is a feasible solution of the problem  $\text{ELP}^*(T, l, c, p, B)$ , whence  $\text{Rad}_B(p) \leq R$ . Suppose, by way of contradiction, that an optimal solution  $x'_p$  of  $\text{ELP}^*(T, l, c, p, B)$  verifies the inequality  $R_{l-x'_p}(p) = R' < R$ . Since  $\text{supp}(x'_p) \neq \emptyset$ , one can decrease  $x'_p$  uniformly on one or several cuts separating  $p$  from  $F_l(p)$  and obtain a feasible solution of  $\text{ELP}(T, l, c, p, R)$  whose cost is strictly less than  $B$ , a contradiction. This establishes the claim.

Let  $B' := \text{Bud}_R(q) < \text{Bud}_R(p) = B$ . Since  $B' < B$ , from the claim and  $R > 0$ , one concludes that  $\text{Rad}_B(p) = R = \text{Rad}_B(q) > \text{Rad}_B(q)$ . Conversely, suppose that  $R' = \text{Rad}_B(q) < \text{Rad}_B(p) = R$ . Hence,  $\text{Bud}_R(p) = B \geq \text{Bud}_R(q)$ . Since  $R' < R$  and  $B > 0$ , we obtain that  $\text{Bud}_R(p) = B = \text{Bud}_R(q) > \text{Bud}_R(q)$ , establishing the inequality  $\text{Bud}_R(p) > \text{Bud}_R(q)$ . ■

As we noticed already, the number  $m$  of events in Algorithm 2 is linear; therefore, the optimal point  $s_m$  of  $e^*$  can be computed in  $O(|V|^2)$  time. Applying this procedure to each edge of  $\mathcal{T}$ , one can find a global minimum of the functions  $\text{Bud}_R$  and  $\text{Rad}_B$  in total  $O(|V|^3)$  number of operations. Now, if  $s^*$  is the global minimum of the function  $\text{Rad}_B$  on  $\mathcal{T}$ , then  $2\text{Rad}_B(s^*)$  is the least diameter of an upgrading of  $T$  under budget  $B$ . Therefore, the DLP problems can be solved in  $O(|V|^3)$  time.

#### 4. QUASICONVEXITY AND A QUADRATIC ALGORITHM FOR SOLVING THE DLP PROBLEMS

One can ask whether the functions  $\text{Bud}_R$  and  $\text{Rad}_B$  are globally convex, that is, if (3) is verified by arbitrary three points  $p, q, r$  such that  $d_l(p, r) + d_l(r, q) = d_l(p, q)$ . Unfortunately, simple examples show that this is not the case, even whence not every local minimum of either of these functions is a global minimum. Nevertheless, the functions  $\text{Bud}_R$  and  $\text{Rad}_B$  verify a property called quasiconvexity, which, together with local convexity, would suffice to justify a “gradientlike” algorithm for their minimization. A function  $f$  defined on  $\mathcal{T}$  is *quasiconvex* if for every three points  $p, q, r$  such that  $d(p, q) = d(p, r) + d(r, q)$  we have

$$f(r) \leq \max\{f(p), f(q)\}.$$

**Proposition 4.**  *$\text{Bud}_R$  and  $\text{Rad}_B$  are quasiconvex functions on  $\mathcal{T}$ .*

**Proof.** Suppose, by way of contradiction, that there exist two points  $p \in e', q \in e''$  and a point  $r$  between  $p$  and  $q$  such that  $\text{Bud}_R(r) > \max\{\text{Bud}_R(p), \text{Bud}_R(q)\}$ . Denote by  $x_p$  and  $x_q$  the optimal solutions of the problems  $\text{ELP}(T, l, c, p, R)$  and  $\text{ELP}(T, l, c, q, R)$  derived by Algorithm 1. Suppose that  $p$  and  $q$  are the closest to  $r$  minima of the function  $\text{Bud}_R$  on  $e'$  and  $e''$ , respectively, and assume, without loss of generality, that  $p$  and  $q$  are end points of  $e'$  and  $e''$ , say  $e' := pv_1$  and  $e'' := v_2q$ , and that the function  $\text{Bud}_R$  is constant along the path  $P(v_1, v_2)$ . Root  $T$  at the point  $r$  and define the subtrees  $T_p$  and  $T_q$  as before. From Proposition 2, we infer that the edges  $e' := pv_1$  and  $e'' := v_2q$  are distinct and that  $\text{Bud}_R$  strongly increases while moving from  $p$  to  $v_1$  and from  $q$  to  $v_2$  (see Fig. 3); in particular,

$$\text{Bud}_R(p) < \text{Bud}_R(v_1) \text{ and } \text{Bud}_R(v_2) > \text{Bud}_R(q). \quad (5)$$

This immediately implies that  $x_p(e') = 0 = x_q(e'')$  (otherwise, if say  $x_p(e') > 0$ , then  $\text{Bud}_R(s) \leq \text{Bud}_R(p)$  for every point  $s$  of  $e'$  at distance at most  $x_p(e')$  from  $p$ ) and

that there must be a leaf  $w_1 \in T_p \cap F_l(q)$  and a leaf  $w_2 \in T_q \cap F_l(p)$ . Next, one can suppose, without loss of generality, that every vertex  $u \neq v_1, v_2$  of the path  $P(v_1, v_2)$  has degree two, that is, that  $T$  coincides with  $T_p \cup P(v_1, v_2) \cup T_q$ , because the inequalities  $\text{Bud}_R(p) < \text{Bud}_R(v_1)$  and  $\text{Bud}_R(v_2) > \text{Bud}_R(q)$  are obviously verified if we will replace  $T$  by the tree  $T_p \cup P(v_1, v_2) \cup T_q$ . Finally, one can assume that the eccentricity of  $p$  in  $T_p$  is exactly  $R$  as well as the eccentricity of  $q$  in the subtree  $T_v$ . Indeed, either of these eccentricities could not be smaller than  $R$ ; otherwise, the function  $\text{Bud}_R$  will strongly decrease while moving from the respective vertex toward  $r$ , which is impossible. Now, if the eccentricity of  $p$  in  $T_p$  is larger than  $R$ , then an optimal solution  $x$  of the problem  $\text{ELP}(T_p, l, c, p, R)$  computed by Algorithm 1 extends in a straightforward way to an optimal solution of the problem  $\text{ELP}(T_p \cup P(p, s), l, c, s, R)$ , where  $s$  is an arbitrary point on the path between  $p$  and  $q$  (see Lemma 1). Thus, replacing  $l$  by  $l - x$ , and eventually contracting all edges whose new lengths are 0, we will obtain a tree in which the inequalities (5) are verified. Hence,  $d_l(p, w_1) = R = d_l(q, w_2)$ , that is,  $x_p(e) = 0$  for every edge  $e \in T_p$  and  $x_q(e) = 0$  for every edge  $e \in T_q$ .

Denote by  $C_p''$  and  $C_q''$  the most expensive cuts in the solutions  $x_p$  and  $x_q$ , respectively (e.g.,  $C_p''$  is either an edge of  $P(p, q)$  or a cut of  $T_q$ ). Further, let  $C_p'$  be a cut of minimum cost of  $T_p$  which separates  $p$  from the set  $F_l(q) \cap T_p$  (notice that all these leaves are at distance  $R$  from  $p$ ). Analogously, let  $C_q'$  be the cut of minimum cost of  $T_q$  separating  $q$  from  $F_l(p) \cap T_q$ . Notice that if  $\text{supp}(x_q) \cap T_p \neq \emptyset$  then, by Algorithm 1, one can suppose that  $C_p'$  belongs to the support of  $x_q$ . This indeed is always the case: If  $\text{supp}(x_q) \cap T_p = \emptyset$ , then, to achieve at  $q$  the eccentricity  $R$ , the path between  $p$  and  $q$  must be fully contracted, that is,  $x_q(e) = l(e)$  for every edge  $e \in P(p, q)$ . This is impossible because  $x_q(e'') = 0$ . Thus,  $C_p'$  and  $C_q'$  belong to the supports of the solutions  $x_q$  and  $x_p$ , respectively, whence  $c(C_p') \leq c(C_q'')$  and  $c(C_q') \leq c(C_p'')$ . On the other hand, since  $x_p(e') = 0$  and the function  $\text{Bud}_R$  is strictly increasing while moving from  $p$  to  $v_1$ , from Algorithm 2, we deduce that  $c(C_p') > c(C_p'')$ . Analogously,  $c(C_q') > c(C_q'')$ . Combining these four inequalities, we obtain

$$c(C_p') \leq c(C_q'') < c(C_q') \leq c(C_p'') < c(C_p'),$$

an obvious contradiction. This establishes the quasiconvexity of the function  $\text{Bud}_R$ . From Lemma 2, we conclude that  $\text{Rad}_B$  is quasiconvex as well. ■

As we already noticed in Section 1, to solve the problem  $\text{DLP}(T, l, c, D)$ , it suffices to find a point  $s^*$  of the tree network  $\mathcal{T}$  which minimizes the function  $\text{Bud}_R$  for  $R = D/2$ . In this section, we will describe an algorithm which computes such a point without an exhaustive search of all minima of  $\text{Bud}_R$  on the edges of  $T$ . We start with a few notions and remarks.

The tree network  $\mathcal{T}$  can be viewed as the disjoint union

of the level sets  $\{p \in \mathcal{T} : \text{Bud}_R(p) = \alpha\}$ ,  $\alpha \geq 0$ , of the function  $\text{Bud}_R$ . For a point  $p \in \mathcal{T}$ , denote by  $L(p)$  the connected component of the level set  $\{q \in \mathcal{T} : \text{Bud}_R(q) = \text{Bud}_R(p)\}$  containing  $p$ . We say that the function  $\text{Bud}_R$  is *locally constant* on an edge  $uv$  in the neighborhood of  $u$  if there exists a point  $p \in uv$ ,  $p \neq u$ , such that the whole segment  $up$  belongs to  $L(u)$ . If  $p$  coincides with  $v$ , then  $\text{Bud}_R$  is constant on the edge  $uv$ . From Proposition 2, we conclude that if  $\text{Bud}_R$  is locally constant on  $uv$  and  $p$  is the furthest from  $u$  point of  $uv$  such that  $up \subseteq L(p)$  then either  $p = v$  and  $\text{Bud}_R$  is constant on  $uv$  or  $\text{Bud}_R$  increases on the segment  $pv$ . On the other hand, Proposition 4 and the previous remark imply that, for a vertex  $u$  of  $T$ , either  $u$  and all points of  $L(u)$  minimize  $\text{Bud}_R$  or there exists a unique vertex  $t \in L(u)$  and a neighbor  $s$  of  $t$  in  $T$  such that  $\text{Bud}_R$  is decreasing while moving from  $t$  to  $s$ . If the minimum of  $\text{Bud}_R$  on the edge  $ts$  is attained at some inner point of  $ts$ , then the quasiconvexity of  $\text{Bud}_R$  implies that this point is the unique global minimum of the function  $\text{Bud}_R$  on  $\mathcal{T}$ .

The algorithm explores the tree network  $\mathcal{T}$  starting from an arbitrary point or vertex  $s_0$  (as  $s_0$  one can take the absolute center of  $\mathcal{T}$ ); to simplify the presentation, we assume that  $s_0$  is a vertex of  $T$ . First, applying UPGRADE with a minor modification, we compute an optimal solution  $x_{s_0}$  of the problem  $\text{ELP}(T, l, c, s_0, R)$ . This modification concerns the choice of the cut  $C$ : Each time when a cut  $C$  of minimum cost is chosen, we pick a cut which additionally minimizes the sum of distances from its edges to the root  $s_0$ . Clearly,  $C$  can be computed in linear time using a postorder of the current tree  $T$  rooted at  $s_0$ : While traversing an edge  $uv$ ,  $u \prec_v v$ , of  $T$  for the last time, if  $c(uv)$  is smaller than or equal to the cost of already computed minimum cost cut of  $T_v$ , the algorithm will pick  $uv$  as a cut of minimum cost of  $T_u$ .

Departing from the solution  $x_{s_0}$  and the associated collection  $\mathcal{C} = \{C_1, \dots, C_n\}$  of cuts, the algorithm explores the tree subnetwork  $L(s_0)$  to find a leaf  $t_0$  of  $L(s_0)$  and an edge  $e_0$  incident to  $t_0$  along which the function  $\text{Bud}_R$  decreases. If  $t_0$  is found, then, applying MOVE to  $t_0$  and  $x_{t_0}$ , we will find a point  $s_1$  minimizing the function  $\text{Bud}_R$  on the edge  $e_0$ . If  $s_1$  is an inner point of  $e_0$ , then the algorithm returns  $s_1$ . Otherwise, if  $s_1$  is another end-vertex of  $e_0$ , then we explore  $L(s_1)$  to find the leaf  $t_1$  of  $L(s_1)$  and the edge  $e_1$  incident to  $t_1$  along which  $\text{Bud}_R$  decreases, and so on. The algorithm halts in iteration  $m$  if the corresponding leaf  $t_m$  of  $L(s_m)$  does not exist, that is, if all leaves of  $L(s_m)$  are local minima of the function  $\text{Bud}_R$  (for an illustration, see Fig. 4). We claim that, in this case,  $s_m$  (as well as any other point of  $L(s_m)$ ) is a global minimum of the function  $\text{Bud}_R$ . Suppose, by way of contradiction, that there exists a point  $p$  such that  $\text{Bud}_R(p) < \text{Bud}_R(s_m)$ . Since  $\text{Bud}_R$  is quasiconvex, while moving from  $s_m$  to  $p$ , first we will stay inside the set  $L(s_m)$ ; then, the function  $\text{Bud}_R$  will decrease along an edge sharing an end-vertex with  $L(s_m)$ , contrary to the halting rule. From the local convexity of  $\text{Bud}_R$ , one concludes also that every  $t_i$  must be a vertex of the tree  $T$ .

It remains to specify how the sets  $L(s_i)$ , the vertices  $t_i$ ,

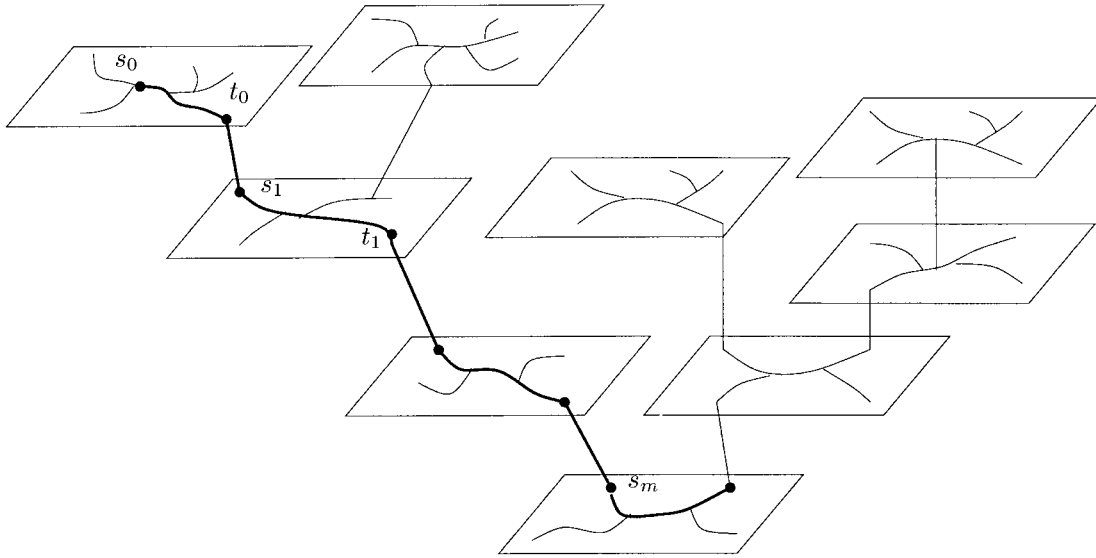


FIG. 4.

and the edges  $e_i$  are computed. In fact, to find the vertex  $t_i$ , we have to explore only a certain subset of vertices and edges of the tree subnetwork  $L(s_i)$ . Let  $s$  be the current vertex of  $L(s_i)$ . For  $s$ , the algorithm has already computed an optimal solution  $x_s$  and the associated collection  $\mathcal{C}$  of elementary cuts. For each yet unexplored neighbor  $v$  of  $s$ , we run the algorithm  $\text{MOVE}_1$  on the edge  $sv$ . As a result, we will conclude that on  $sv$  the function  $\text{Bud}_R$  either increases or decreases or is locally constant. In the first case, the subtree  $T_v$  is discarded from the subsequent exploration. In the second case, we set  $t_i := v$ ,  $e_i := sv$ , and then run  $\text{MOVE}$  on the entire edge  $e_i$  to find the global minimum  $s_{i+1}$  of  $\text{Bud}_R$  on this edge (the algorithm also returns an optimal solution  $x_{s_{i+1}}$  and updates the collection  $\mathcal{C}$ ). Finally, assume that  $s$  is a local minimum of the function  $\text{Bud}_R$ ; however,  $\text{Bud}_R$  is locally constant on some edges  $sv$ . Since  $t_i$  is a vertex of  $T$ , we must explore only the vertices  $v$  such that  $\text{Bud}_R$  is constant on the edge  $sv$  and discard all other vertices adjacent to  $s$  from the subsequent search. Hence, it remains to detect efficiently the edges  $e = sv$  on which  $\text{Bud}_R$  is constant and, for every respective vertex  $v$ , to compute an optimal solution  $x_v$  and the underlying collection of cuts. This is an obvious task if the respective edge  $e$  is fully contracted in the solution  $x_s$  (i.e., if  $x_s(e) = l(e)$ ). In this case, we simply set  $x_v = x_s$  and insert  $v$  in  $L(s_i)$ . The collection of cuts does not change, only the edge  $e$  passes from  $\mathcal{C} \cap T_s$  to  $\mathcal{C} \cap (T \setminus T_s)$ . Next, we prove that all other vertices adjacent to  $s$  can be discarded. For this, we establish a slightly more general assertion. For an edge  $e = uv$ , let  $T'$  and  $T''$  be the connected components of  $T - e$ , where  $u \in T'$  and  $v \in T''$ .

**Claim.** *Let  $e = uv$  be an edge of  $T$  and let  $x_u$  be an optimal solution for  $\text{ELP}(T, l, c, u, R)$  computed by the modification of  $\text{UPGRADE}$ . If  $x_u(e) < l(e)$  and  $\text{Bud}_R$  is locally*

*constant on  $e$  in a neighborhood of  $u$ , then  $\text{Bud}_R(s) > \text{Bud}_R(u)$  for every point  $s \in \mathcal{T}''$ .*

**Proof.** We distinguish two cases:

**CASE 1.**  $x_u(e) > 0$ . Let  $\mathcal{C}$  be the collection of cuts defining the solution  $x_u$ , and let  $\mathcal{C}' = \{C \cap T' : C \in \mathcal{C}\}$  and  $\mathcal{C}'' = \{C \cap T'' : C \in \mathcal{C}\}$ . Since  $0 < x_u(e) < l(e)$ , Algorithm 1 infers that  $e$  is a cut of maximum cost in the collection  $\mathcal{C}''$ . Moreover, the amend to this algorithm implies that  $e$  is the unique cut of maximum cost in  $\mathcal{C}''$ . Since  $\text{Bud}_R(s)$  is locally constant while moving from  $u$  to  $v$ , from Algorithm 2, one concludes that the minimum cut of the tree  $T' + e$  endowed with the length function  $l - x_u$  has also cost  $c(e)$ . Hence,  $\text{Bud}_R$  is constant between  $u$  and the point  $p$  of  $e$  located at distance  $x_u(e)$  from  $u$  and then it increases (indeed, at this moment, the new maximum cut in  $\mathcal{C}''$  will have a cost strictly smaller than  $c(e)$ ). Hence,  $\text{Bud}_R(v) > \text{Bud}_R(u)$ , and the quasiconvexity of  $\text{Bud}_R$  yields  $\text{Bud}_R(p) \geq \text{Bud}_R(v)$  for every  $p \in \mathcal{T}''$ . This concludes Case 1.

**CASE 2.**  $x_u(e) = 0$ . Suppose, by way of contradiction, that  $\text{Bud}_R(p) < \text{Bud}_R(u)$  for some point  $p \in \mathcal{T}''$ . Since  $\text{Bud}_R$  is locally constant and convex on  $e$ , the point  $p$  cannot belong to  $e$ . So, assume that  $p$  belongs to an edge  $e' = u'v'$  distinct from  $e$ . Let  $d_l(u', v) < d_l(v', v)$ . One may suppose that the edge  $e = uv$  and the point  $p$  are selected so that the path  $P(u, v')$  between  $u$  and  $v'$  contains a minimum number of edges. Denote by  $e, e_1, e_2, \dots, e_k, e'$  the edges on this path, where  $e_1 = vv_1, e_2 = v_1v_2, \dots, e_k = v_kv'$ . Proposition 4 and the choice of  $e'$  imply that  $\text{Bud}_R$  is constant on all edges of  $P(u, v')$ , except the last edge  $e'$  (see Fig. 5).

From  $x_u$ , we derive an optimal solution  $x_v$  for  $v$  by repeatedly applying Algorithm 2. Let  $C_1''$  be the cut of

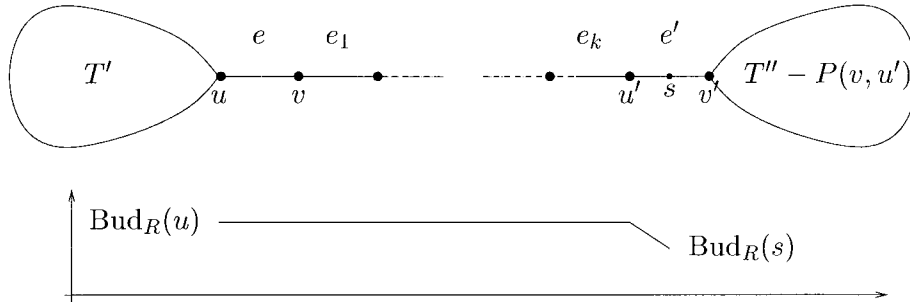


FIG. 5.

maximum cost of  $\mathcal{C}''$  (defined as in Case 1) and let  $C'_1$  be the cut of minimum cost in the tree  $T' + e$  with the length function  $l - x_u$ . Since  $\text{Bud}_R$  is constant on  $e$ , one has  $c(C'_1) = c(C''_1)$ . Moreover,  $c(e) > c(C''_1)$  holds; otherwise, according to the modification in Algorithm 1,  $e$  must belong to  $\text{supp}(x_u)$ . Applying  $\text{MOVE}_1$ , we construct a point  $u_1 \in e$  and an optimal solution  $x_{u_1}$ . By repeating the same procedure, we compute the points  $u_2, \dots, u_{m-1}, u_m = v$  using the cuts  $C'_2, \dots, C'_m$  and  $C''_2, \dots, C''_m$  defined in a similar way as  $C'_1$  and  $C''_1$  (note that  $c(C'_j) = c(C''_j)$  for every  $j = 1, \dots, m$ ). For each  $u_i$ , an optimal solution  $x_{u_i}$  is computed. Since  $e$  has cost larger than  $c(C''_1)$ , the edge  $e$  is never contracted, in particular,  $x_v(e) = x_{u_m}(e) = 0$ .

Now, we assert that  $x_v(e_1) = l(e_1)$ . Indeed, if  $0 < x_v(e_1) < l(e_1)$ , then  $\text{Bud}_R(s) > \text{Bud}_R(v) = \text{Bud}_R(u)$  by Case 1. On the other hand, if  $x_v(e_1) = 0$ , since  $\text{Bud}_R$  is constant on  $e_1$ , we get a contradiction with the minimality choice of  $e$  and  $s$ . Thus,  $x_v(e_1) = l(e_1)$ , whence  $x_v$  is an optimal solution for  $v_1$ . Applying the same argument to  $e_2$  instead of  $e_1$ , one concludes that  $x_v(e_2) = l(e_2)$ . Continuing this way, we deduce that all the edges  $e_1, e_2, \dots, e_k$  are fully contracted in  $x_v$  and that  $x_v$  is an optimal solution for each of the points  $v_1, \dots, v_k, u'$ .

Since  $\text{Bud}_R$  is convex on  $e'$ , when we move from  $u'$  to  $v'$ , the function  $\text{Bud}_R$  decreases. Hence, one can suppose that  $p$  and  $x_p$  are obtained from  $u'$  and  $x_v$  by running  $\text{MOVE}_1$ . Suppose that  $C'_{m+1}$  and  $C''_{m+1}$  are the cuts according to which the optimal solution  $x_p$  is computed. As  $\text{Bud}_R(u) = \text{Bud}_R(v) = \text{Bud}_R(u') > \text{Bud}_R(p)$ , we must have  $\text{supp}(x_v) \cap T' \neq \emptyset$  and  $\text{supp}(x_v) \cap (T'' - P(v, u')) \neq \emptyset$ ; in particular, the cuts  $C'_{m+1}$  and  $C''_{m+1}$  are nonempty. Since  $\text{Bud}_R(p) < \text{Bud}_R(u')$ , we have

$$c(C'_{m+1}) < c(C''_{m+1}). \quad (6)$$

The cut  $C'_{m+1}$  cannot contain  $e$ , because  $c(C'_{m+1}) \leq c(C''_{m+1}) < c(e)$ . Also, it is disjoint from  $e_1, \dots, e_k$ , because all these edges are fully contracted in  $x_v$ . Now, if  $C'_{m+1}$  coincides with the minimum cut  $C'$  of the tree  $T'$  with the length function  $l - x_v$ , then  $c(C') \geq c(C'_1) = c(C''_1) \geq c(C''_{m+1})$ , leading to a contradiction. Therefore,  $C'_{m+1} = \{e'\}$ . From (6) and Algorithm 1, one concludes that  $e'$  must be fully contracted in  $x_v$ , that is,  $x_v(e') = l(e')$ . But,

then, Algorithm 2 must choose  $e'$  as the cut  $C''_{m+1}$ , contrary to (6).  $\blacksquare$

After this, it is clear that, instead of exploring  $T$  level by level as we explained above, one can use a recursive procedure which will explore  $T$  in the depth-first-search manner (it has the same time complexity and its correctness follows from our previous discussion):

**Algorithm 3**  $\text{EXPLORE}(T, l, c, s, x)$

```

for each unexplored neighbor  $v$  of  $s$  do
   $(s', x') := \text{MOVE}_1(T, l, c, s, v, x)$ 
  if  $\text{Bud}_R(s') < \text{Bud}_R(s)$  then
     $(s, x) := \text{MOVE}(T, l, c, s', s, v, x')$ 
    if  $s \neq v$  then
      return  $s$ 
    else
      return  $\text{EXPLORE}(T, l, c, v, x)$ 
  else
    if  $\text{Bud}_R(s') = \text{Bud}_R(s)$  and  $x(sv) = l(sv)$  then
       $s'' := \text{EXPLORE}(T, l, c, v, x)$ 
      if  $\text{Bud}_R(s'') < \text{Bud}_R(s)$  then
        return  $s''$ 
enddo
return  $s$ 

```

It remains to examine the complexity of this algorithm. An optimal solution  $x_{s_0}$  can be computed in  $O(|V|^2)$  time by employing Algorithm 1. For each unexplored neighbor  $v$  of  $s$ , we run in linear time the algorithm  $\text{MOVE}_1$  on the edge  $sv$ ; therefore, the overall complexity of this operation is at most  $O(|V|^2)$ . To find the points  $s_1, \dots, s_m$ , we have to run  $\text{MOVE}$  on each of the edges  $e_0, \dots, e_{m-1}$ . Notice that the points  $s_0, s_1, s_2, \dots, s_m$  and the edges  $e_0, \dots, e_{m-1}$  all lie on a common path of  $T$ ; therefore, the total number of events is still bounded by  $|V| + |E|$  if one executes  $\text{MOVE}$  on all edges of this path. Hence, the total complexity of  $\text{MOVE}$  is again  $O(|V|^2)$ . Finally, while traversing fully contracted edges, we have to transfer those edges from  $\mathcal{C}''$  to  $\mathcal{C}'$ . Obviously, this can be done within the same time

bounds. Therefore, the overall complexity of Algorithm 3 is  $O(|V|^2)$ . The point  $s_m := s$  and the solution  $x_{s_m}$  returned by EXPLORE provide an optimal solution for the diameter lowering problem  $DLP(T, l, c, 2R)$ . To solve the second diameter lowering problem  $DLP^*(T, l, c, B)$ , one has to start from an optimal solution  $x_{s_0}$  of the problem  $ELP^*(T, l, c, s_0, B)$  computed by Algorithm 1', then run Algorithm 3, and for the point  $s_m$  and the solution  $x_{s_m}$  returned by this algorithm, again run Algorithm 1'. By Lemma 2, we will obtain a global minimum of the function  $\text{Rad}_B$ , therefore, an optimal solution for  $DLP(T, l, c, 2R)$ . Summarizing, here is the main contribution of this paper:

**Theorem.** *Each of the DLP and ELP problems on a tree  $T = (V, E)$  can be solved in  $O(|V|^2)$  time.*

## Acknowledgments

The authors would like to acknowledge the referees for suggestions and critical remarks which improved the readability and the presentation of the paper. This research has grown up from questions raised by the second author during the “Network Development Days” in Marseille and Würzburg.

## REFERENCES

- [1] K.U. Drangmeister, S.O. Krumke, M.V. Marathe, H. Nolte-meier, and S.S. Ravi, Modifying edges of a network to obtain short subgraphs, *Theor Comput Sci* 203 (1999), 91–121.
- [2] B.C. Tansel, R.L. Francis, and T.J. Lowe, Location on networks: A survey I, II, *Mgmt Sci* 29 (1983), 482–511.