Minimizing the Diameter in Tree Networks under Edge Reductions

Susanne E. Hambrusch*  
Department of Computer Sciences  
Purdue University  
West Lafayette, IN 47907, USA  
seh@cs.purdue.edu

Hyeong-Seok Lim†  
Department of Computer Science  
Chonnam National University  
Kwangju, 500-757, Korea  
hslim@chonnam.chonnam.ac.kr

July 14, 1998

Abstract

Edge reductions model the ability of speeding up communication between processors in a network. In this paper we consider performing edge reductions in a tree network so that the diameter of the tree is minimized. The diameter is directly related to the time needed to broadcast a message. In particular, for non-blocking message communication ignoring start-up costs, the diameter corresponds to the broadcasting time. For an \( n \)-vertex tree \( T \) and a quantity \( B \), we present an \( O(n) \) time algorithm to determine edge reductions such that the resulting tree has diameter at most \( B \) and the cost of the reduction is a minimum (over all reductions resulting in a diameter of at most \( B \)).

Keywords: Broadcasting, tree networks, diameter, analysis of algorithm.

---

*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 U.S. government.

†Research supported in part by Korea Research Foundation, 1996. Research was done while visiting Purdue University.
1 Introduction

Broadcasting a message in a network is a fundamental communication operation and broadcasting algorithms for different communication models have been developed [1, 2, 5]. The objective of a broadcasting algorithm is to minimize the time at which the message is received by the last processor. The diameter of the network is one of the parameters having crucial impact on the broadcasting time. For some communication models (e.g., non-blocking message broadcasting ignoring the set-up cost), the diameter accurately represents the broadcasting time. For others, the diameter is a lower bound which enters the actual broadcasting time along with other parameters.

In this paper we consider where to place edge reductions in a tree network so that the diameter of the resulting tree is a minimum. Edge reductions model the ability of speeding up communication between vertices. Our work thus determines where to locate the faster communication links in order to reduce the broadcasting time. Edge reductions in longest path computations with applications in circuit layout and project management have been studied in [4]. Edge reductions for blocking communication models are considered in [3].

Let \( T = (V, E) \) be an undirected, weighted, \( n \)-vertex tree representing a broadcasting network. The weight of edge \((i, j)\), \( w(i, j)\), represents the time needed for sending a message from vertex \( i \) to vertex \( j \). We assume \( w(i, j) \geq 0 \) and \( w(i, j) = w(j, i) \). Let \( d(i, j) \) be the length of the path from \( i \) to \( j \), where the length is the sum of weights of the edges. For vertex \( i \), we use \( B_i = \max_{1 \leq j \leq n} d(i, j) \) to denote the maximum time for completing a broadcast initiated at \( i \). The \textit{broadcasting cost} of \( T \), which is also the \textit{diameter} of \( T \), is \( B_T = \max_{1 \leq i \leq n} \{ B_i \} \).

An edge reduction \( R \) assigns to every edge \((i, j)\) a non-negative quantity \( r(i, j) \). The reduced weight \( w_r(i, j) \) of edge \((i, j)\) is a function of the edge's weight and its reduction. An edge reduction \( R \) is called a \textit{linear reduction} if for every edge \((i, j)\), \( r(i, j) \) is a non-negative real and

\[
  w_r(i, j) = w(i, j) - r(i, j).
\]

Given a reduction \( R \) for tree \( T \), the \textit{reduced tree} \( T_R \) is obtained from \( T \) by replacing each edge weight \( w(i, j) \) by its reduced weight \( w_r(i, j) \). Let \( M(T_R) \) denote the cost of reduction \( R \); i.e.,

\[
  M(T_R) = \sum_{(i,j) \in E} r(i,j).
\]

Given a tree \( T \) and a quantity \( B, B < B_T \), we describe an \( O(n) \) time algorithm to determine an edge reduction \( R \) so that \( T_R \) has broadcasting cost at most \( B \) and \( M(T_R) \) is a minimum.
(over all linear reductions resulting in a broadcasting cost of at most $B$). In Section 2 we give relevant characterizations of an optimal linear reduction. In Section 3 we describe the $O(n)$ time algorithm.

## 2 Characterizations of optimal reductions

In this section we give characterizations of optimal reductions which form the basis of the $O(n)$ time algorithm. As already stated, the diameter $B_T$ of tree $T$ corresponds to the cost of broadcasting a message in a non-blocking communication model. Consider all paths of length $B_T$ in $T$. Let $T_p$ be the subtree of $T$ containing the edges belonging to these longest paths. Let $v_1, \ldots, v_l$ be the leaves of $T_p$, $l \geq 2$. For the tree shown in Figure 1(a), $T_p$ corresponds to the bold edges in 1(b). Next, we define the absolute center in a tree.

![Tree Diagrams](image)

(a) tree $T$ with $B_T = 30$  
(b) trees $T$ and $T_p$ with absolute center $c$  
(c) reduced tree for $B=18$

Figure 1: Trees $T$ and $T_p$ with $B_T = 30$ and $B = 18$

The absolute center vertex $c$ corresponds either to a vertex of the tree or a virtual vertex on an edge. Center vertex $c$ satisfies $d(c, v_i) = B_T/2$ for all leaves $v_i$ in $T_p$, $1 \leq i \leq l$. When the absolute center corresponds to a position of an edge $(a, b)$, we introduce a new vertex $c$ in $T_p$ and $T$, remove edge $(a, b)$ and add edges $(a, c)$ and $(c, b)$. The weights on the new edges are set so that $c$ satisfies the absolute center conditions. The center vertex can be found as follows. Consider any longest path in tree $T$. Assume the endpoints of the path are $x$ and $y$. Then,
determine the vertex or create a virtual vertex $c$ on the path such that $d(c, x) = d(c, y) = B_T/2$. Vertex $c$ is unique for the chosen path. Furthermore, for any other longest path in $T$, vertex $c$ is also the center vertex. If this would not be the case, $T$ would contain a path having length greater than $B_T$. Clearly, for all leaves $u$ in $T$ which are not in $T_p$ we have $d(c, u) \leq B_T/2$.

Figure 1(b) shows tree $T$ with its absolute center vertex $c$ being a virtual vertex on edge $(u_1, u_2)$. Each one of the three leaf vertices in $T_p$ has distance 15 to vertex $c$.

Throughout, we use $P(x, y)$ to denote the set of edges on the path from $x$ to $y$. From the context it will be clear whether the path is in $T$ or $T_p$. Recall that given a target cost $B$, our algorithm determines an edge reduction $R$ of minimum cost so that the length of every path in $T_R$ is at most $B$. We next characterize the length of the paths $P(v_i, c)$ in an optimal reduction: first for the case when $v_i$ is a leaf of $T_p$ and then for arbitrary leaves.

**Lemma 1** There exists an optimal reduction $R$ with $d_R(c, v_j) = B/2$ for every leaf $v_j$ in $T_p$, $1 \leq j \leq l$.

**Proof:** Let $R^*$ be an optimal reduction for which the lemma does not hold. Consider first the case when there exists a leaf $v_j$ in $T_p$ such that the length of the path from $c$ to $v_j$ exceeds $B/2$; i.e., $d_{R^*}(c, v_j) = B/2 + \alpha$ for some $j$, $1 \leq j \leq l$, $\alpha > 0$. Let $v_i$ be any other leaf in $T_p$ such that the path from $v_i$ to $v_j$ contains vertex $c$ (at least one such vertex $v_i$ exists). We then have $d_{R^*}(c, v_i) \leq B/2 - \alpha$. Further, for any leaf $x$ of $T$ such that the path from $x$ to $v_j$ contains vertex $c$, we have $d_{R^*}(c, x) \leq B/2 - \alpha$. Let $e = (a, b)$ be the edge on path $P(c, v_i)$ closest to vertex $v_i$ which has a reduction in $R^*$. Choose $a$ to be the vertex closer to $c$, as shown in Figure 2. Since we have $d(c, v_i) = B_T/2$ by the definition of $c$ and we assume $B < B_T$, such an edge $e$ exists. Let $\beta = \min\{\alpha, r^*(e)\}$.

Consider the reduction $R'$ obtained from $R^*$ by decreasing the reduction on edge $e$ by $\beta$ (i.e., $r'(e) = r^*(e) - \beta$) and increasing the reduction on edges on the path from $c$ to $v_j$ by $\beta$. Reduction $R'$ performs this increase on edges as close to vertex $c$ as possible. The increase can occur on a single edge or on a number of edges. The remaining reductions of $R'$ are as in $R^*$. We claim that $R'$ is also an optimal reduction. Since its cost is equal to that of $R^*$, we only need to show that every path in $T_{R'}$ has length at most $B$.

Assume $T_{R'}$ contains a path of length greater than $B$. Any such path must contain edge $e$ and it cannot contain all the edges on path $P(c, v_j)$ that have received an additional reduction.
Figure 2: Two possibilities for path $P(x,v_i)$ when $d_{R'}(c,v_j) = B/2 + \alpha$.

Let $P(x,y)$ be a path with $d_{R'}(x,y) > B$. Choose $y$ such that $P(y,v_i)$ does not contain edge $e$. We can assume that $y = v_i$ since (i) no edge on the path from $b$ to $v_i$ is reduced, (ii) $v_i$ is a leaf in $T_p$, and (iii) no path from $b$ to a vertex $w$ not containing edge $e$ can have length exceeding $d_{R'}(b,v_i) = d(b,v_i)$. We distinguish between three possible positions for vertex $x$:

- Path $P(x,v_i)$ does not contain center vertex $c$. Figure 2(a) illustrates this case. Since $d_{R'}(c,v_i) \leq B/2 - \alpha$, $d_{R'}(c,x) \leq B/2 - \alpha$, and the weight of edge $e$ increases by $\beta \leq \alpha$, we have $d_{R'}(x,v_i) \leq B - \alpha$. This contradicts our assumption that path $P(x,y)$ exceeds cost $B$ in reduction $R'$.

- Path $P(x,v_i)$ does contain $c$, but no edge on the path from $c$ to $v_j$ is in $P(x,v_i)$. We then also have $d_{R'}(c,y) \leq B/2 - \alpha$ and contradict our assumption.

- Path $P(x,v_i)$ contains at least one edge on the path from $c$ to $v_j$. (Observe that it cannot contain all edges having received additional reduction.) Recall that $R'$ assigned the additional reduction of cost $\beta$ to edges as close to $c$ as possible. Assume the path from $c$ to $x$ experiences $\beta_1$ of the reassigned reduction, $\beta_1 < \beta$. Let $z$ be the lowest common ancestor of $x$ and $v_j$, as shown in Figure 2(b). Since $d_{R'}(c,z) = \beta_1$ and $d_{R'}(c,z) = 0$, we have $d_{R'}(z,x) = d_{R'}(z,x) \leq B/2 - \alpha + \beta_1$, and thus

$$d_{R'}(x,v_i) = d_{R'}(c,v_i) + d_{R'}(z,x) \leq (B/2 - \alpha + \beta) + (B/2 - \alpha + \beta_1) < B.$$ 

This contradicts our assumption.
It thus follows that reassigning a reduction of $\beta$ from edge $e$ to the specified edges on the path $P(c, v_j)$ does not create a path of length greater than $B$ in $T_{R'}$. Thus, $R'$ is an optimal solution. If $R'$ satisfies the lemma, we are done. Otherwise, we apply the same argument to reduction $R'$. 

The second part of the proof considers the case when $d_{R'}(c, v_i) \leq B/2$ for all leaves $v_i$ in $T_p$ and there exists at least one leaf $v_j$ with $d_{R'}(c, v_j) < B/2$. Choose $v_j$ such that $d_{R'}(c, v_j) = B/2 - \epsilon$, and $\epsilon$ is a maximum over all vertices with $d_{R'}(c, v_i) < B/2$.

Let $e$ be the edge on path $P(c, v_j)$ closest to vertex $v_j$ which has a reduction in $R^*$. Let $\beta = \min\{\epsilon, r^*(e)\}$. Consider reduction $R'$ obtained from $R^*$ by decreasing the reduction on edge $e$ by $\beta$. Since $R^*$ is an optimal reduction, $T_{R'}$ must contain a path of length greater than $B$. Let $P(x, y)$ be such a path (which contains edge $e$). As already argued, we can assume that $y = v_j$ (since there are no reductions on the edges between $v_j$ and edge $e$). Further, we know that $x$ does not correspond to a leaf vertex in $T_p$. If $x$ were a leaf in $T_p$, $d_{R'}(c, x) \leq B/2$ by assumption. This implies that the length of path $P(x, v_j)$ in $R'$ could not exceed $B$.

To reduce the maximum path length in $T_{R'}$ to $B$, we make reduction increases totaling $\beta$ to edges in $R'$. When path $P(x, v_j)$ does contain vertex $c$, we increase reductions on edges on the path from $c$ to $x$ and as close to vertex $c$ as possible. When path $P(x, v_j)$ does not contain vertex $c$, let $z$ be the lowest common ancestor of $v_j$ and $x$. We then increase reductions on edges on the path from $z$ to $x$ and as close to vertex $z$ as possible. Making this change to reduction $R'$ results in a new reduction $R'$ having the same cost as $R^*$. New reduction $R'$ does not contain a path exceeding length $B$. If $T_{R'}$ would contain a path exceeding cost $B$, such a path would be between some vertex $w$ and vertex $v_j$, it would contain edge $e$, but not all edges which received reduction increases. The existence of such a path implies the existence of a path of length greater than $B$ between $w$ and $x$ in $T_{R'}$, which is not possible. Hence, $R'$ is an optimal solution. As stated for the first case, if $R'$ satisfies the lemma, we are done. Otherwise, we repeat the argument of reassigning reductions. After a finite number of iterations, we converge to the desired reduction. \hfill \Box

**Lemma 2** There exists an optimal reduction $R$ with $d_R(c, x) \leq B/2$ for every leaf $x$ in $T$.

**Proof:** We only need to consider leaves $x$ which are not in $T_p$ (since the previous lemma covers the leaves in $T_p$). Assume there exists a leaf $x$ with $d_R(c, x) > B/2$. There exists at least one
leaf \(v_p\) in \(T_p\) such that \(P(v_p, x)\) contains vertex \(c\). Since \(d_{R'}(c, v_p) = B/2\), the path from \(v_p\) to \(x\) in \(R\) has cost > \(B\), contradicting the assumption that \(R\) is an optimal solution. \(\square\)

Let \(u_1, \ldots, u_k\) be the vertices adjacent to center \(c\) in \(T_p\), \(k \geq 2\). Assume \(0 < w(c, u_i) \leq w(c, u_1), 1 \leq i \leq k\). If \(w(c, u_1) = 0\), we make the children of \(u_1\) children of \(c\) and continue with the resulting tree. Let \(B_{sl}\) be the length of a second longest path from \(c\) to a leaf in \(T\), \(B/2 < B_{sl} < B_T/2\). The next lemma partially characterizes where reductions take place in an optimal solution by giving a lower bound on reductions placed on the edges in \(T_p\) incident to \(c\). In this characterization, only second longest paths from \(c\) to a leaf which are edge disjoint from longest paths are of interest. Hence, if for every second longest path the edge incident to vertex \(c\) is in \(T_p\), we set \(B_{sl} = 0\). Using this definition of \(B_{sl}\), let

\[
\delta = \min\{B_T/2 - B_{sl}, w(c, u_1)\}.
\]

For the tree shown in Figure 1 we have \(B_T = 30\) and \(B_{sl} = 0\). The two second longest paths from \(c\) to a leaf have cost 13, but each one of them contains an edge incident to \(c\) in \(T_p\). We thus have \(\delta = \min\{15, 3\} = 3\).

**Lemma 3** There exists an optimal reduction \(R\) with \(r(c, u_i) \geq \delta\) for every \(i, 1 \leq i \leq k\).

**Proof:** Let \(R^*\) be an optimal reduction not satisfying Lemma 3, but satisfying Lemma 1. In some sense, we generate the reduction satisfying Lemma 3 recursively as well as incrementally. Assume first that all edges incident to \(c\) have \(r^*(c, u_j) > 0\) and at least one edge \((c, u_i)\) has \(r^*(c, u_i) = \gamma < \delta\). Choose edge \((c, u_i)\) so that \(\gamma\) is a minimum. Generate from \(T\) a new tree \(T'\) with \(w'(c, u_j) = w(c, u_j) - \gamma, 1 \leq j \leq k\), and \(w'(u, v) = w(u, v)\) for all other edges of \(T\). Tree \(T'\) has a diameter of \(B_T - 2\gamma\). Next, obtain an optimal reduction \(R'\) satisfying Lemma 3 for tree \(T'\). Reduction \(R'\) can be turned into an optimal reduction for \(T\) satisfying Lemma 3 by increasing the reduction on the edges which experienced a weight decrease.

The process described above cannot be applied when there exists an edge \((c, u_i)\) with \(r^*(c, u_i) = 0\); i.e., edge \((c, u_i)\) has no reduction in \(R^*\). We then generate from \(R^*\) an optimal reduction \(R'\) with \(r^*(c, u_i) > 0\). Assume \(r^*(c, u_i) = 0\) and let vertex \(v_j\) be a leaf in \(T_p\) reachable from \(u_i\) without going through vertex \(c\). See Figure 3 for an illustration. Let \(e = (a, b)\) be the edge closest to vertex \(v_j\) with \(r^*(e) > 0\). Let \(\beta = \min\{\delta, r^*(e)\}\). We next generate reduction \(R'\)
Figure 3: Illustration of a path with $d_{R'}(x, v_j) > B$.

from $R^*$ by moving a reduction of cost $\beta$ from edge $e$ to edge $(c, u_i)$. Clearly, $R'$ and $R^*$ have the same cost. If every path in $T_{R'}$ has length at most $B$, then $R'$ is an optimal reduction.

Assume thus that $T_{R'}$ contains a path $P(x, y)$ with $d_{R'}(x, y) > B$. Path $P(x, y)$ contains edge $e$, but not $(c, u_i)$. Choose $y$ such that $P(y, v_j)$ does not contain edge $e$. We can again assume that $y = v_j$. Let vertex $z$ be the lowest common ancestor of vertices $x$ and $v_j$ in $T$. We then have

$$d_{R'}(z, v_j) \leq B/2 - \beta \text{ and } d_{R'}(z, v_j) \leq B/2$$

and thus

$$d_{R'}(z, x) = d_{R'}(z, x) > B/2.$$ 

This implies that there exists a leaf $x$ with $d_{R'}(c, x) > B/2$. By Lemma 2 this cannot happen and thus $R'$ is an optimal reduction.

If $\beta = \delta$, reduction $R'$ is an optimal reduction satisfying Lemma 3. Otherwise, we repeat the above procedure. If there exists an edge with $r'(c, u_i) = 0$, we repeat the process of reassigning reductions. If all edges incident to $c$ have nonzero reduction, we continue with a tree whose edges incident to $c$ have smaller weights. Generating the desired reduction for this tree allows us to generate the desired reduction for the original tree. $\square$

3 Determining an optimal reduction

We now describe the $O(n)$ time algorithm to determine, from a given tree $T$ and a target broadcasting cost $B$, a reduction $R^*$ such that $T_{R^*}$ has broadcasting cost at most $B$ and
$M(T_R)$ is a minimum. The first step of the algorithm is to determine center vertex $c$ and to root the tree at $c$. To determine $c$, first root $T$ at some arbitrary vertex, say $r$. Then, sweep up from the leaves of $T$ towards root $r$ and determine the length of the longest path between any two vertices. This quantity is $B_T$. Next, use this path having length $B_T$ to identify or create center vertex $c$. As described in Section 2, $c$ is unique and it is the center vertex for all other longest paths. We then root tree $T$ at $c$ and determine for every subtree rooted at a vertex $v$ the quantity $L(v)$, the length of the longest path from a leaf (in the subtree) to vertex $v$. We initialize $r^*(u,v) = 0$ for every edge $(u,v)$ in $T$. These preprocessing steps can be done in $O(n)$ time.

We first sketch an iterative approach for generating $R^*$ based directly on the characterization of an optimal reduction given in Lemma 3. The resulting algorithm alters $T$ and works with $T_p$. We then describe how to translate this approach into an $O(n)$ time algorithm which does not explicitly generate subtree $T_p$, but works with the computed $L(\cdot)$-entries. Let $u_1, u_2, \ldots, u_k$ be the vertices adjacent to $c$ in subtree $T_p$. Let $\delta$ be the quantity defined before Lemma 3. We then set $r^*(c, u_i) = r^*(c, u_i) + \delta, 1 \leq i \leq k$. After these reductions are performed, one of two events must have occurred.

**Event 1:** An edge $(c, u_i)$ has weight 0. In this case, future reductions have to be performed on other edges in the subtree rooted at $u_i$. Let $a_1, a_2, \ldots, a_l$ be the vertices adjacent to vertex $u_i$ in $T_p$ (excluding vertex $c$). We delete the edges $(c, u_i), (u_i, a_1), \ldots, (u_i, a_l)$ from $T$ and $T_p$ and add the edges $(c, a_1), \ldots, (c, a_l)$ with $w(c, a_i) = w(u_i, a_i)$. In addition, we record that the actual endpoints of every edge $(c, a_i)$ are $u_i$ and $a_i$ and that reductions made on edge $(c, a_i)$ correspond to reductions on edge $(u_i, a_i)$. We then continue with the new $T$ and $T_p$.

**Event 2:** New longest paths have emerged. In this case all edges $(c, u_i)$ still have positive weights, but the weight decrease on the edges incident to $c$ resulted in a new tree $T_p$; i.e., edges of $T$ which were not on a longest path before the reduction are now on a longest path. The new edges in $T_p$ include an edge incident to vertex $c$. This holds since a new longest path contains vertex $c$ and a previously unreduced edge incident to $c$. Hence, after the reductions on the edges incident to $c$ have been made and no edge has weight 0, we continue with new tree $T_p$ in which the number of edges incident to $c$ receiving a reduction increased.

For the tree shown in Figure 1(b), we initially have $B_T = 30$ and $\delta = 3$. The first iteration decreases the weights of edges $(c, u_1)$ and $(c, u_2)$ from 3 to 0. Then, an event 1 is processed.
The next iteration reduces the two incoming edges of vertex $u_1$ and edge $(u_3, u_2)$.

\[
\begin{align*}
(a) & \text{ tree } T \text{ rooted at } c \\
(b) & \text{ reduced tree for } B=18
\end{align*}
\]

Figure 4: Tree $T$ with $B_T = 30$, $B = 18$, $L(\cdot)$-entries are next to vertices in (a)

Using the approach driven by acting on either event 1 or event 2, the following $O(n)$ time algorithm emerges. First, observe that the edges receiving a reduction form a tree containing vertex $c$. Further, if edge $(u, v)$ receives a reduction and $v$ is the parent of $u$, then every edge on the path from $v$ to $c$ receives a reduction equal to the weight of the edge. If $(u, v)$ receives a reduction with $r^*(u, v) < w(u, v)$, then no edge into vertex $u$ receives a reduction. This allows us to generate an optimal reduction $R^*$ by determining the reduction on every edge $(u, v)$ directly using:

\[
r^*(u, v) = \begin{cases} 
 w(u, v) & \text{if } B/2 \leq L(u) \\
 L(u) + w(u, v) - B/2 & \text{if } L(u) < B/2 < L(u) + w(u, v) \\
 0 & \text{otherwise.}
\end{cases}
\]

Figure 4(a) shows the tree from Figure 1 with the $L(\cdot)$-entry for each vertex. For edge $(u_2, c)$, we have $B/2 = 9 < 12 = L(u_2)$ and thus edge $(u_2, c)$ receives a reduction of 3 (which corresponds to a weight of 0). For edge $(u_3, u_2)$, we have $L(u_3) = 6$, $6 < 9 < 12$, and edge $(u_3, u_2)$ receives a reduction of $L(u_3) + w(u_3, u_2) - B/2 = 3$. Figure 4(b) shows the weights of the reduced tree achieving diameter 18.
After preprocessing, the reduction for one edge can be determined in $O(1)$ time. This results in overall $O(n)$ time. We point out that the algorithm determines the reductions as done in the problem of finding edge reduction minimizing the longest path length in trees described in [4]. However, the arguments as to why the approach is correct are different for the two problems. From our discussion, we conclude:

**Theorem 4** Given an $n$-vertex, weighted tree $T$ and a target broadcasting cost $B$, we can determine a minimum cost reduction $R^*$ with $R^*_n = B$ in $O(n)$ time.

4 Acknowledgements

We would like to thank the referees for their helpful comments.

References


