ELSEVIER

Discrete Applied Mathematics 91 (1999) 93–117

DISCRETE APPLIED MATHEMATICS

Minimizing broadcast costs under edge reductions in tree networks

Susanne E. Hambrusch[a,*,1], Hyeong-Seok Lim[b,2]

[a] *Department of Computer Sciences, Purdue University, West Lafayette, IN 47907, USA*

[b] *Department of Computer Science, Chonnam National University, Kwangju, 500-757, South Korea*

Received 18 June 1997; received in revised form 21 May 1998; accepted June 1998

Abstract

We study the broadcasting of messages in tree networks under edge reductions. When an edge is reduced, its cost becomes zero. Edge reductions model the decrease or elimination of broadcasting costs between adjacent nodes in the network. Let T be an n-vertex tree and B be a target broadcast cost. We present an $O(n)$-time algorithm for determining the minimum number of edges of T to reduce so that a broadcast cost of B can be achieved. We present an $O(n \log n)$-time algorithm to determine the minimum number of edges to reduce so that a broadcast initiated at an arbitrary vertex of T costs at most B. Characterizations of where edge reductions are placed underly both algorithms and imply that reduced edges can be centrally located. © 1999 Elsevier Science B.V. All rights reserved.

Keywords: Analysis of algorithms; Message broadcasting; Blocking communication model; Tree networks

1. Introduction

The broadcasting of messages in a communication network is a fundamental operation in parallel and distributing systems. Whether a broadcasting strategy is efficient depends on the underlying communication model, in particular on how available communication links can be used. In the blocking communication model, a message is transmitted to only one adjacent processor at any time. Transmitting the message can be viewed as making a telephone call and as blocking the communication abilities for both the sender and receiver. The blocking communication model has been studied

* Corresponding author. E-mail: seh@cs.purdue.edu.

[1] Research supported in part by DARPA under contract DABT63-92-C-0022ONR. The views and conclusions contained in this paper are those of the authors and should not be interpreted as representing official policies, expressed or implied, of the US government.

[2] Research supported in part by Korea Research Foundation. Research was done while visiting Purdue University.

extensively from a theoretical point of view [6, 7, 9, 16] and it underlies numerous broadcast algorithms [1, 2, 10–12].

In this paper we study broadcasting in tree networks under edge reductions for blocking communication. Edge reductions model the decrease or elimination of broadcasting costs between adjacent nodes of the network. For example, when a communication link is replaced by a faster link, the communication time reduces and may become negligible compared to other costs. We study where to place such edge reductions so that broadcasting costs are minimized. Edge reductions in longest path computations have previously been studied in [8] and have application in circuit layout and project management [3, 5]. Related work on network upgrading through changes on vertex or edge weights can be found in [14, 15].

Let $T=(V,E)$ be an n-vertex tree with unary edge weights. When vertex i sends a message to vertex j, i and j are engaged in communication for one time unit. Then, vertices i and j are free to continue broadcasting. Assume that some vertex s broadcasts a message to all vertices of T. Let B_s be the minimum cost of a broadcast initiated by s (i.e., the time at which the message arrives at the last vertex). Clearly, the order in which messages are sent out from a vertex to its adjacent vertices is crucial when determining B_s. Further, the cost of a broadcast initiated at vertex s can differ from that of a broadcast initiated at some other vertex. In problem *Br_Min* we determine, for a given tree T, the set of vertices from which a broadcast of minimum time can be initiated. Let $B_{Min}(T)$ be this broadcast time; i.e., $B_{Min}(T)=\min_{1\leqslant s\leqslant n}\{B_s\}$. In problem *Br_Arb* we bound the cost of a broadcast initiated at an arbitrary vertex of T; i.e., $B_{Arb}(T)=\max_{1\leqslant s\leqslant n}\{B_s\}$. In [16], Slater et al. present an O(n)-time algorithm for determining $B_{Min}(T)$. The vertices at which initiating a broadcast costs $B_{Min}(T)$ represent the center set and they form a star consisting of at least 2 vertices. In [16] it is also shown that the value of $B_{Arb}(T)$ is the sum of $B_{Min}(T)$ and the longest path length to a vertex in the center set.

Assume we are given a tree T and a target broadcast cost B. In problem *Br_Min under edge reductions* we determine which edges to reduce so that the resulting tree T_R has a broadcast cost of at most B (i.e., $B_{Min}(T_R)\leqslant B<B_{Min}(T)$) and the number of reduced edges is a minimum. In *Br_Arb under edge reductions* we determine a reduction so that $B_{Arb}(T_R)\leqslant B<B_{Arb}(T)$ and the number of reduced edges is a minimum. In this paper we present an O(n)-time algorithm for *Br_Min* and an O($n\log n$)-time algorithm for *Br_Arb* under edge reductions. We show that for both problems there exists an optimal reduction in which the reduced edges form a tree. This implies that when edge reductions correspond to fast communication links, such links can be centrally located.

The paper is organized as follows. In Section 2 we review the O(n)-time algorithm for determining $B_{Min}(T)$ described in [16]. Our edge reduction algorithms make use of this algorithm in a preprocessing step and work with the broadcast entries it generates. In Section 3 we first develop characterizations of where the edge reductions are placed in an optimal reduction for *Br_Min*. We then present the O(n)-time algorithm. Problem *Br_Arb* is discussed in Section 4. Since the final broadcast cost is now the sum of a longest path length and the cost of a *Br_Min* instance, different characterizations are established in Section 4.1. The edge reduction algorithm for problem *Br_Arb* is presented in Section 4.2. The data structures achieving the $O(n \log n)$ time bound are described in Section 4.3. Section 5 concludes.

2. Review of the $O(n)$ algorithm for *Br_Min*

The algorithm described in [16] solves problem *Br_Min* in $O(n)$ time by performing a sweep through T which starts at the leaves. Our algorithms for *Br_Min* and *Br_Arb* under edge reductions use this algorithm as a preprocessing step and use the entries it generates. For completeness sake, we describe the algorithm given in [16]. In order to avoid confusion, we refer to this algorithm as Algorithm *CF* (*CF* for center finding). Algorithm *CF* generates the following:

1. Algorithm *CF* roots tree T at some vertex, say vertex c. For every vertex u, let $p(u)$ be the parent of u.
2. Algorithm *CF* determines center set C, $C \subseteq V$. For every vertex $u \in C$, a broadcast initiated at u has cost $B_{Min}(T)$.
3. For every vertex u, Algorithm *CF* determines a broadcast entry $b(u)$; $b(u)$ represents the cost of a broadcast initiated at vertex u to the vertices in the subtree rooted at u. The b-values determine the order in which vertex u sends the message to its children.

Algorithm *CF* starts by initializing the b-entries so that every leaf u has $b(u) = 0$ and every other vertex has $b(u) = \infty$. A vertex is either marked or unmarked. A marked vertex has its final b-entry and its parent in the rooted version of T is known. Initially, every leaf is marked and its parent is the other vertex on the incident edge. Every other vertex is unmarked.

Let $deg(u)$ be the degree of vertex u. Next, Algorithm *CF* considers all vertices u adjacent to $deg(u) - 1$ leaves. For every such vertex u we invoke procedure *Determine_b*(u) given in Fig. 1. Vertex u then has a b-entry, but it has not yet been marked (and no parent vertex has been recorded). Once the leaves and possibly their adjacent vertices have been processed, the main iteration of Algorithm *CF* begins. In each step of this iteration, Algorithm *CF* selects an unmarked vertex with minimum b-value (ties are broken by using the vertex index). Let vertex u be the unmarked vertex chosen. Let x be the one unmarked vertex adjacent to u. Vertex u is marked and x is made the parent of u in the rooted tree. Then,

- If vertex x is the last unmarked vertex, x is the root of tree T. We invoke *Determine_b*(x) to determine the final value of $b(x)$. Note that $b(x)$ received a value in a previous iteration which now increases. Algorithm *CF* then terminates.
- If x is adjacent to $deg(x) - 1$ marked vertices, invoke *Determine_b*(x). Vertex x thus receives a b-value.
- If x is adjacent to more than 1 unmarked vertex, no action is taken.

```
Procedure Determine_b(u)

Input: vertex $u$ which has $u_1, u_2, \dots, u_q$ as the marked adjacent vertices.
Output: $b(u)$

(1) Arrange vertices $u_1, u_2, \dots, u_q$ so that $b(u_1) \geqslant b(u_2) \geqslant \cdots \geqslant b(u_q)$
(2) $b(u) = \max_{1 \leqslant i \leqslant q}\{b(u_i) + i\}$
```

Fig. 1. Procedure *Determine_b*(u).

Let c be the vertex processed last (i.e., the last unmarked vertex) with children $c_1, \dots, c_q$. Let s be the smallest index such that $b(c_s) + s = b(c)$, $1 \leqslant s \leqslant q$. Initiating a broadcast at $c_1, \dots, c_s$ or c results in a broadcast cost of $b(c)$ and $C = \{c, c_1, \dots, c_s\}$. We refer to [16] for the details on how to implement Algorithm *CF* in $O(n)$ time.

Vertex c is the root of the rooted version of tree T. Throughout the paper, if not stated otherwise, we assume that T is rooted at c. We further assume that the children of each vertex u are arranged by non-increasing b-values. We illustrate the last iteration of Algorithm *CF* using tree T shown in Fig. 4(a). The integers next to the vertices represent the corresponding b-values. The shaded vertices represent the vertices in center set C. When vertex c_1 is selected and marked, it has $b(c_1) = 5$ and vertex c is unmarked with $b(c) = 6$. Every other vertex is already marked. After c_1 is marked, we have $x = c$. Procedure *Determine_b*(u) sets $b(c) = 7$ and Algorithm *CF* terminates.

Algorithm *CF* will be applied to the initial tree T as well as to trees T_R with edge reductions. We conclude this section with a brief description of the changes of Algorithm *CF* for a tree with 0/1 weights. Step (2) in Algorithm *Determine_b*(u) changes. Let n_i be the number of unreduced edges among (u_j, u), $1 \leqslant j \leqslant i$. We set

$$b(u) = \max_{1 \leqslant i \leqslant q} \{b(u_i) + n_i\}.$$

Maintaining the $O(n)$ time is straightforward. When edge (u, u_i) is reduced, a message at vertex u at time t is also available at time t at u_i. When (u, u_i) is not reduced, the message arrives at u_i at time $t + n_i$. The quantity $b(u_i) + n_i$ for a reduced edge (u, u_i), $i > 1$, cannot result in the smallest index inducing the maximum. This holds since there exists a j, $j < i$, with $b(u_j) \geqslant b(u_i)$ and $n_j = n_i$. Hence, $b(u)$ is determined correctly.

3. *Br_Min* under edge reductions

In problem *Br_Min* we determine (i) the set of vertices at which initiating a broadcast costs minimum time and (ii) the order in which each vertex sends out the message to adjacent vertices. Algorithm *CF* described in the previous section solves *Br_Min* in $O(n)$ time. In problem *Br_Min* under edge reductions we further determine an optimal reduction R containing the edges to be reduced. Let T_R be the tree obtained from T when every edge in R has weight 0. As already stated, reduction R is an optimal

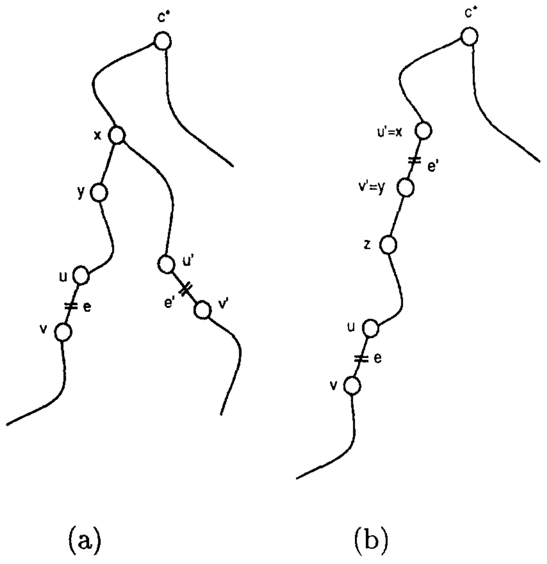

Fig. 2. Positions of e and e' when E^* does not form a tree ('=' indicates a reduced edge).

reduction when $B_{Min}(T_R) \leqslant B$ and the number of edges in R is a minimum. In this section we describe Algorithm *Reduce_Br_Min* which determines such an optimal reduction in $O(n)$ time. When it is clear from the context, we refer to problem *Br_Min* under edge reductions simply as problem *Br_Min*.

Assume Algorithm *CF* has been applied to tree T. We assume throughout that c is the last vertex processed by Algorithm *CF* and that set C contains the vertices in the center set. Furthermore, $p(u)$ denotes the parent of u when T is rooted at c and the children of u are arranged by non-increasing b-values. Optimal reductions are not unique. In Lemmas 1–3 we show that among all optimal reductions there exists one which satisfies the characterization given in Lemma 3; i.e., there exists an optimal reduction so that the reduced edges form a tree and vertex c is incident to a reduced edge. This characterization is the basis for our $O(n)$-time algorithm.

Lemma 1. *There exists an optimal reduction R^* so that the edges in R^* form a tree.*

Proof. Assume the edges in R^* do not form a tree, but a forest F^*. Then, there exist two edges $e=(u,v)$ and $e'=(u',v')$ in R^* such that there is no path between them in F^*. We choose e and e' so that no edge on the path between them is in R^*.

Let c^* be the last vertex processed when Algorithm *CF* is applied to tree T_{R^*}. Let x be the lowest common ancestor of edges e and e' when is T rooted at c^*. Let u (resp. u') be the vertex of edge e (resp. e') closer to x. Either u or u' can be equal to x, but not both. W.l.o.g., assume $u \neq x$ and let y be the child of x on the path from x to u. For the situation shown in Fig. 2(a), let $R' = R^* - \{(u,v)\} \cup \{(x,y)\}$. The situation for

e and e' can also be as shown in Fig. 2(b); i.e., $x = u'$ and $v' = y$. In this case, let z be the vertex adjacent to y on the path from y to u and let $R' - \{(u,v)\} \cup \{(y,z)\}$. R' and R^* reduce the same number of edges and $B_{Min}(T_{R^*}) = B_{Min}(T_{R'})$. Reduction R' is thus also an optimal reduction. Compared to R^*, one reduction is performed closer to the root c^*. If the edges in R' form a tree, we have obtained the desired reductions. Otherwise, apply the edge reduction swap again. This process will eventually produce a desired reduction. □

The next two lemmas show that there exists an optimal reduction R^* forming a tree for which vertex c is incident to a reduced edge.

Lemma 2. *There exists an optimal reduction R^* so that R^* forms a tree and c^* is incident to an edge in R^*.*

Proof. Assume that the edges in R^* are connected, but that no edge is incident to c^*. Let c_i^* be the vertex incident to c^* so that all edges in R^* are in the subtree rooted at c_i^*. Let u be the vertex closest to c_i^* incident to a reduced edge. Let $p^*(u)$ be the parent of u in tree T_{R^*} with root c^*. By reducing edge $(u, p^*(u))$ and not reducing one of the edges incident to u in R^*, we obtain a reduction R' with $B_{Min}(T_{R^*}) \geqslant B_{Min}(T_{R'})$. Making this change may have caused the edges in R' to get disconnected. However, by applying Lemma 1 to R', we obtain another optimal reduction containing edge $(u, p^*(u))$ and R' connected. We repeat this process of trading edges until an edge incident to c^* is reduced. □

Lemma 3. *Let R^* be an optimal reduction so that the edges in R^* form a tree and vertex c^* is incident to an edge in R^*. Then, vertex c is incident to an edge in R^*.*

Proof. Let $c_1, \ldots, c_q$ be the children of vertex c in T and let T/c_j be the tree obtained from T when the subtree rooted at c_j is deleted, $1 \leqslant j \leqslant q$. We use $B_c(T/c_j)$ to denote the cost of broadcast in T/c_j initiated at c. We make use of the following two inequalities which hold for every j, $1 \leqslant j \leqslant q$:

$$b(c_j) \leqslant B_c(T/c_j), \tag{1}$$

$$B_c(T/c_j) \geqslant B_{Ft}(T) - 1. \tag{2}$$

The first equation holds since $b(c_j) > B_c(T/c_j)$ would imply that c is not the last vertex processed when Algorithm *CF* is applied to T. To show (2), assume there exists a c_j such that $B_c(T/c_j) < B_{Ft}(T) - 1$. Consider the following broadcast in T initiated at c_j: vertex c_j sends the message to c; c broadcasts the message to all the other vertices in subtree T/c_j; simultaneously, c_j broadcasts the message to the vertices in the subtree rooted at c_j. Let $B_{c_j}(T)$ be the cost of this broadcast. Then, $B_{c_j}(T) \leqslant \max\{1 + B_c(T/c_j), 1 + b(c_j)\}$. Using (1) and the assumption $1 + B_c(T/c_j) < B_{Ft}(T)$, we get $B_{c_j}(T) \leqslant 1 + B_c(T/c_j) < B_{Ft}(T)$. This contradicts that a broadcast initiated at c has minimum cost and (2) follows.

```
Algorithm Reduce_Br_Min

(1) execute Algorithm CF on tree $T$;
    vertex $c$ is the root of the rooted tree $T$
    every vertex $u$ receives its broadcast value $b(u)$;

(2) for every vertex $u$ of $T$ do
      let $v_1, v_2, \ldots, v_q$ be the children of $u$ with $b(v_1) \geqslant b(v_2) \geqslant \cdots \geqslant b(v_q)$

      (2.1) let $l$ be the largest index such that $b(v_i) \geqslant B$
            make forced reducctions: reduce edges $(v_i, u)$, $1 \leqslant i \leqslant l$

      (2.2) $m = \max\{b(v_{l+1}) + 1, b(v_{l+2}) + 2, \ldots, b(v_q) + q - l\}$
            if $m > B$ then
               make $m - B$ choice reductions:
               reduce edges $(v_{l+1}, u), (v_{l+2}, u), \ldots, (v_{l+m-B}, u)$
    endfor
```

Fig. 3. Algorithm *Reduce_Br_Min*.

Assume that in reduction R^* vertex c is not incident to a reduced edge. Let c_i be the child of c in T so that all edges in R^* are in the subtree rooted at c_i ($c_i = c^*$ is possible). A broadcast initiated at c^* reaches vertex c via c_i and then continues in tree T/c_i. At least one time step is needed to broadcast from c^* to c and thus $b^*(c^*) \geqslant 1 + B_c(T/c_i)$. Making use of (2) gives $b^*(c^*) \geqslant B_{Ft}(T)$. Since $b^*(c^*) = B_{Min}(T_{R^*}) \leqslant B$ by definition of *Br_Min*, we get $B \geqslant B_{Min}(T)$, a contradiction. □

We are now ready to describe Algorithm *Reduce_Br_Min* which is given in Fig. 3. As already stated, the first step is to invoke Algorithm *CF* on tree T. Reduction R is determined by considering every vertex u and determining which of u's incoming edges are reduced. We distinguish between two types of reductions. A forced reduction is placed on every edge (v_i, u) with $b(v_i) \geqslant B$. Choice reductions are placed on β edges to achieve a broadcast cost of B for vertex u. Algorithm *Reduce_Br_Min* chooses the leftmost β edges for the choice reductions. We point out that for the problem *Br_Arb* we need different choice reductions when underlying instances of *Br_Min* are solved.

From the way Algorithm *Reduce_Br_Min* places reductions, it follows that the reduced edges form a tree containing vertex c and that the broadcast cost is B. We next describe the center set C_R for tree T_R. When Algorithm *Reduce_Br_Min* reduces edge (u, v), a broadcast initiated at either vertex u or v results in a broadcast cost of B. Hence, vertices u and v are in C_R. A vertex $c_i \in C$ not incident to a reduced edge is not necessarily in C_R. Fig. 4(a) shows such an example. We have $C = \{c, c_1, c_2, c_3, c_4\}$, $C_R = \{c, c_1, c_2, c_3, c_5\}$, and c_4 is in C, but c_4 is not in C_R. On the other hand, for the trees shown in Fig. 4(b) we have $C = C_R = \{c, c_1, c_2, c_3, c_4, c_5\}$ and C_R contains vertices not incident to reduced edges.

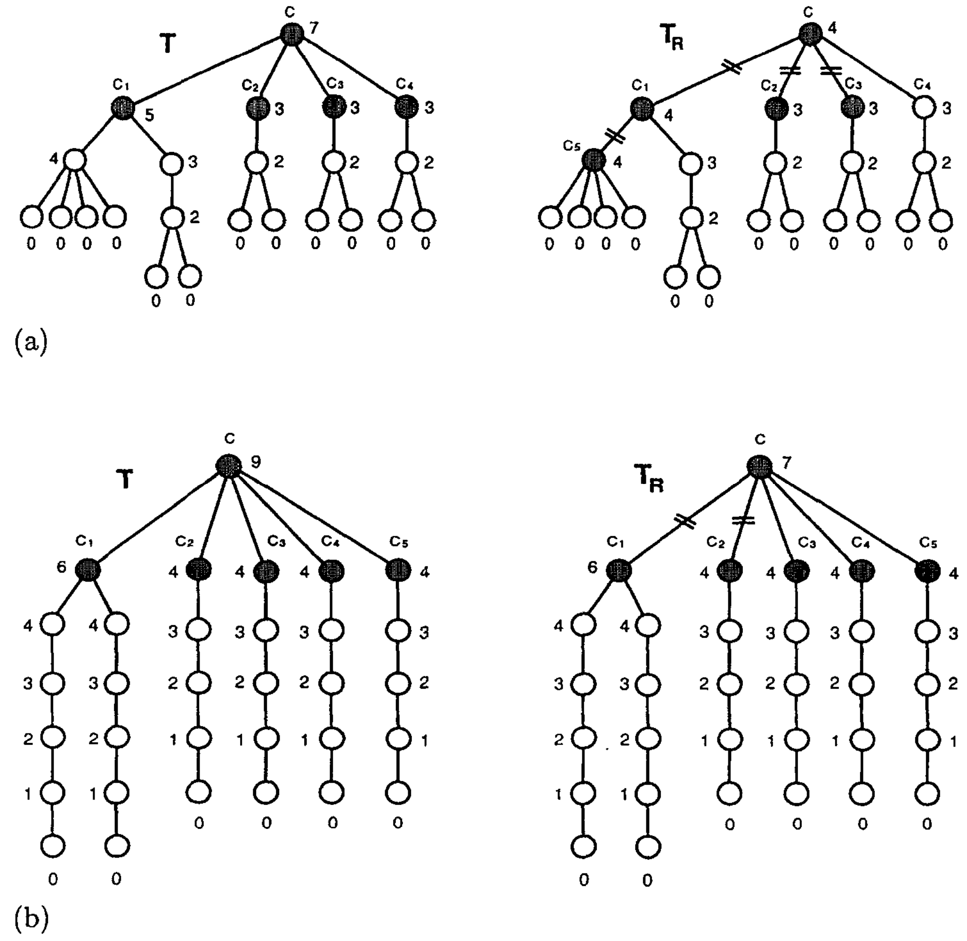

Fig. 4. Simple and extended center sets; vertices in center set are shaded. (a) $B_{Min}(T)=7$ and $B=4$; reduction R with a simple center set in which not every vertex in C is in C_R. (b) $B_{Min}(T)=11$ and $B=8$; reduction R with an extended center set.

We say that C_R is a *simple center set* if it contains only vertices incident to reduced edges and it is an *extended center set* if it contains vertices not incident to reduced edges. Center set C_R is a simple center set if at least two edges need to be reduced to achieve broadcast cost $B-1$. This is equivalent to the existence of at least two vertices in T_R which experience a broadcast cost of B. Fig. 4(a) shows a tree T_R with a simple center set.

In an extended center set, the vertices in C_R not incident to a reduced edge have a common parent which we call the *extended center vertex* x_e. Fig 4(b) shows an extended center set with $x_e=c$. When T_R contains an extended center vertex x_e having a child x_i such that edge (x_e,x_i) is not reduced and $x_i \in C_R$, reducing (x_e,x_i) achieves a broadcast cost of $B-1$. Hence, in an extended center set the reduction of a single edge decreases the broadcast cost. Extended center sets play only a minor role in *Br_Min*, but they have considerable impact on the algorithm for *Br_Arb*. From the way reductions are made by Algorithm *Reduce_Br_Min*, it follows that when C_R is an extended center set, we have $x_e = c$ and all reductions are on edges incident to c.

We conclude this section with the optimality of the reduction generated by Algorithm *Reduce_Br_Min*. Let b_R be the resulting broadcast values obtained when Algorithm *CF* is applied to tree T_R. Observe that vertex c is not necessarily the last vertex processed. However, there exists an order of choosing vertices with the same b_R-value so that c is the last vertex processed. Hence, w.l.o.g. we can assume that c is the last vertex processed.

Theorem 4. *Algorithm Reduce_Br_Min solves problem Br_Min under edge reductions in* $O(n)$ *time.*

Proof. Let R be the reduction generated by Algorithm *Reduce_Br_Min*. It is clear that $B_{Min}(T_R) = B$ and that the edges in R form a tree. Let R^* be an optimal reduction. Using the previous lemmas, we can assume that the edges in R^* form a tree and that vertex c is processed last when running Algorithm *CF* on T_{R^*}. We show that there exists an R^* with $R^* = R$.

Consider a vertex v and assume that R and R^* reduce the same edges on the path from c to v. Let $u_1, u_2, \ldots, u_k$ be the children of v with $b(u_1) \geqslant \cdots \geqslant b(u_k)$. Assume that R reduces edges $(v,u_1), (v,u_2), \ldots, (v,u_l)$. If R^* also reduces these l edges and no more, we are done handling vertex v. Otherwise, we proceed as follows.

Assume first that there exists a vertex u_r such that edge (v,u_r) is reduced in R, but not in R^*, $r \leqslant l$. Choose r to be as small as possible. Then, edges $(u,v_1), \ldots, (u,v_{r-1})$ are reduced in R as well as R^* and $b_R(u_r) \leqslant b^*(u_r) = b(u_r)$. The equality holds since no further edges are reduced in the subtree rooted at u_r in R^*. Since R^* is an optimal solution, we have $b(u_r) < B$. Edge (v,u_r) thus has a choice reduction in R (i.e., it is reduced to decrease the delay of the message arriving at the nodes $u_r, u_{r+1}, \ldots, u_l$). Hence, there exists a β, $\beta \geqslant 0$, so that $b(u_{r-\beta}) + \beta + 1 > B$.

If R^* contains an edge $(v,u_{r+\alpha})$, $\alpha \geqslant 1$, which is reduced, we change R^* so that edge (v,u_r) is reduced and edge $(v,u_{r+\alpha})$ is not. This swap does not change the broadcasting cost and thus results in another optimal solution. If the new R^* does not form a tree, we apply Lemma 1. If there exists no edge $(v,u_{r+\alpha})$, $\alpha \geqslant 1$, which is reduced in R^*, R^* would not be an optimal solution. This follows since there exists a β such that $b(u_{r+\beta}) + \beta + 1 > B$. We can thus assume that there exists an optimal solution which also reduces edge (v,u_r).

Finally, consider the situation when R does not reduce edge (v,u_r), but R^* does. Observe that R and R^* agree on their action of the first $r-1$ edges incident to v. Not reducing the edges (v,u_j), $r \leqslant j \leqslant k$, or any edges in the subtrees rooted at these vertices does not cause tree T_{R^*} to exceed the target broadcast cost. It thus follows that R^* does not need to make a reduction on edge (v,u_r). Since R^* is optimal solution, R^* will not make such a reduction. It follows that there exists an $R^* = R$ and thus R

is an optimal reduction. The O(n)-time bound for generating R follows from the O(n) time bound of the Algorithm *CF* and how edge reduction are determined. □

4. *Br_Arb* under edge reductions

In problem *Br_Arb* under edge reductions we determine, for a given tree T and a target broadcast cost B, a reduction R so that (i) a broadcast initiated at an arbitrary vertex is completed by time B in T_R and (ii) the number of reduced edges is a minimum. Recall the cost of a tree T in *Br_Arb*: $B_{Arb}(T) = \max_{1 \leqslant i \leqslant n} \{B_i\}$, where B_i is the cost of completing a broadcast initiated at vertex i. Slater et al. show in [16] that for trees with unary weights $B_i = d(i,c_i) + B_{Min}(T)$, where $d(i,c_i)$ is the length of the path from i to c_i and c_i is the vertex in center set C closest to vertex i. Let $d_m = \max_{1 \leqslant i \leqslant n} d(i,c_i)$. It then follows that $B_{Arb}(T) = d_m + B_{Min}(T)$.

In *Br_Arb* under edge reductions the decision on where to place edge reductions is thus determined by the cost of a *Br_Min* instance and a longest path length. In Section 4.1 we present characterizations of an optimal reduction which allow us to efficiently identify one optimal reduction among all possible optimal reductions. In particular, we show that we can again assume that the reduced edges form a tree containing vertex c and that the cost of an *Br_Arb* instance under edge reductions is the sum of a longest path length and the cost of an *Br_Min* instance. Making use of these characterizations leads to a solution for *Br_Arb* which considers all possible distances and solves a *Br_Min* instance for each distance. If Algorithm *Reduce_Br_Min* were invoked each time, O(n^2) time would follow. In Section 4.2 we describe Algorithm *Reduce_Br_Arb* which achieves the claimed O($n \log n$) time bound. Algorithm *Reduce_Br_Arb* also considers all possible distances, but avoids recomputations and makes updates on existing reductions. In order to make these updates fast, we introduce entries defined on the edges and we change how Algorithm *Reduce_Br_Min* selects edges for choice reductions.

4.1. Characterizations for Br_Arb

The next two lemmas show that there exists an optimal reduction in which reductions are made as characterized in Lemma 3; i.e., the reduced edges form a tree containing vertex c. Let R^* be an optimal reduction and let c^* be the last vertex processed when Algorithm *CF* is applied to tree T_{R^*}.

Lemma 5. *There exists an optimal reduction R^* so that the edges in R^* form a tree and c^* is incident to a reduced edge.*

Proof. The reduction trading operations described in the proofs of Lemmas 1 and 2 do not increase the broadcast cost. We now show that reduction trading does not increase the cost of the *Br_Arb* instance. Assume R^* does not satisfy the lemma and let R' be the reduction in which a reduction on edge $(u, p^*(u))$ is removed and a reduction is placed on edge $(v, p^*(v))$, where v is an ancestor of u in T_{R^*}. From the way edges are chosen in Lemmas 1 and 2, it follows that the path from $p^*(u)$ to v does not contain a reduced edge.

When vertex v is not in the center set of T_{R^*}, neither the longest path length to a center vertex nor the broadcast cost increases. Hence, the cost of reduction R' is not larger than that of R^*.

When vertex v is in the center set, v is a child of c^*. Placing the reduction on edge (v, c^*) can increase the longest path length to a vertex in the center set. It can also result in vertex c^* no longer being the last vertex processed when applying Algorithm *CF*. More specifically:

- The longest path length can increase by 1. This can happen when the size of the center set decreases and a vertex in C_{R^*} is no longer in $C_{R'}$. Or, it can happen when the length of the path from a vertex in the subtree rooted at u to vertex v increases by 1 (since edge $(u, p^*(u))$ is no longer reduced).
- Since edge $(u, p^*(u))$ looses its reduction, it is possible that $b'(v) > b^*(v)$. This can result in vertex v being the new root of tree $T_{R'}$.

We next show that neither of these two events increases the cost of reduction R'. If $b'(v) = b^*(v)$, we have $B_{Min}(T_{R'}) = B_{Min}(T_{R^*}) - 1$ and c^* remains the root of the tree. Any increase in the longest path length is thus offset by the decrease in the cost of the underlying *Br_Min* instance. Hence, the cost does not increase for *Br_Min*.

Assume $b'(v) = b^*(v) + 1$. Since edge (v, c^*) is not reduced in R^*, we have $b^*(v) < b^*(c^*)$. When $b^*(v) \leqslant b^*(c^*) - 2$, vertex c^* remains the root and a broadcast initiated at c^* is completed by time $b^*(c^*) - 1$; i.e., $B_{Min}(T_{R'}) = B_{Min}(T_{R^*}) - 1$. The broadcast cost decreases by 1 and thus offsets any increase in the longest path length. The cost of $B_{Arb}(T_{R'})$ does not increase. The interesting case is $b^*(v) = b^*(c^*) - 1$. Center set C_{R^*} contains c^* and v and v is the only vertex in C_{R^*} not incident to a reduced edge. Let $B_{c^*}(T_{R^*}/v)$ be the cost of a broadcast initiated at c^* in the tree obtained from T_{R^*} when vertex v and the subtree rooted at v are deleted. We have $B_{c^*}(T_{R^*}/v) = b^*(c^*) - 1 = b'(c^*)$. Since we assume $b^*(v) = b^*(c^*) - 1$, we have $b'(v) = b^*(c^*)$. When Algorithm *CF* is applied to $T_{R'}$, vertex v is processed last and is made the root. However, the broadcast cost does not change; i.e., $B_{Min}(T_{R'}) = B_{Min}(T_{R^*})$. Fig. 5 shows trees T_{R^*} and $T_{R'}$ for which this happens. For clarity, the edges are directed towards the corresponding roots. Observe that longest path length to a vertex in the center set is 3 for both trees. We next show that the longest path length cannot increase in $T_{R'}$.

Center set $C_{R'}$ contains v (since it is the root) and c^* (since edge (v, c^*) is reduced). It also contains children of vertex v in $T_{R'}$. In particular, let w be the child of v so that the subtree rooted at w contains u (recall that edge $(u, p^*(u))$ lost its reduction in the reduction trading). Since w induces cost $B_{Min}(T_{R'}) = b'(v) = b^*(c^*)$, w is in $C_{R'}$. In Fig. 5(b), we have $w = v_3$. Consider now a path from some vertex x to v containing edge $(u, p^*(u))$. The length of this path increases by 1 in $T_{R'}$ compared to T_{R^*}. However, vertex w is now in the center set and the length of x to the closest

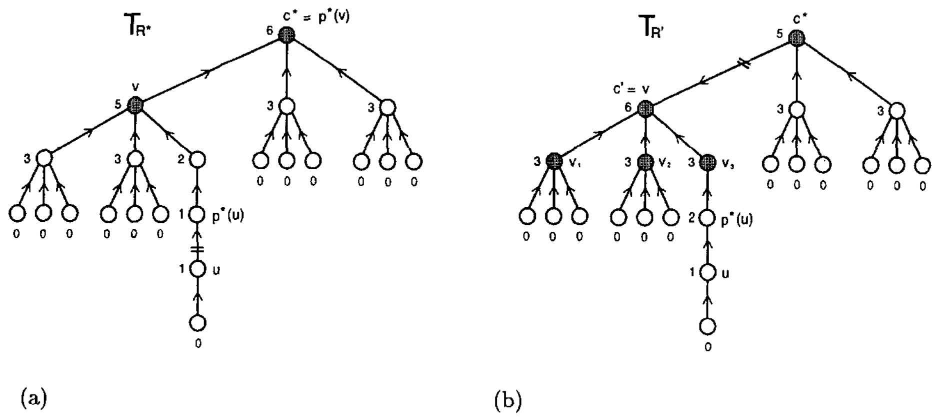

Fig. 5. Example for v becoming the new root when $b'(v) = b^*(v) + 1$ and $b^*(v) = b^*(c^*) - 1$. (a) $C_{R^*} = \{v, c^*\}$, $B_{Min}(T_{R^*}) = 6$ (b) $C_{R'} = \{v, c^*, v_1, v_2, v_3\}$, $B_{Min}(T_{R'}) = 6$.

vertex in the center set does not increase. Since C_{R^*} contains no vertices incident to unreduced edges besides v and c^*, and these vertices are in $C_{R'}$, the length of the distance from any other vertex to a vertex in the center set does not increase. Hence, $B_{Arb}(T_{R'}) = B_{Arb}(T_{R^*})$ and the lemma follows. □

Lemma 6. *Let R^* be an optimal reduction forming a tree so that vertex c^* is incident to a reduced edge. Then, vertex c is incident to a reduced edge.*

Proof. Assume that in reduction R^* vertex c is not incident to a reduced edge. Let c_i be the child of c in T so that all edges in R^* are in the subtree rooted at c_i. Recall the inequalities from Lemma 3: $B_c(T/c_i) \geqslant B_{Min}(T) - 1$ and $b^*(c^*) \geqslant B_{Min}(T)$. While the second inequality led to a contradiction for problem *Br_Min*, it does not do so for *Br_Arb*. The cost of reduction R^*, which is at most B, is now the sum of two quantities: a longest path length and $B_{Min}(T_{R^*})$. It is conceivable that an optimal reduction increases the broadcast cost for the sake of reducing the distance to the vertices in the center set.

To show that vertex c is incident to a reduced edge we show that, when Algorithm *CF* is applied to tree T_{R^*}, vertices can be marked so that vertex c^* is not the root, but a vertex on the path from c to $p(c^*)$ is. We can then apply Lemma 5 to obtain a reduction containing an edge incident to the new root. By repeating this argument we eventually generate a reduction satisfying the lemma.

Using the above inequalities we have $B_{Min}(T_{R^*}) = b^*(c^*) \geqslant B_{Min}(T)$ and $B_c(T/c_i) = b^*(c) \geqslant B_{Min}(T) - 1$. Let $d_1, d_2, \ldots$ be the children of c^* in T_{R^*} arranged by non-increasing b^*-values. Consider a child d_j not on the path from c to c^*. A broadcast initiated at vertex d_j in tree T costs at most $B_{Min}(T) - 2$. The broadcast cost for d_j can only decrease in R^* and thus $b^*(d_j) \leqslant B_{Min}(T) - 2$. When Algorithm *CF* is applied to tree T_{R^*}, vertex c is marked before c^*. Since $b^*(d_j) \leqslant B_{Min}(T) - 2$ for all but one child of c^*, c^* already has a b^*-value at the time vertex c is marked. Vertex c^* is not chosen

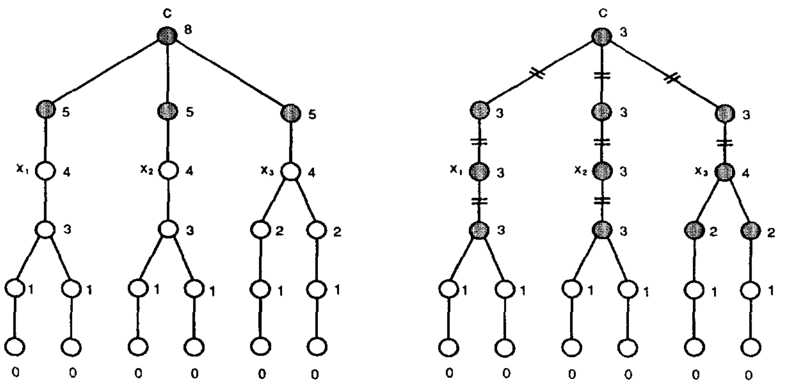

Fig. 6. Tree with $B_{Min}(T)=8$ and an extended center set for $B=6$ and $d=2$.

and thus c^*'s b^*-value at that time is at least $B_{Min}(T)-1$. Since $b(c^*)\leqslant B_{Min}(T)-1$ (otherwise the broadcast cost for tree T would be larger than $B_{Min}(T)$), it follows that c^*'s and c's b^*-value at the time c is marked is $B_{Min}(T)-1$. Hence Algorithm *CF* can mark vertex c^* instead vertex c. This implies that the root of the resulting tree is a vertex on the path from c to $p(c^*)$, not vertex c^*. Let c' be the new root. If $c=c'$, we use Lemma 6 to reposition reductions so that an edge incident to c is reduced. If $c\neq c'$, we apply the argument again with $c^*=c'$. $\square$

Let R^* be a reduction satisfying the characterization of Lemma 6. Let C^* be the center set for tree T_{R^*} and let d^* be the length of the longest path from a vertex in T_{R^*} to a vertex in C^*. In *Br_Arb*, edges receive reductions for distance as well as broadcast reasons. Arbitrary vertices can now be extended center vertices. For example, Fig. 6(a) shows a tree T with $B_{Min}(T)=8$. Fig. 6(b) shows the optimum solution for $B=6$ and $d=2$. Eight edges are reduced, all for the purpose of achieving distance at most 2 to a vertex in the center set. Vertex x_3 is the extended center vertex and the two children of x_3 are in the center set. Vertex x_3 is the last vertex processed when applying Algorithm *CF*. It is the only vertex experiencing a broadcast cost of 4.

Let $b^*(u)$ be the broadcast value for vertex u when Algorithm *CF* is applied to tree T_{R^*} and let $B_{Min}(T_{R^*})=B^*$. As already indicated, vertex c is not necessarily the last vertex processed and $b^*(c)<B^*$ is possible. The following lemma gives a characterization of an optimal solution which forms the basis of our algorithm.

Lemma 7. *Let R^* be an optimal reduction. Then, $B_{Arb}(T_{R^*})=d^*+B^*=B$.*

Proof. Let x and y be two vertices such that $y\in C^*$ and the distance from x to y in T_{R^*} is d^*. Assume first that C^* is a simple center set. In this case, c is the vertex processed last. When C^* is a simple center set there exists either a vertex $w\in C^*$ such that w has two children w_1 and w_2 with $b^*(w_1)=b^*(w_2)=B^*$, or all vertices with a broadcast cost of B^* form a path P starting at vertex c and ending at some vertex w.

In the first case, a broadcast initiated at x costs $d^* + B^*$. In the second case, vertex c has a child not on path P also inducing a broadcast cost of B^* for c. Independent of the position of vertex x, a broadcast initiated at x costs time $d^* + B^*$.

Assume now that C^* is an extended center with vertex x_e as the extended center vertex. Vertex x_e is now the last vertex processed. Let $x_1, x_2, \ldots$ be the children of x_e, in non-increasing order of their b^*-values. Let x_i be the smallest-indexed child of x_e inducing $b^*(x_e) = B^*$. Observe that edges incident to x_e may be reduced (because edges can be on long paths). Vertices $x_1, \ldots, x_i$ are in C^*. If the path from x to y does not lie in the subtree rooted at x_e, a broadcast initiated at x costs $d^* + B^*$. Assume thus that the path from x to y is in the subtree rooted at x_e. If $y = x_j$ with $j \leqslant i$, then a broadcast initiated at x costs $d^* + B_{Min}(T_{R^*})$. The same is true when $y = x_e$. Finally, y can be positioned so that there exists a path from x_e to y consisting of reduced edges. Again, a broadcast initiated at x costs $d^* + B^*$.

We have thus shown that $d^* + B^*$ is a lower bound for $B_{Arb}(T_{R^*})$. Clearly, it also is an upper bound and thus $B_{Arb}(T_{R^*}) = d^* + B^*$. □

4.2. *Description of algorithm Reduce_Br_Arb*

In the previous section we showed that the cost of an optimal reduction is the sum of a longest path distance and the cost for a *Br_Min* instance. Assume again that Algorithm *CF* has been applied to tree T and that T has been rooted at vertex c. Let $h(v)$ be the height of the subtree of T rooted at vertex v. Algorithm *Reduce_Br_Arb* determines an optimal solution by considering all possible longest path values. Clearly, $h(c)$ is an upper bound. If a tree T_R were to contain a path of cost $h(c)$, a broadcast initiated at vertex c would cost at least $h(c)$. Since the sum of the longest path and the cost of a broadcast initiated at c cannot exceed B, $d_{\max} = \min\{h(c), B/2\}$ is an upper bound.

For each distance d, $1 \leqslant d \leqslant d_{\max}$, let T_d be the tree containing the minimum number of reductions so the length of the longest path from any vertex to vertex c is at most d and $B_{Min}(T_d) = B - d$. Algorithm *Reduce_Br_Arb* generates $T_{d_{\max}}, \ldots, T_1$. During the generation of the trees, an edge (i) may not be reduced, (ii) may have a distance reduction (i.e., it is reduced to decrease the length of a longest path), or (iii) may have a broadcast reduction (i.e., it is reduced to decrease the broadcast time from the vertices in the center set). For each distance d, iteration d determines the best reduction having

(i) every edge $(u, p(u))$ with $h(u) \geqslant d$ has a distance reduction and all other reductions are broadcast reductions,

(ii) an extended center set, a longest path length of exactly $d + 1$ from a vertex to c, and distance at most d from any vertex to a vertex in the center set.

Observe that the distinction between the "distance to vertex c" and the "distance to a vertex in the center set" is crucial in the discussion of the algorithm.

Whether a broadcast reduction done for achieving cost $B-d$ needs to be made for broadcast cost $B-d+1$ depends on b-values and whether newly made distance reductions influence broadcast costs. To make this decision quickly, we use broadcast entries on edges, the *be*-entries. Recall how choice reductions on the edges into a vertex u were made in Algorithm *Reduce_Br_Min*: if, say, β choice reductions are to be made, *Reduce_Br_Min* makes them of the β leftmost unreduced edges. Making choice reductions this way in the *Br_Min* instance underlying each iteration of problem *Br_Arb* makes it difficult to determine the impact of distance reductions on existing broadcast reductions. We now make the choice reductions so that preference is given to vertices of large height. Let u be a vertex with children v_1 and v_2. When giving preference to subtrees of large height, then, if either edge (v_1,u) or edge (v_2,u) can receive a choice reduction, we reduce (v_1,u) iff $h(v_1)\geqslant h(v_2)$. We point out that it is possible solve *Br_Min* in $O(n)$ time when giving preference to vertices of large height in choice reductions.

Algorithm *Reduce_Br_Arb* does not explicitly solve *Br_Min* instances. Instead, the *be*-values determined in a preprocessing step contain the necessary information about reductions in *Br_Min* instances. Every edge (u,v) receives a broadcast edge value $be(u,v)$. If $be(u,v)=B'$, edge (u,v) has a broadcast reduction for every target broadcasting value $\leqslant B'$. Edge (u,v) is not reduced for a target value $\geqslant B'+1$. Hence, B' is the largest broadcast value for which edge (u,v) receives a broadcast reduction. The *be*-entries are determined by giving preference to vertices with a larger height when making choice reductions. In Section 4.3 we describe how to compute the *be*-entries an $O(n\log n)$ time.

As an example, consider vertex u in Fig. 9(a). Its eight children have the b-values $12,10,10,10,8,8,8,6$ and the h-values $4,5,3,6,6,6,6,6$, respectively. Vertex v_7 induces $b(u)=8+7=15$. One edge (v_i,u) with $1\leqslant i\leqslant 7$ receives a *be*-value of 14. Vertex v_4 is one of the vertices of the largest height and we set $be(v_4,u)=14$. The *be*-values for the eight edges are $12,11,10,14,13,9,8,6$, as shown on the edges.

Assume that distance $d+1$ has been considered and that tree T_{d+1} has been generated. At this point, an edge $(u,p(u))$ is in one of 3 states:

- $(u,p(u))$ has a distance reduction; implies $h(u)\geqslant d+1$. Distance reductions made remain as d decreases.
- $(u,p(u))$ is not reduced; implies $h(u)<d+1$ and $be((u,p(u))<B-d-1$.
- $(u,p(u))$ has a broadcast reduction; implies $h(u)<d+1$ and $be((u,p(u))\geqslant B-d-1$. Broadcast reductions can be removed or change into distance reductions as d decreases.

Observe that the longest path from a vertex in T_{d+1} to center vertex c can be smaller than $d+1$ (since broadcast reductions can decrease the longest path length). Compared to T_{d+1}, tree T_d will have more distance reductions and fewer broadcast reductions.

Fig. 7 gives an outline of Algorithm *Reduce_Br_Arb*. The first task for a particular distance d is the generation of set $\mathscr{E}$ which contains the vertices to be considered as extended center vertices for distance d. When u is an extended center vertex, a broadcast initiated at u costs $B-d$ and the broadcast cost experienced by every other vertex is $\leqslant B-d-1$. When the extended center set contains vertex u and a child v of u having height d, no distance reduction needs to be made on edge (u,v). Hence, extended centers may not require all distance reductions to be made.

Algorithm Reduce_Br_Arb

```
/* Generate initial reduction */
$d_{\max} = \min\{h(c), B/2\}$;
for each child $v$ of $u$ do
    if $h(v) \geq d_{\max} + 1$ then edge $(u,v)$ gets a distance reduction
    else if $be(u,v) \geq B - d_{\max} - 1$ then edge $(u,v)$ gets a broadcast reduction
endfor
$tot\_red$ = number of edges having received a reduction;
$current\_best = total\_red$;
for $d = d_{\max}$ down to 1 do
  (1) /* Generate and process set $\mathcal{E}$ */
  $\mathcal{E} = \{\ \}$;
  for each vertex $v$ with $h(v) = d$ do
     if $be(v, p(v)) \geq B - d - 1$ and $b(v) \leq B - d - 1$ then $\mathcal{E} = \mathcal{E} \cup \{p(v)\}$
  endfor
  $new\_d\_red$ = number of unreduced edges receiving a distance reduction
    in $T_d$
  Process_Extended_Set($\mathcal{E}$)

  (2) /* Make new distance reductions */
  for each edge $(v, p(v))$ with $h(v) = d$ do
     $(v, p(v))$ gets a distance reduction
  endfor

  (3) /* Increase broadcast cost to $B - d$ */
  for each edge $e$ with $be(e) = B - d - 1$ do
     if $e$ has no distance reduction then remove the reduction on $e$
  endfor

  (4) /* compute cost of reduction for $T_d$ */
  $rem\_b\_red$ = number of removed broadcast reductions;
  $tot\_red = tot\_red + new\_d\_red - rem\_b\_red$;
  if $tot\_red < current\_best$ then $current\_best = tot\_red$
endfor
```

Fig. 7. Outline of algorithm *Reduce_Br_Arb*.

Set $\mathcal{E}$ contains the vertices to be considered as extended center vertices in iteration d. A vertex cannot be considered in each iteration. Doing so would result in $O(n^2)$ time. When u has no child of height d, making u an extended center vertex does not save on distance reductions. Hence, only vertices u having a child v with $h(v) = d$ are of interest. Let u be a vertex with children $v_1, \ldots, v_q$. Tree T_{d+1} has a broadcast cost of $B-d-1$. Vertex u is considered as an extended center vertex and put into set $\mathcal{E}$ if at

Algorithm Process_Extended_Set($\mathscr{E}$)

```
for each vertex $u$ in $\mathscr{E}$ do
    (1) among all children $v_1, v_2, \dots$ of $u$ let $i$ be the largest index
    such that edge $(v_i, u)$ has a broadcast reduction (implies that $h(v_i)$
    is a minimum)

    (2) remove the broadcast reduction on edge $(v_i, u)$

    (3) determine the smallest-indexed vertex $v_s$, $s \geqslant i$, such that
    $v_s$ induces a broadcast cost of $B-d$ for vertex $u$

    (4) determine the cost of the reduction having $u$ as the
    extended center vertex:
        $in\_cen$ = number of vertices $v_j$ with $h(v_j) = d$, $j < s$, and
                $(v_j, u)$ has no broadcast reduction
        $cost\_ext = tot\_red + new\_d\_red - in\_cen - 1$
        if $cost\_ext < current\_best$ then $current\_best = cost\_ext$

    (5) restore the broadcast reduction on edge $(v_i, u)$
endfor
```

Fig. 8. Outline of algorithm *Process_Extended_Set*.

least one child has height d and there exists a child v_i such that (i) edge (v_i, u) has a broadcast reduction (i.e., $be(u, v_i) \geqslant B - d - 1$) and (ii) the subtree rooted at v_i does not contain any broadcast reductions (i.e., $b(v_i) \leqslant B - d - 1$).

Fig. 8 describes Algorithm *Process_Extended_Set* which determines for each vertex u in $\mathscr{E}$ the cost of the reduction when u is an extended center vertex. Step (1) determines the child v_i of u such that (u, v_i) has a broadcast reduction, i is a maximum, and $h(v_i)$ is a minimum over all such edges. Choosing the vertex with the minimum h-value results in a center set of maximum size. This is crucial for the correctness argument. We then remove the broadcast reduction on edge (u, v_i). Observe that u is indeed an extended center vertex: when applying algorithm *CF* to the corresponding tree: vertex u is the last vertex processed and its broadcast cost is $B-d$. Further, reducing one edge, namely (u, v_i) decreases the broadcast cost for u by 1. Step (3) determines v_s, the vertex inducing a broadcasting cost of $B-d$ at vertex u (ties are broken in favour of the smallest index). The center set contains all vertices incident to reduced edges and vertices $\{u, v_1, \dots, v_s\}$. Edges (u, v_j) with $j \leqslant s$, $h(v_j) = d$ and $be(u, v_j) \leqslant B-d-1$ do not require a distance or broadcast reduction. All other edges $(x, p(x))$ with $h(x) = d$ receive a distance reduction. Step (4) determines the cost of the resulting reduction.

After the vertices in set $\mathscr{E}$ have been handled, Step (2) of Algorithm *Reduce_Br_Arb* determines the new distance reductions for T_d and Step (3) removes the broadcast reductions no longer necessary for a broadcast cost of $B - d$. Step (4) determines the cost of the reduction which makes all distance reductions for distance d and has a broadcast cost of $B - d$.

Fig. 9 illustrates Algorithm *Reduce_Br_Arb* for a vertex u and its 8 children with a target cost $B = 17$. The *be*-values correspond to the integers on the edges. The "current broadcast time" entry within a vertex corresponds to the cost of a broadcast initiated to the vertices in the subtree. Fig. 9(a) shows vertex u and its children with associated *b*-, *be*-, and *h*-entries in the original tree T. Current broadcast values are thus identical to initial broadcast time values. Fig. 9(b) shows vertex u and its children in T_7. In T_7, a broadcast initiated at u costs 10. Edge $(u, p(u))$ has a distance reduction (since $h(u) = 7$). Five of u's incoming edges have a broadcast reduction. Observe that the subtree rooted at v_1 contains further broadcast reductions (since $b(v_1) = 12$). The subtrees incident to v_2, v_3, v_4, and v_5 contain no broadcast reductions. Consider now the iteration with $d = 6$. Vertices v_4 and v_5 satisfy the condition that places u into $\mathscr{E}$. When vertex u is processed in *Process_Extended_Set*, the broadcast reduction on edge (v_5, u) is removed (since v_5 has the smallest *b*-values). This causes the broadcast cost at u to increase to 11 and vertex v_7 is the vertex inducing cost 11. The extended center set contains $v_1, \ldots, v_6, v_7$. The solution for $d = 6$ with u as the extended center is shown in Fig. 9(c). It contains 4 broadcast reductions and one distance reduction on incoming edges for u. Observe that edges (v_5, u), (v_6, u), and (v_7, u) do not require a distance reduction. Fig. 9(d) shows the status of the edges in T_6. A broadcast initiated at u (or any other vertex) now costs at most 11. In T_6, five of u's incoming edges have a new distance reduction. The broadcast reduction on (u, v_3) is removed, while the broadcast reductions on (u, v_1) and (u, v_2) remain.

The next theorem shows the correctness of Algorithm *Reduce_Br_Arb*. Its $O(n \log n)$ running time is discussed in the next section.

Theorem 8. *Algorithm Reduce_Br_Arb solves problem Br_Arb under edge reductions.*

Proof. Let R^* be an optimal solution for *Br_Arb*. Among all optimal reductions we choose one whose reduced edges form a tree containing vertex c and which gives preference to vertices of large height (the height is defined with respect to the tree rooted at c). Let c^* be the last vertex processed when Algorithm *CF* is applied to tree T_{R^*}. Let $h^*(u)$ be the height of vertex u in T_{R^*} (edges in T_{R^*} are directed towards c^* and reduced edges do not contribute to the height).

Making use of Lemma 7, the cost of R^* can be expressed as $B_{Arb}(T_{R^*}) = d^* + B^* = B$. Clearly, $1 \leqslant d^* \leqslant \min\{h(c), B/2\}$. To show that Algorithm *Reduce_Br_Arb* generates a solution of cost $B_{Arb}(T_{R^*})$, consider iteration d^*. Tree T_{d^*+1} contains the reductions at the beginning and T_{d^*} the reductions at the end of iteration d^*, respectively. The height of vertex c^* in T_{R^*} is either d^* or $d^* + 1$. We considers these two cases separately.

Case 1: $h^*(c^*) = d^*$. We show that T_{d^*} is an optimal reduction. Assume R^* is chosen so that it agrees with T_{d^*} on as many reduced edges as possible. When R^* corresponds to a simple center set, the longest path from c^* (or from any vertex

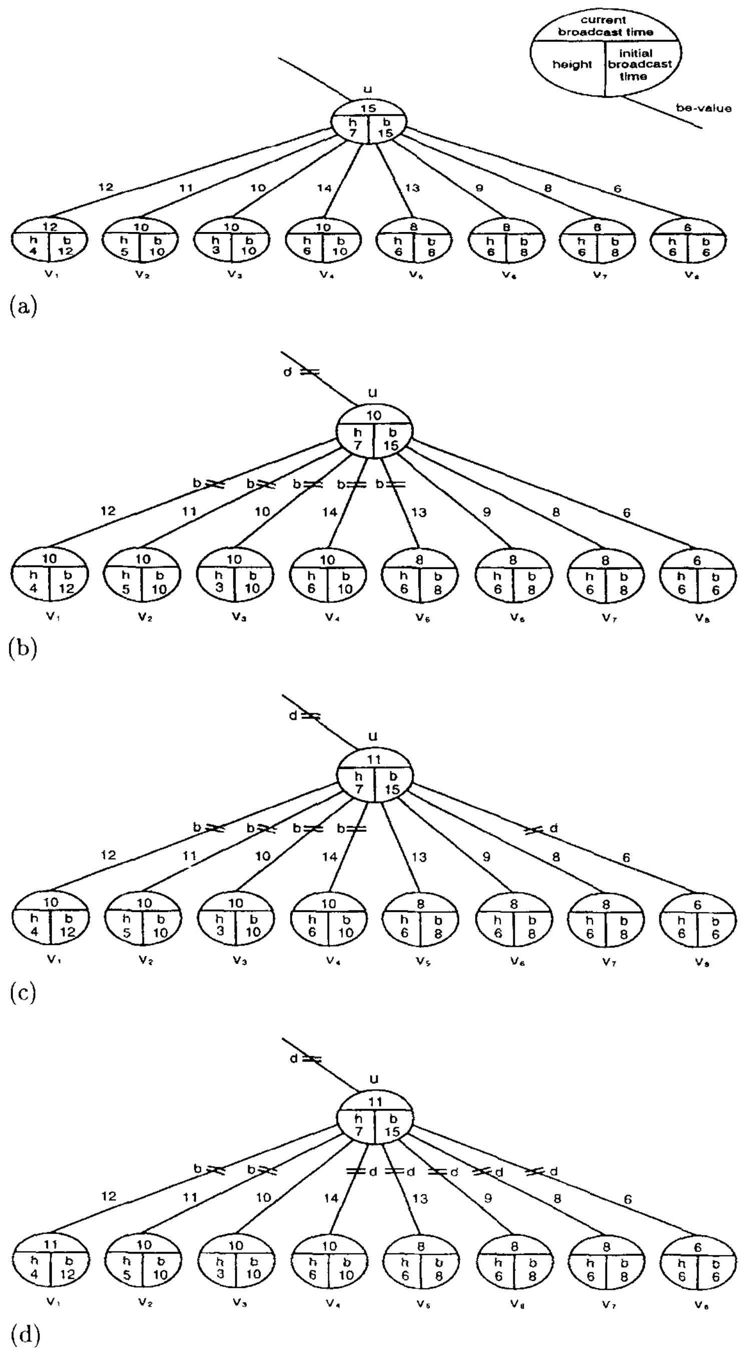

Fig. 9. Illustration of Algorithm *Reduce_Br_Arb*. (a) Vertex u with children $v_1, v_2, \ldots, v_8$ in T; $B = 17$. (b) Vertex u and its children in T_7 with $B - 7 = 10$. (c) Vertex u as the extended center vertex in iteration $d = 6$. (d) Vertex u in T_6 with $B - 6 = 11$.

incident to a reduced edge) to a vertex in T_{R^*} is d^*. When R^* corresponds to an extended center set, every vertex u in the center set with $(u, p(u))$ not in R^* has $h^*(u) < d^*$. Whether a simple or extended center set, $h^*(c^*) = d^*$ implies that every distance reduction made in T_{d^*} is in R^*. To show that the broadcast reductions of T_{d^*} are also in R^* we only need to consider edges $(u, p(u))$ with $h(u) < d^*$.

Observe that the directions on the edges in T_{d^*} and T_{R^*} differ only for the edges on the path from c to c^*. Edges on this path are reduced in T_{d^*} (because they have a distance reduction) as well as T_{R^*} (because the tree formed by the reduced edges contains c). Let $(u, p(u))$ be an edge with $h(u) < d^*$. Assume $(u, p(u))$ is reduced in T_{d^*}. We consider two cases, depending on why edge $(u, p(u))$ received the broadcast reduction in T_{d^*}.

- $b(u) \geqslant B - d^*$: $(u, p(u))$ has a forced broadcast reduction in T_{d^*}. Since a broadcast initiated at $p(u)$ costs at most $B - d^*$ in T_{R^*} and the reduced edges form a tree, edge $(u, p(u))$ is also in R^*.
- $b(u) < B - d^*$ and $b(p(u)) \geqslant B - d^*$: Assume $(u, p(u))$ has a broadcast reduction in T_{d^*} and $(u, p(u))$ is not in R^*. Were $(u, p(u))$ not reduced in T_{d^*}, the broadcast cost for vertex $p(u)$ would increase to $B - d^* + 1$. Let u' be the leftmost child of $p(u)$ inducing $B - d^* + 1$ when edge $(u, p(u))$ is not reduced in T_{d^*}. Then, there exists a child u'' to the left of u' (including u') such that edge $(u'', p(u))$ is in R^*, but $(u'', p(u))$ is not reduced in T_{d^*}. This implies $b(u'') < B - d^*$ and $h(u'') < d^*$. Generate reduction R' such that $R' = R^* - \{(u'', p(u))\} \cup \{(u, p(u))\}$. Clearly, R' is another optimal solution and it agrees with T_{d^*} on more edges. This contradicts our assumption that R^* and R agree on as many edges as possible. Hence, $(u, p(u))$ is in R^*.

It follows that every edge having a broadcast reduction in T_{d^*} is in R^*. Since R^* is an optimal reduction, there exists no edge which is reduced in R^*, but not in T_{d^*}. Hence, $T_{d^*} = T_{R^*}$.

Case 2: $h^*(c^*) = d^* + 1$. Reduction R^* has now an extended center set with c^* as the extended center vertex. Further, c^* has a child v belonging to the center set with $h^*(v) = h(v) = d^*$ and edge (v, c^*) not in R^*.

In iteration d^*, Algorithm *Reduce_Br_Arb* considers only selected vertices as extended center vertices. More specifically, a vertex x_e is put into set $\mathscr{E}$ when x_e has a child v_i such that $h(v_i) = d^*$, no edge in the subtree rooted at v_i has a broadcast reduction, and (v_i, x_e) has a broadcast reduction in T_{d^*+1}. The first two conditions are obviously satisfied for vertices c^* and v. We next show that vertex c^* has a child v_i for which all three condition are satisfied.

A broadcast initiated at c^* costs $B - d^*$ in T_{R^*}, while a broadcast initiated at a child of c^* costs at most $B - d^* - 1$ in T_{R^*}. Let u' be the leftmost child of c^* inducing cost $B - d^*$ for c^* in T_{R^*}. A broadcast initiated at c^* costs at most $B - d^* - 1$ in T_{d^*+1}. Hence, there exists an edge (u, c^*) such that u is to the left of u', (u, c^*) is reduced in T_{d^*+1}, but (u, c^*) is not in R^*. It follows that no edge in the subtree rooted at u is reduced in T_{d^*+1} (otherwise (u, c^*) would be in R^*). Since the algorithm gives preference to edges of large height, we have $h(u) = d^*$. (If the height of u were less, the existence of edge (v, c^*) would contradict Algorithm *Reduce_Br_Arb.*) Vertex u is thus a child of c^* which satisfies the conditions for placing c^* into set $\mathscr{E}$.

Let R' be the reduction whose cost is computed when c^* is handled in *Process_Extended_Set*. We next show that the number of edges reduced in R' equals that of the reduced edges in R^*. Analog to Case 1, the edges on the path from c to c^* are reduced in both reductions. Hence, when referring to an edge $(u, p(u))$ below, the parent is the same vertex in both trees. Consider first an edge $(u, p(u))$ with $p(u) \neq c^*$ (and u not on the path from c to c^*). When $h(u) \geqslant d^*$, the edge is reduced in both R' and R^*. For an edge with $h(u) < d^*$, we argue as done in Case 1. It then follows that R' and R^* agree on the action for the edges with $p(u) \neq c^*$.

Consider now the children of vertex $p(u) = c^*$. Clearly, an edge $(u, p(u))$ with $h(u) \geqslant d^* + 1$ is in both R' and R^*. If $b(u) \geqslant B - d^*$, edge $(u, p(u))$ is reduced in both R' and R^*. The interesting edges are $(u, p(u))$ with $h(u) \leqslant d^*$ and $b(u) < B - d^*$ when $b(p(u)) \geqslant B - d^*$. Consider how reduction R' is determined in *Process_Extended_Set*. Let v_s be the vertex determined in Step (3); i.e., v_s is the vertex inducing broadcast cost $B - d^*$ after edge (v_i, u) looses its broadcast reduction. Let R'' be the reduction represented by tree T_{d^*+1} without the broadcast reduction on (v_i, u). Observe that in R'' the reductions for distance d^* have not yet been done (i.e., edges (v_k, c^*) with $h(v_k) = d^*$ do not yet have a distance reduction).

Let v_t be the leftmost vertex inducing broadcast cost $B - d^*$ in R^*. We next show that $s \geqslant t$. Assume to the contrary that $s < t$. Let $l_{a,b}$ (resp. $m_{a,b}$) be the number of edges **not** reduced among edges (v_j, c^*), $a \leqslant j \leqslant b$, in R'' (resp. R^*). In R'', vertex v_s is the leftmost vertex inducing $B - d^*$ and thus

$$b(v_s) + l_{1,s} = B - d^* \quad \text{and} \quad b(v_t) + l_{1,s} + l_{s-1,t} \leqslant B - d^*.$$

In R^*, v_t is the leftmost vertex inducing cost $B - d^*$ and thus

$$b(v_s) + m_{1,s} < B - d^* \quad \text{and} \quad b(v_t) + m_{1,s} + m_{s+1,t} = B - d^*.$$

It follows that $l_{1,s} > m_{1,s}$. From the way vertex v_i was chosen in Step (1) of *Process_Extended_Set* it follows that no edge (v_k, c^*) with $s < k \leqslant t$ has broadcast reduction and, if it has a distance reduction, it has $h(v_k) > d^*$. This implies $l_{s+1,t} \geqslant m_{s+1,t}$. Using these two inequalities, we get

$$b(v_t) + m_{1,s} + m_{s+1,t} = B - d^* < b(v_t) + l_{1,s} + l_{s+1,t} \leqslant B - d^*,$$

which is a contradiction. It thus follows that $s \geqslant t$. When $s \geqslant t$, every distance reduction made in R' is also in R^*. Similar to the argument given for Case 1, we can show that R^* and R' agree on the broadcast reductions done. Hence, R' is an optimal solution and the correctness of Algorithm *Reduce_Br_Arb* follows. □

4.3. Data structures for achieving O($n \log n$) time

In this section we describe the data structures giving $O(n \log n)$ time. Algorithm *Reduce_Br_Arb* assumes that the height and b-value of every vertex and the be-value of every edge have already been determined. Height and b-values can easily be computed in $O(n)$ time. The computation of the be-values when giving preference to vertices of large height in choice reductions is described in Section 4.3.1. The optimum reduction is determined by considering decreasing distance values and generating trees $T_{d_{\max}}, \dots, T_1$. For each distance value d, we invoke Algorithm *Process_Extended_Set* which handles the potential extended set vertices for distance d. Since a vertex u can be considered as an extended set vertex a number of times (each time in a different iteration), *Process_Extended_Set* uses additional data structures in order to achieve the $O(n \log n)$ overall time. In Section 4.3.2 we describe these data structures. The remaining implementation issues of *Reduce_Br_Arb* are straightforward and are sketched in Section 4.3.2.

4.3.1. Computing the be-entries

Recall that if $be(u,v) = B'$, edge (u,v) is reduced in an *Br_Min*-problem with target broadcast value $\leqslant B'$. Assume u is a vertex with children $v_1, v_2, \dots, v_q$ and $b(v_1) \geqslant \cdots \geqslant b(v_q)$. For a target broadcast cost B' with $B' \geqslant b(u)$, no edge in the subtree rooted at u needs to be reduced. For a target broadcast cost B' with $B' \leqslant b(v_i)$, edge (v_i, u) needs to be reduced. We thus have $b(v_i) \leqslant be(v_i,u) \leqslant b(u) - 1$.

When every v_i is a leaf, we have $b(v_i) = 0$, $1 \leqslant i \leqslant q$, and

$$be(v_1,u) = q - 1, be(v_2,u) = q - 2, \dots, be(v_q,u) = 0.$$

When determining the be-values for arbitrary vertices, our solution asks queries about the height and about quantities initially of the form $b(v_i) + i$. Let $\mathscr{H}_u$ be a data structure for vertex u containing vertices $v_1, v_2, \dots, v_q$ and their heights and supporting the following queries:

- *Delete*($\mathscr{H}_u, v_i$): delete vertex v_i from $\mathscr{H}_u$,
- *Max-Range*($\mathscr{H}_u, i$): determine the index of the vertex having maximum height among $v_1, \dots, v_i$.

By imposing a binary tree $\mathscr{H}_u$ upon vertices $v_1, v_2, \dots, v_q$ and maintaining appropriate entries, each query takes $O(\log q)$ time. The entries in $\mathscr{H}_u$ represent for each node a of the tree the quantity $max\_h(a)$ which contains the maximum height (and the vertex inducing this height) among the leaves in the subtree rooted at a.

A second data structure, $\mathscr{B}_u$, organizes, for each vertex u, vertices $v_1, v_2, \dots, v_q$ by their $b'(v_i) = b(v_i) + i$ value. Data structure $\mathscr{B}_u$ supports the following operations in $O(\log q)$ time:

- *Delete*($\mathscr{B}_u, v_i$): delete vertex v_i from $\mathscr{B}_u$,
- *Max*($\mathscr{B}_u$): determine the maximum entry b'-entry in $\mathscr{B}_u$ (ties are broken in favor of vertices with small indices),
- *Decrease*($\mathscr{B}_u, h$): decrease the value of every entry $b'(v_i)$ with $i > h$ by 1,
- *Value*($\mathscr{B}_u, i$): compute the current $b'(v_i)$ value of vertex v_i.

$\mathscr{B}_u$ is implemented by imposing as a binary tree on $v_1, v_2, \dots, v_q$. In order achieve $O(\log q)$ time for each operation, we maintain decrease- and max-values for each node in $\mathscr{B}_u$. Let $d(a)$ and $max\_b'(a)$ be the decrease- and max-value for node a,

```
Algorithm Comp_Be(u)
while $\mathscr{H}_u$ and $\mathscr{B}_u$ are not empty do
    (1) $r = \text{Max}(\mathscr{B}_u)$ and $h = Max - Range(\mathscr{H}_u, r)$
    (2) $be(v_h, u) = \text{Value}(\mathscr{B}_u, r) - 1$
    (3) $Decrease(\mathscr{B}_u, h)$
    (4) $Delete(\mathscr{B}_u, v_h)$ and $Delete(\mathscr{H}_u, v_h)$
endwhile
```

Fig. 10. Determining the *be*-values for the children of u.

respectively. The current value of $b'(v_i)$ is determined by subtracting the $d(\cdot)$-values of the nodes on the path from the root to leaf v_i from the original value of $b'(v_i)$ (which remains stored at leaf v_i). Entry $max\_b'(a)$ corresponds to the current maximum b'-value associated with vertices in the subtree rooted at a (this includes the d-values for nodes in the subtree at a, but not the ones outside the subtree). Operation $Max(\mathscr{B}_u)$ can be implemented in O(1) time by using the max_b'-value associated with the root of $\mathscr{B}_u$. Operation $Decrease(\mathscr{B}_u, h)$ first locates leaf v_h and then follows the path from this leaf to the root. Assume we are at node a and reached node a from child b. Node b could have a new, smaller $max\_b'(b)$ value. If b is the right child of a, we check whether $max\_b'(a)$ needs to be updated ($max\_b'(a)$ decreases if node b contributed its value and $max\_b'(b)$ was decreased). If b is the left child of a, let c be the right child. We then set $d(c) = d(c) - 1$ and check whether $max\_b'(a)$ needs to be updated. The remaining details are straightforward.

Setting up data structures $\mathscr{H}_u$ and $\mathscr{B}_u$ for vertex u costs $O(q)$ time. Once $\mathscr{H}_u$ and $\mathscr{B}_u$ are available, $be(v_1, u), be(v_2, u), \ldots, be(v_q, u)$ are computed by invoking Algorithm *Comp_Be*(u) outlined in Fig. 10.

At any iteration of *Comp_Be*(u), r is the index of the leftmost child of u currently inducing the maximum broadcast value at u (edges having already received a *be*-value make no contribution). Further, h is the index of a vertex to the left of r (including r) such that (v_h, u) has no *be*-value yet and v_h has maximum height. Edge (v_h, u) is the next edge to receive its *be*-value. Then, the b'-values for the vertices to the right of v_h are decreased and v_h is deleted from both data structures. For the example given in Fig. 9(a), the first iteration generates $r = 7$ since v_7 has maximum b'-value (i.e., $b'(v_7) = 8 + 7 = 15$). There exist four vertices of maximum height 6 and the algorithm chooses $h = 4$. It then sets $be(v_4, u) = 14$. The correctness of Algorithm *Comp_Be*(u) is straightforward and we omit further details. The $O(n \log n)$ running time for computing the *be*-values follows from the described implementations of the data structures.

4.3.2. *Algorithm Process_Extended_Set*

In this section we describe the data structures used by Algorithm *Process_Extended_Set* so that the total work done in *Process_Extended_Set* is bounded by $O(n \log n)$. We assume that the h-, b-, and *be*-entries have been already been determined. Recall that these entries are determined with respect to the tree rooted at vertex c. Algorithm

Reduce_Br_Arb accesses vertices sorted by h- an *be*-entries and such lists can be set up easily. In order to achieve $O(n \log n)$ time for the processing of all extended center vertices, we set up the following two data structures. For every vertex u which has at least one child of height d, list $H_{u,d}$ contains the children of u having height d. The children do not need to be arranged in a particular order. Clearly, these lists can be set up so that for a given vertex u and degree d, a pointer to list $H_{u,d}$ is generated in $O(\log n)$ time.

The second data structure is similar to the $\mathscr{B}_u$ trees used for computing the *be*-entries: for every vertex u with children $v_1, \dots v_q$, $b(v_1) \geqslant b(v_2) \geqslant \cdots \geqslant b(v_q)$, we create a binary tree structure $\mathscr{B}'_u$. Vertex v_i has initially the entry $b'(v_i) = b(v_i) + i$ associated with it. When reductions are made, these entries are updated to reflect the current broadcast time. The data structure supports the following operations:

- *Rightmost*($\mathscr{B}'_u$): return the rightmost vertex v_i such that edge (v_i, u) has a broadcast reduction,
- *Decrease*($\mathscr{B}'_u, h$): decrease the value of every entry $b'(v_i)$ with $i > h$ by 1,
- *Max*($\mathscr{B}'_u$): return the maximum b'-value (ties are broken in favor of smaller indices).

Step (1) of Algorithm *Process_Extended_Set* corresponds to $v_i =$ *Rightmost*($\mathscr{B}'_u$). Next, the necessary updates for the temporary removal of the broadcast reduction on edge (v_i, u) are performed. Then, $v_s =$ *Max*($\mathscr{B}'_u$). Step (4) determines the cost of the new reduction. We use list $H_{u,d}$ to determine the number of edges contributing to *in_cen*. This can be done by simply scanning the entries in list $H_{u,d}$. Overall, vertex u is handled in $O(\log q + |H_{u,d}|)$ time. For each distance d, vertex u is handled at most once and an edge (v_i, u) can cause u to be considered at most once. Hence, the total cost for vertex u during *Reduce Br_Arb* is $O(q \log q)$. The time spent in *Process_Extended_Set* for all vertices is thus bounded by $O(n \log n)$.

We conclude this section with a brief discussion of the remaining steps of Algorithm *Reduce_Br_Arb*. As already stated, *Reduce_Br_Arb* has available the vertices sorted by height and b-values as well as the edges sorted by *be*-values. For every edge, we record whether the edge is reduced and the type of reduction. The vertices of height d are used to determine the new distance reductions in T_d. The edges with *be*-value $B - d - 1$ are used to determine the removal of broadcast reductions. Ignoring the preprocessing steps and the work done in *Process_Extended_Set*, the work done in *Reduce_Br_Arb* is $O(n)$. Hence, Algorithm *Reduce_Br_Arb* solves *Br_Arb* under edge reductions in $O(n \log n)$ time.

5. Conclusions

We presented an $O(n)$ time algorithm for problem *Br_Min* and an $O(n \log n)$-time algorithm for *Br_Arb* under edge reductions in trees with unary weights. For both problems we showed that there exists an optimal reduction in which the reduced edges form a tree containing the root of the broadcast tree without edge reductions. A natural generalization is to consider trees with arbitrary, positive weights. Our algorithm for

Br_Min under edge reductions can be generalized to handle arbitrary weights. The running time increases to $O(n \log n)$. However, our approach for *Br_Arb* under edge reductions fails for trees with arbitrary weights. The main reason is that for *Br_Arb* with arbitrary weights it is no longer true that the cost of a broadcasting from an arbitrary vertex is the sum a longest path distance plus the cost of a *Br_Min* instance. The actual cost can be lower. While this cost can be characterized, the characterization does not seem to lead to an efficient algorithm.

References

[1] A. Bar-Noy, S. Kipnis, Designing broadcasting algorithms in the postal model for message-passing systems, Math. System Theory 27 (5) (1994) 431–452.

[2] A. Bar-Noy, S. Kipnis, B. Schieber, Optimal broadcasting in telephone communication networks, Proc. 6th IEEE Symp. on Parallel and Distributed Processing, 1994, pp. 216–223.

[3] H.-C. Chen, D.H.-C. Du, L.-R. Liu, Critical path selection for performance optimization, IEEE Trans. on Comput.-Aided Des. Integrated Circuits Systems 12 (1993) 185–195.

[4] T.H. Cormen, C.E. Leiserson, R.L. Rivest, Introduction to Algorithms, MIT Press, Cambridge, MA, 1990.

[5] G.D. Eppen, F.J. Gould, C.P. Schmidt, Introductory Management Science, Prentice-Hall, Englewood Cliffs, NA, 1992.

[6] A.M. Farley, Broadcast time in communication networks, SIAM J. Appl. Math. 39(2) (1980) 385–390.

[7] P. Fraigniaud, E. Lazard, Methods and problems of communication in usual networks, Discrete Appl. Math. 53 (1994) 79–133.

[8] S.E. Hambrusch, H.-Y. Tu, Edge weight reduction problems in directed, acyclic graphs, J. Algorithms, 24 (1997) 66–93.

[9] S.M. Hedetniemi, S.T. Hedetniemi, A.L. Liestman, Survey of gossiping and communication networks, Networks 18(4) (1988) 319–349.

[10] C.T. Ho, Optimal broadcasting on SIMD hypercubes without indirect addressing capability, J. Parallel Distributed Comput. 13 (1991) 246–255.

[11] S.L. Johnsson, C.-T. Ho, Optimum broadcasting and personalized communication in hypercubes, IEEE Trans. Comput. 38 (1989) 1249–1268.

[12] R.M. Karp, A. Sahay, E. Santos, K.E. Schauser, Optimal broadcast and summation in the logP model, Proc. 5th ACM Symp. on Parallel Algorithms and Architectures, 1993, pp. 142–153.

[13] J.-M. Koh, D.-W. Tcha, Information dissemination in trees with nonuniform edge transmission times, IEEE Trans. Comput. 40 (1991) 1174–1177.

[14] S.O. Krumke, M.V. Marathe, H. Noltemeier, R. Ravi, S.S. Ravi, R. Sundaram, H.C. Wirth, Improving spanning trees by upgrading nodes, Proc. 24th ICALP, Lecture Notes in Computer Sciences, vol. 1256, 1997, 281–291.

[15] D. Paik, S. Sahni, Network upgrading problems, Networks 26 (1995) 45–58.

[16] P.J. Slater, E.J. Cockayne, S.T. Hedetniemi, Information dissemination in trees, SIAM J. Comput. 10 (1981) 692–701.