Computer Science and Systems Analysis

Computer Science and Systems Analysis

Technical Reports

Miami University *Year* 2008

Degree-constrained Minimum Latency Trees are APX-Hard

Bo Brinkman* Michael Helmick†

*Miami University, brinkmwj@muohio.edu

†

This paper is posted at Scholarly Commons at Miami University.
http://sc.lib.muohio.edu/csa_techreports/76

MIAMI UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE & SYSTEMS ANALYSIS

TECHNICAL REPORT: MU-SEAS-CSA-2008-002

Degree-constrained Minimum Latency Trees are APX-Hard

Bo Brinkman and Michael T. Helmick

School of Engineering & Applied Science | Oxford, Ohio 45056 | 513-529-5928

Degree-constrained Minimum Latency Trees are APX-Hard

Bo Brinkman and Michael T. Helmick
Miami University

Abstract

When transmitting data from a single source to many recipients, it is often desirable to use some recipients of the stream to re-broadcast the stream to other users. In such multicast systems each client may be used as a server, serving up to B other clients.

Formally, we will take a set X of clients, along with a distance function d that specifies the latency (in the host network) between each pair of clients in X. Our goal will be to produce a directed spanning tree of X, rooted at some specified $root \in X$, with out-degree bounded by B, and minimizing the sum of the latencies from $root$ to every point in X.

In addition to being motivated by current experimental algorithms work, the problem also interpolates naturally between the traveling repairman problem (when $B = 1$) and single source shortest paths (when $B = n - 1$). The former problem is APX-complete (in metric spaces) and the latter is in P. We explore the hardness of the problem for other values of B. In particular, we show that the problem remains APX-Hard at least up to $B = C\sqrt{n}$ for some universal constant C when the host space is a general semi-metric.

1 Introduction

We consider multicasting, the process of partitioning server access by using participants in the transmission as relay points for other participants. This implicitly creates a directed, rooted spanning tree as an overlay on the original network. Data transmissions are sent from the $root$ to its direct children, who then retransmit to their children, repeating until all clients have been reached. This alleviates the demand required for a large amount of subscribers to receive information directly from the origin server.

Multicasting ability is built into the Internet Protocol (IP) specification as described by Deering [7]. However, IP multicasting has not seen widespread adoption, and research attention has focused on application level multicasting [6, 8, 9, 5]. Application-level multicasting uses unicast, point-to-point, messages to power multicast delivery from within a particular application or application framework.

There are several properties, or measures of efficiency, that are desirable for multicast systems of this type:

1. Each client should have bounded out-degree. This is because an individual client may be quite limited in computational power or bandwidth.
2. The total latency experienced by the whole network should be small (in some sense).
3. The congestion incurred in the host network should be small (because of limitations in the capacity of network routers).

In this paper we will focus only on the first two items. The problem of finding bounded degree spanning trees with low total latency is already a difficult computational problem. In addition, it naturally interpolates between two well-studied problems: The traveling repairman problem and the single source shortest path problem.

Throughout this paper we will consider the problem of constructing an overlay network built on top of some host network. In order to get meaningful results, we focus on the case where the latencies between hosts form a semi-metric.

DEFINITION 1. (SEMI-METRIC) *A function $d : X \to [0, \infty)$ (where X is a set) such that*

1. $\forall x, y \in X$, $d(x, y) = d(y, x)$
2. $\forall x, y \in X$, $d(x, y) \geq 0$. *The fact that it is possible for $x \neq y$ to have $d(x, y) = 0$ is the reason that d is not a metric.*
3. $\forall x, y, z \in X, d(x, z) + d(z, y) \geq d(x, y)$

It is valid for any node to connect "directly" to any other node using the underlying host network. The latency experienced by an edge (x, y) is just $d(x, y)$. Note that, in general, latencies in a real network tend to vary over time, and need not be symmetric. Even for this very simplistic model of network latencies, however, we are able to get some inapproximability results. Inspired by Helmick and Annexstein [12], we define:

DEFINITION 2. (B-MLTREE) *Given a finite semi-metric space $M = (X, d)$, where all distances integers represented by at most $c_1 n^{c_2}$ bits, and a designated source point $root \in X$, find a directed spanning-tree T of X which:*

- *Is rooted at $root$*
- *Has out-degree $\leq B$*
- *Minimizes $\sum_{x \in X} d_T(root, x)$, where d_T is the shortest-path distance in the tree T.*

Note that the restriction to distances that are polynomially representable integers only strengthens our result.

1-MLTREE corresponds to the traveling repairman problem, which is also known as the minimum latency problem (MLP) or the minimum latency tour problem (MLT). It is well known that 1-MLTREE is APX-Complete, a fact that follows from the proof of Papadimitriou and Yannakakis [15] that TSP is APX-Complete when all distances are one or two.

We explore the complexity of B-MLTREE for larger values of B. In particular, even poly-log dependence on n in the degree might be acceptable in many applications. For this reason we are interested in algorithms and hardness results for all B, not just $B = \Theta(1)$.

1.1 Previous work The B-MLTREE problem incorporates elements of single source shortest paths, minimum latency tours, and bounded-degree spanning trees.

The largest body of work on this class of problems is on 1-MLTREE. The first polynomial-time approximation algorithm for the minimum latency problem was given by Blum et al [4]. They point out that the APX-Hardness of finding a minimum latency tour follows from the APX-Hardness of TSP, and they give a poly-time algorithm based on TSP approximators. The approximation was subsequently improved by Goemans and Kleinberg [10] and Archer and Williamson [1].

Arora and Karakostas [2] also considered the problem of creating approximation schemes for the MLP. Because the problem is APX-Hard, it is unlikely that a poly-time approximation scheme exists: Instead, they provide $(1 + \varepsilon)$-approximation algorithms with time roughly $n^{O(\lg n/\varepsilon)}$.

Of course, at the other end of the spectrum, $(n - 1)$-MLTREE may be solved trivially in polynomial time by building a star with all points directly connected to the root.

Outside of the theoretical community there has been a significant amount of work to develop heuristic algorithms for B-MLTREE. For example, Banerjee et al [3] give an algorithm (called OMNI) that works well in practice, and also gives a $(\log n)$-approximation for 2-MLTREE. Helmick and Annexstein [12], borrowing ideas from the minimum-diameter spanning tree algorithm of Könemann et al [13], improve on OMNI by making better use of clusters of multicast clients.

Some recent advances in bounded-degree network design problems may provide insight as to where to go next. Goemans [11] gave a multi-criteria formulation of the degree-bounded minimum spanning tree problem. He showed that, rather than finding a sub-optimal degree B tree, he could instead find a tree with

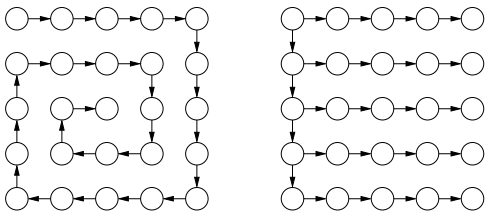

Figure 1: A $\sqrt{n} \times \sqrt{n}$ ℓ_1 grid with a 1-MLTREE and 2-MLTREE

weight as good as the optimal degree B tree, but with degree B+2. This was then improved and simplified by Mohit Singh et al [16, 14]. These authors, in various combinations, were able to reduce the degree to B+1, and also develop bi-criteria approximation algorithms of this type for other network construction problems.

1.2 Choice of latency objective We should take a moment to scrutinize our choice of latency objective. Different clients will experience different amounts of latency, and it is not completely clear how the "latency" of the whole network should be computed, based on the latencies of individual clients. Various authors have tried to minimize:

1. The total (or average) latency over all nodes in the tree.
2. The maximum latency experienced by any single node, which is related to the diameter.
3. The maximum multiplicative factor increase in latency for a single node.

We focus on the total latency summed over all nodes in the tree, which allows us to make the link to traveling repairman directly. This is a relaxation of the third option, minimizing the maximum "stretch" incurred by any single node, and several of our algorithms actually do minimize the maximum stretch.

The second choice is more "fair" than the first, because it bars one from building a network that excessively penalizes one user in order to benefit others just a little bit. For the specific case of multicasting at the application level, however, this characterization of the problem misses some important cases. For example, if there are a few clients that are very far away from the rest, the algorithm does not have to do anything very useful.

1.3 Our results In this tech report, we show that B-MLTREE remains APX-Hard on semi-metrics as long as $B < C\sqrt{n}$, for some universal constant C.

2 2-MLTREE is NP-Complete

Though it is fairly well known that 1-MLTREE does not admit a PTAS unless $P = NP$, it does not follow that B-MLTREE is hard for all other values of B. As we have seen, for B close to n it can be solved exactly in polynomial time. On the other hand, it is also not clear whether or not 2-MLTREE even has a constant approximation. As B grows, the algorithm becomes more powerful, because it has a wider set of allowable out-degrees. At the same time, the optimal latency can also improve very rapidly as the allowed degree goes up, putting the optimal latency farther out of reach. For example, consider points in the ℓ_1 plane. For the metric depicted in Figure 1, the optimal latency degree 1 tree (which is also a MST of the metric) has total latency roughly n^2. The degree 2 tree pictured has total latency roughly $n^{3/2}$, which is optimal (up to constant factors) for all $B \geq 2$. Therefore, even an optimal solution for B-MLTREE might be very far from optimal for $(B+1)$-MLTREE.

A priori, it is not clear how the hardness of approximation depends on B. In what follows we establish that the problem stays NP-Complete (and also APX-Hard) even for quite large B. B-MLTree is clearly in NP for all values of B.

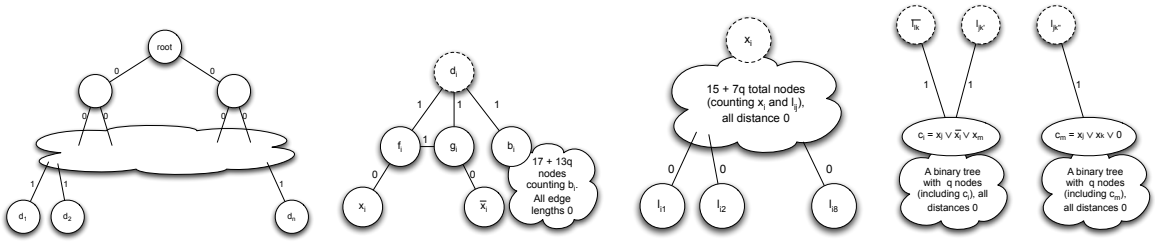

Figure 2: Top, decider, distributor and clause constructions

As a warm up, we will prove that 2-MLTREE is NP-Hard. Our proof of APX-Hardness will follow from this same reduction, after a bit of extra accounting. We reduce from MAX-3-SAT(13). Recall that MAX-3-SAT(13) is the problem of maximizing the number of satisfied clauses in a 3-CNF where each variable can occur (negated or not) in at most 13 clauses.

THEOREM 2.1. $MAX-3-SAT(13) \leq_P$ 2-MLTREE

In proving this theorem, we develop a fairly simple direct reduction from MAX-3-SAT(13). In contrast, the result of Papadimitriou and Yannakakis [15] proving that TSP with distances one and two is APX-Hard is really a reduction from a form of MAX-3-SAT to HAMILTONIAN PATH. The fact that 1-MLTREE is also APX-Hard is a corollary of their result, and does not follow from our reduction.

First, to simplify our construction, let us assume that there are no trivially satisfied clauses. If there are, we will just delete them, and keep a count of how many such clauses were removed. Also remove all clauses of the form $(0 \vee 0 \vee 0)$.

Given a 3-CNF(13) representation of a boolean formula, we will build a graph, and show that its minimum latency achievable by a degree 2 tree depends exactly on the maximum number of satisfiable clauses.

In what follows, every edge has length 0, unless otherwise noted. Let the variables of the boolean formula be called x_1 up to x_n. The construction of the graph proceeds as follows:

1. Create a defined $root$ node
2. Create a balanced binary tree rooted at $root$ with exactly n children, labeled d_i, where each d_i is associated with the variable x_i. All edges in this sub-tree have length 0, except that the in-coming edges for each d_i have length 1. (See Figure 2)
3. For each d_i, add a "decider" widget rooted at d_i (See Figure 2). Note that q, which also appears in the clauses (below), is a constant that will be selected later to tune the proof.
4. For each node labeled x_i or $\overline{x_i}$, add a distributor widget. These are balanced binary trees with 8 leaves. If the root is x_i, the leaves are labeled l_{i1} up to l_{i8}. If the root is $\overline{x_i}$, they are labeled $\overline{l_{i1}}$ up to $\overline{l_{i8}}$. Since each variable appears at most 13 times, these 8 nodes will be sufficient to connect the variable to its clauses. (See Figure 2)
5. For each clause j, add a new node labeled c_j. If a clause j contains a term x_i, add an edge from l_{ik} to c_j, for some k such that l_{ik} does not yet have two children. If it contains a term $\overline{x_i}$, add an edge from $\overline{l_{ik'}}$. Note again that q will be chosen later to tune the proof.

For a particular boolean formula F (after the pre-processing above), let us call this graph $G(F)$.

In order to get a truth assignment from a spanning tree of this graph, we will look at the latency of nodes in the sub-tree rooted at x_i. If x_i has latency 2 (which is the minimum possible), then the variable x_i will be considered to be true. Otherwise, consider the variable x_i to be false.

$(n-1) \times 0$	(Inner nodes of "top")
$n \times 1$	(d_i nodes)
$n(49+27q) \times 2$	($F_i, G_i,$ and B_i)
$mq \times 3$	(Clauses)

Figure 3: Minimum latency without degree constraints

$(n-1) \times 0$	(Inner nodes of "top")
$n \times 1$	(d_i nodes)
$n(33+20q) \times 2$	(B_i and either F_i or G_i)
$n(16+7q) \times 3$	(F_i or G_i)
$sq \times 3$	(Satisfied clauses)
$(m-s)q \times 4$	(Unsatisfied clauses)

Figure 4: Latency of the proposed tree

The body of the proof has been moved to an appendix due to space considerations, but, in order to give some intuition, we will describe what a degree-2 minimum latency tree must look like in this graph. Let F_i denote the set of nodes with distance 0 to f_i in G(F), and similarly for G_i, B_i, and C_j.

LEMMA 2.1. *Let T be any minimum latency degree-2 spanning tree of such a graph G(F). Then:*

1. *T must include a binary tree on the nodes of "top," with the d_i being the leaves*
2. *For each node d_i, one of its children must be a member of B_i, and the other a node from either F_i or G_i. If it is a node of F_i that is the child of d_i, then some node in F_i takes a node from G_i as one of its children. The case where a node of G_i is the child of d_i is symmetric.*
3. *All nodes in B_i have the same latency as b_i. The same is also true for F_i and f_i, G_i and g_i, and C_j and c_j.*
4. *For each C_j, the incoming edge to C_j must come from some F_i or G_i such that x_i (negated or not) appears in clause j.*

The proof of this lemma is a fairly standard application of case analysis and induction. Refer to the Appendix for details.

Now we will prove some useful facts about the graph $G(F)$, and about the proposed degree-2 tree. Let m denote the number of clauses, and let s denote the number of clauses satisfied by T. In Figure 3 we calculate the minimum latency of any spanning tree of G(F), and in Figure 4 we calculate the latency of the tree proposed in Lemma 2.1. Notice that the total latency incurred by our degree-2 solution, over the necessary minimum, is just $n(16+7q)+(m-s)q$.

The following lemma is a corollary of Lemma 2.1.

LEMMA 2.2. *A 3-CNF(13) formula F has an assignment that satisfies at least s clauses if and only if the graph G(F) has a degree-2 spanning tree with latency at most $115n+61qn+4mq-sq$.*

Proof. If we have an assignment that satisfies s clauses, we can use it to pick a tree with the required latency. In our description of the "proposed" structure of the optimal tree, just choose whether F_i or G_i should be served by d_i based on whether or not x_i is set to true. For each clause c_j, choose its parent to be whichever of its terms has lowest latency. It is easy to see that this leads to the guaranteed latency.

If we have an optimal spanning tree, it can also be used to construct an assignment that satisfies at least s clauses. Lemma 2.1 shows that for each x_i, exactly one of F_i and G_i has latency 2. If we set x_i to true if and only if some node of F_i has latency 2, then this assignment must satisfy at least s clauses, because this is the only way that the latency of a c_j node can be decreased to 3 in a latency optimal tree.

3 2-MLTREE is APX-Hard

In the previous section, we proved that if the total latency of the optimal degree-2 spanning tree of $G(F)$ is $\leq 115n+61nq+4mq-sq$, then there is an assignment that satisfies at least s clauses of F. The existence

$$
\begin{aligned}
(1+\varepsilon)(115n+61nq+4mq-sq) &= 115n+61nq+4mq-tq \\
(\varepsilon)(115n+61nq+4mq)-(1+\varepsilon)sq &= -tq \\
(1+\varepsilon)-(\varepsilon)((115n/sq)+61n/s+4m/s) &= t/s \\
1+\varepsilon-(\varepsilon)(690/q+366+8) &\leq t/s \\
1-375\varepsilon &\leq t/s
\end{aligned}
$$

Figure 5: Derivation of approximation ratio for proposed MAX-3SAT(13) algorithm.

of a polynomial-time approximation scheme (PTAS) for 2-MLTREE also implies the existence of a PTAS for MAX-3-SAT(13). In other words,

THEOREM 3.1. *MAX-3-SAT(13)* $\leq_{PTAS}$ 2-MLTREE

Consider any $(1+\varepsilon)$-approximation for 2-MLTREE. It is still true that if we have a latency $L = 115n+61nq+4mq-tq$ spanning tree, we know that there is an assignment that satisfies at least t clauses of F. In Figure 5 we solve for t in the case that we have a $(1+\varepsilon)$-approximation algorithm for 2-MLTREE. In the fourth line we use the fact that $m/s \geq 2$ and that $n \leq 3m$.

The fact that at least half the clauses must be satisfied is well known: Just take each variable, one at a time. Look at all remaining clauses where it appears, and decide whether or not to set it to true based on which value satisfies the most clauses. Then throw away all of the clauses that contained that variable, and continue with the next variable. This guarantees that we satisfy at least half of the clauses. In the last line we chose $q = 690$ just to drive down the final constant: It is not particularly significant, and could be set to any constant ≥ 1.

Given a degree 2 spanning tree T of $G(F)$ that has latency within a $(1+\varepsilon)$ factor of optimal, we construct the satisfying assignment for the x_i essentially by following the fix-ups given in the proof of Lemma 2.1, in the Appendix. In the proof we describe how to convert any spanning tree T into one that has all points in F_i having the same latency as f_i and all points in G_i having the same latency as g_i. We also describe how to transform the graph so that exactly one of F_i and G_i has latency $= 2$. All these transformations can be applied without increasing latency, and in polynomial time. As a result, we can convert T into a new tree T' such that for each i exactly one of F_i and G_i has latency 2. If it is F_i, then x_i is set to true. If it is G_i, then x_i is set to false. This is guaranteed to satisfy at least $t = (1-375\varepsilon)s$ clauses of the formula F, or else a latency of $115n+61nq+4mq-tq$ would not have been possible.

Hence, if a $(1+\varepsilon)$-approximation exists for 2-MLTREE, then a $(1-375\varepsilon)$-approximation exists for MAX 3-SAT(13). This cannot happen unless $P = NP$.

3.1 Extensions to higher B Every part of this construction generalizes to the case of $B > 2$. The construction for "top" is mostly unchanged. The only problem is that the number of available out-going edges from top will be $(B-1)p+1$, where p is the number of nodes in the top. It is quite possible that the number of variables in the formula cannot be written in such a form, resulting in some d_i that do not correspond to a variable. These d_i still must have their associated decider widgets attached in order for the construction to work. The number of such "useless" deciders is at most $B-2$. Because $B < n$, this only changes the coefficient of ε in the approximation factor, and the result still holds.

The other main difference is that each d_i will now have $B+1$ children, and $B-1$ of them will have a sub-tree of $17+13q$ nodes, like b_i in the original construction.

Notice that we cannot take this construction too far. The total number of vertices in the graph is $\Theta((n+B)B+m)$. In other words, where $N = |V|$, there is some constant $c > 0$ such that $(c\sqrt{N})$-

B	Hardness	Best approximation algorithm
1	APX-Hard (Corollary of [15])	$O(1)$ [4, 10, 1]
2 to $\Theta(\sqrt{n})$	APX-Hard (This paper)	$O(\lceil \log_B n \rceil)$ [3, 12]
$n^{1/c}, c < 2$	?	O(1) ([3, 12])
$n - 1$	P	1 (Star graph)

Table 1: Table of known results about B-MLTREE on general semi-metrics.

MLTREE is APX-Hard. Beyond this point our proof no longer shows anything. We conclude this section with a summary of known results for B-MLTREE in general semi-metrics, which is presented in Table 1.

4 Conclusion and open problems

In this tech report we have given the first inapproximability results for the problem of building bounded degree low latency spanning trees. We have shown that this problem is unlikely to admit polynomial time approximation schemes, at least for practically useful degree bounds. Given our results, however, there are a number of directions of research that might still greatly benefit practitioners that must build multicast trees. Let us conclude with some specific problems that might be of practical use, even though B-MLTREE is APX-Hard in general:

1. Either give a poly-time constant-factor approximation for c-MLTREE (for some fixed constant $c \geq 2$), or prove that such an algorithm cannot exist, assuming $P \neq NP$.
2. Given that there are most likely no poly-time approximation schemes for constant B, is it possible to give a Goemans-style [11, 16, 14] bi-criteria approximation, where the output tree has degree $\alpha B + \beta$ and total latency at most $(1 + \varepsilon)OPT(B)$? Notice that our result does not preclude this possibility, because even increasing the allowed degree by 1 can change the minimum latency by a factor polynomial in n.
3. Are there any reasonably realistic network models that admit polynomial time solutions, or polynomial time approximation schemes?

References

[1] Aaron Archer and David P. Williamson. Faster approximation algorithms for the minimum latency problem. In *SODA '03: Proceedings of the fourteenth annual ACM-SIAM Symposium on Discrete Algorithms*, pages 88–96, Philadelphia, PA, USA, 2003. Society for Industrial and Applied Mathematics.

[2] Sanjeev Arora and George Karakostas. Approximation schemes for minimum latency problems. In *STOC '99: Proceedings of the thirty-first annual ACM Symposium on Theory of Computing*, pages 688–693, New York, NY, USA, 1999. ACM Press.

[3] S. Banerjee, C. Komareddy, K. Kar, B. Bhattacharjee, and S. Khuller. Construction of an efficient overlay multicast infrastructure for real-time applications. In *Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies*, pages 1521–1531, 2003.

[4] A. Blum, P. Chalasani, D. Coppersmith, B. Pulleyblank, P. Raghavan, and M. Sudan. The minimum latency problem. In *STOC '94: Proceedings of the twenty-sixth annual ACM Symposium on Theory of Computing*, pages 163–171, New York, NY, USA, 1994. ACM Press.

[5] M. Castro, P. Druschel, A-M. Kermarrec, and A. Rowstron. Scribe: A large-scale and decentralised application-level multicast infrastructure. In *IEEE Journal on Selected Areas in Communications (JSAC) (Special issue on Network Support for Multicast Communications)*, 2002.

[6] Woan Sun Chang and Robert Simon. End-host multicasting in support of distributed real-time simulation systems. In *ANSS '04: Proceedings of the 37th Annual Symposium on Simulation*, page 7, Washington, DC, USA, 2004. IEEE Computer Society.

[7] S. E. Deering. Multicast routing in internetworks and extended lans. In *SIGCOMM '88: Symposium proceedings on Communications Architectures and Protocols*, pages 55–64, New York, NY, USA, 1988. ACM Press.

[8] C. Diot, B. N. Levine, B. Lyles, H. Kassem, and D. Balensiefen. Deployment issues for the IP multicast service and architecture. In *IEEE Network Magazine*, January/February 2000.

[9] Aditya Ganjam and Hui Zhang. Connectivity restrictions in overlay multicast. In *NOSSDAV '04: Proceedings of the 14th international workshop on Network and Operating Systems Support for Digital Audio and Video*, pages 54–59, New York, NY, USA, 2004. ACM Press.

[10] Michel Goemans and Jon Kleinberg. An improved approximation ratio for the minimum latency problem. In *SODA '96: Proceedings of the seventh annual ACM-SIAM Symposium on Discrete Algorithms*, pages 152–158, Philadelphia, PA, USA, 1996. Society for Industrial and Applied Mathematics.

[11] Michel X. Goemans. Minimum bounded degree spanning trees. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, pages 273–282, Los Alamitos, CA, USA, 2006. IEEE Computer Society.

[12] Michael T. Helmick and Fred S. Annexstein. Depth-latency tradeoffs in multicast tree algorithms. In *Proceedings of the IEEE 21st International Conference on Advanced Information Networking and Applications*, pages 555–564, Los Alamitos, CA, USA, 2007. IEEE Computer Society.

[13] J. Könemann, A. Levin, and A. Sinha. Approximating the degree-bounded minimum diameter spanning tree problem. *Algorithmica*, 41(2):117, 2004.

[14] Lap Chi Lau, Joseph (Seffi) Naor, Mohammad R. Salavatipour, and Mohit Singh. Survivable network design with degree or order constraints. In *STOC '07: Proceedings of the thirty-ninth annual ACM Symposium on Theory of Computing*, pages 651–660, New York, NY, USA, 2007. ACM Press.

[15] C. Papadimitriou and M. Yannakakis. The traveling salesman problem with distances one and two. *Mathematics of Operations Research*, 18:1–11, 1993.

[16] Mohit Singh and Lap Chi Lau. Approximating minimum bounded degree spanning trees to within one of optimal. In *STOC '07: Proceedings of the thirty-ninth annual ACM Symposium on Theory of Computing*, pages 661–670, New York, NY, USA, 2007. ACM Press.

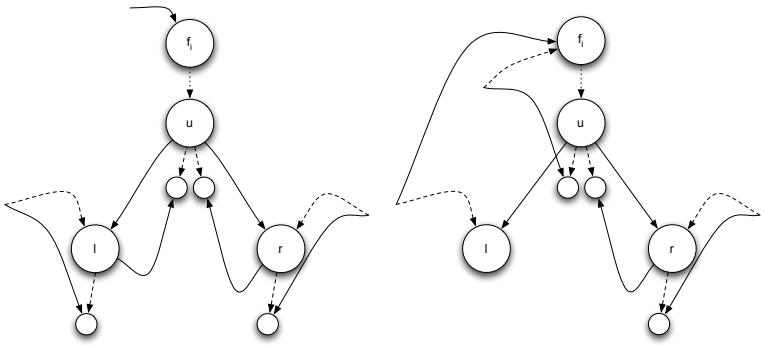

Figure 6: Fix-up operations. The left-hand figure shows the fix if f_i has lower latency than either l or r. The right-hand figure shows the fix if l has the lowest latency. The case for r having lowest latency is symmetric. Dotted lines show edges before the operation, solid lines show edges after the operation. Note that any grandchildren of u that are not effected are not shown.

5 Appendix

5.1 Proof of Lemma 2.1

LEMMA 5.1. *For any optimal latency degree-2 tree T, there is another degree-2 tree T' with the same latency, but with the property that each F_i is organized into a binary tree rooted at f_i with exactly the edge set specified by the construction of G(F) (and similarly for G_i, B_i and C_i). In particular, f_i and g_i only have one child each.*

Proof. Handle all nodes of F_i, G_i, B_i and C_i in top-down fashion. Let u be any node that has not yet been handled, but such that its parent *in G(F)* has been handled. Let l denote the left child of u *in G(F)*, and similarly r the right child. Let p be the parent of u *in T*, p_l similarly the parent of l and p_r the parent of r. Depending on whichever of f_i, r or l has the lowest latency in T, there are three different operations that can be used to make u the parent of l and r without incurring more latency (See Figure 6). In this picture, note that u will always have the same latency as f_i at the start of the fix-up (by induction). The fix-ups work because all points within a single F_i (or G_i, B_i or C_i) all have distance 0 to each other. The lemma follows by induction.

In essence, this lemma shows that there need not be any tricky back-and-forth movement from one F_i, G_i, B_i or C_i to the others.

CLAIM 1. (PART 1 OF LEMMA 2.1) *If T is a minimum latency degree-2 spanning tree of G(F), then T must include a binary tree on the nodes of "top" with the d_i being the leaves.*

Proof. It is enough to show that every d_i must have latency 1. Because the tree is degree 2, it is easy to see that the only way to get latency 1 for all d_i is to have the d_i at the leaves of a binary tree.

Let us consider the tree T' guaranteed by Lemma 5.1. If any d_i has latency ≥ 2, then one of the following must be true:

1. There is some internal node u of "top" that has latency > 0. This is not possible however. Just make u the left child of $root$, $root$'s left child becomes u's left child, and u's left child becomes the child of u's old parent. This saves at least one unit of latency, which is a contradiction.

2. There is some internal node u of "top" that has less than two children. In this case, just connect it to d_i, saving at least one unit of latency. This is a contradiction, however, because T was claimed to be optimal.

3. There is some internal node u of "top" that points directly to a f_i, g_i, or b_i. Note that by Lemma 5.1, if u points to some node of F_i, G_i or B_i other than f_i, g_i or b_i, then there is some other u' that is internal to "top" that points to f_i, g_i or b_i.

   If d_i (with the same value of i as the node pointed to by u) is the node from "top" with latency 2, then do the following: Make d_i a child of u in place of f_i (resp. g_i, b_i), make f_i the left child of d_i, and make d_i's old parent the new parent of d_i's old left child. This will decrease the total latency by at least one.

   On the other hand, consider the case where some f_i, g_i, or b_i is a child of u that is internal to "top," but such that d_i is also a child of an internal node of "top." In a previous case we proved that all internal nodes of "top" must have latency 0, so this means the maximum number of nodes not internal to "top" that can have parents internal to "top" is n. Hence, there must be some j such that none of f_j, g_j and b_j have latency ≤ 2. Assuming that Lemma 5.1 has been applied, we may switch u to point to d_j, and then d_j to b_j and f_j, and f_j to g_j. This saves latency at least $(33 + 20q) - 13q$.

   Either way, we reach a contradiction.

4. There is some internal node u of "top" that points directly to a clause. By an argument similar to the previous one, this is also impossible.

Hence, there is in fact no way that any d_i can have latency other than 1. Every case that arises when $d_i > 1$ leads to a contradiction of the optimality of T.

CLAIM 2. (PART 3 OF LEMMA 2.1) *In any optimal latency degree-2 tree, T, all nodes in a single F_i (or G_i or B_i) must have the same latency.*

Proof. The proof is a corollary of Lemma 5.1. If one follows the algorithm there, the result is that all nodes in F_i get latency equal to the minimum over all nodes of F_i before the operation, and no new latency is incurred anywhere. If the latency of any node were to decrease by this operation, that would contradict the optimality of T.

CLAIM 3. (PART 2 OF LEMMA 2.1) *Each d_i must take B_i as one child, and either F_i or G_i as the other. Whichever of F_i or G_i is not taken becomes the descendant of the other.*

Proof. Let us prove the second part first. By Lemma 5.1, f_i and g_i only need to have degree 1 in order to connect up F_i and G_i as required. There is always a spare edge available to allow f_i to be g_i's parent, or vice versa. If all points in F_i and B_i have latency 2, then G_i must descend from F_i: Any node not in F_i has distance at least 2 from G_i and latency at least 2, making the latency of points in G_i at least 4. Hence, making f_i point to g_i results in a guaranteed reduction of at least $16 + 7q$ latency. The same argument works if it is G_i and B_i whose points have latency 2.

By part 1 of the Lemma, we know that no F_i, G_i or B_i can be served directly from an internal node of "top," hence the only way to get latency 2 or 3 for any of them is to be served by d_i, and it is possible for every F_i, G_i and B_i to have latency either 2 or 3. Hence, serving any F_i, G_i or B_i by way of any other d_j, resulting in latency ≥ 4, is sub-optimal, which is again a contradiction.

Therefore, assume that all nodes in F_i, G_i and B_i have latency either 2 or 3. Some node in B_i must be the child of d_i. If not, all nodes of B_i have latency at least 4, for a total latency of $2(17 + 13q)$ greater than our proposed solution. On the other hand, the only way that this change could save latency is for d_i to take both F_i and G_i as children, saving at most $(16 + 7q) + 6q$. This additional $6q$ accounts for the fact that this tree can satisfy at most 6 more clauses than the proposed tree. Hence, any optimal tree must have some node in B_i as a child of its d_i.

CLAIM 4. (PART 4 OF LEMMA 2.1) *In any latency optimal degree-2 spanning tree of G(F), any satisfied clause must descend from either F_i or G_i, where x_i is one of the variables in the clause.*

Proof. If c_j descends from such an F_i or G_i, its latency is at most 4. By parts 1, 2 and 3 of the lemma, we know that c_j cannot be served by any node of "top." If c_j is served by any path not going through any of its variables, then the path must have length at least 5. Hence, by switching c_j to descend from some variable in its clause, we can save at least q latency, contradicting the optimality of T.