Improving Minimum Cost Spanning Trees by Upgrading Nodes[*]

S. O. Krumke[†] M. V. Marathe[‡] H. Noltemeier[†] R. Ravi[§] S. S. Ravi[¶]
R. Sundaram[||] H.-C. Wirth[†]

Abstract

We study *budget constrained network upgrading problems*. We are given an undirected edge weighted graph $G = (V, E)$ where node $v \in V$ can be upgraded at a cost of $c(v)$. This upgrade reduces the weight of each edge incident on v. The goal is to find a minimum cost set of nodes to be upgraded so that the resulting network has a minimum spanning tree of weight no more than a given budget D. The results obtained in the paper include the following:

1. On the positive side, we provide a polynomial time approximation algorithm for the above upgrading problem when the difference between the maximum and minimum edge weights is bounded by a polynomial in n, the number of nodes in the graph. The solution produced by the algorithm satisfies the budget constraint, and the cost of the upgrading set produced by the algorithm is $\mathcal{O}(\log n)$ times the minimum upgrading cost needed to obtain a spanning tree of weight at most D.
2. In contrast, we show that, unless $\mathsf{NP} \subseteq \mathsf{DTIME}(n^{\mathcal{O}(\log\log n)})$, there can be no polynomial time approximation algorithm for the problem that produces a solution with upgrading cost at most $\alpha < \ln n$ times the optimal upgrading cost even if the budget can be violated by a factor $f(n)$, for any polynomial time computable function $f(n)$. This result continues to hold, with $f(n) = n^k$ being any polynomial, even when the difference between the maximum and minimum edge weights is bounded by a polynomial in n.
3. Finally, we show that using a simple binary search over the set of admissible values, the dual problem can be solved with an appropriate performance guarantee.

AMS 1980 subject classification. 68R10, 68Q15, 68Q25

Keywords. Approximation algorithms, Bicriteria problems, Spanning trees, Network design, Combinatorial algorithms.

[*] A preliminary version of this paper appeared as [KM+97].

[†] Dept. of Computer Science, University of Würzburg, Am Hubland, 97074 Würzburg, Germany. Email: `{krumke,noltemei,wirth}@informatik.uni-wuerzburg.de`.

[‡] Los Alamos National Laboratory, P.O. Box 1663, MS B265, Los Alamos, NM 87545, USA. `marathe@lanl.gov`.

[§] R. Ravi, GSIA, Carnegie Mellon University, Pittsburgh, PA 15213, USA. Supported by NSF Career Grant CCR-9625297. Email: `ravi+@cmu.edu`.

[¶] S. S. Ravi, Dept. of Computer Science, University at Albany – SUNY, Albany, NY 12222, USA. Email: `ravi@cs.albany.edu`.

[||] R. Sundaram, Delta Global Trading LP, Boston, MA 02111, USA. Work done while at MIT, Cambridge MA 02139. Email: `koods@delta-global.com`.

1 Introduction and Problem Formulation

1.1 Motivation

Several problems arising in areas such as communication networks and VLSI design can be expressed in the following general form: Enhance the performance of a given network by upgrading a suitable subset of nodes. In communication networks, upgrading a node corresponds to installing faster communication equipment at that node. Such an upgrade reduces the communication delay along each edge emanating from the node. In signal flow networks used in VLSI design, upgrading a node corresponds to replacing a circuit module at the node by a functionally equivalent module containing suitable drivers. Such an upgrade decreases the signal transmission delay along the wires connected to the module [PS95]. Usually, there is a cost associated with upgrading a node, and this motivates the study of problems of the following type: find an upgrading set of minimum cost so that the resulting network satisfies certain performance requirements.

The performance of the upgraded network can be quantified in a number of ways. In this paper, we consider the weight of a minimum spanning tree in the upgraded network as the performance measure. We show that this network problem is NP-hard. So, the focus of the paper is on the design of efficient approximation algorithms.

1.2 Preliminary Definitions

1.2.1 Node upgrade model

The *node based upgrading model* discussed in this paper can be formally described as follows. Let $G = (V, E)$ be a connected undirected graph. For each edge $e \in E$, we are given three integers $d_0(e) \geq d_1(e) \geq d_2(e) \geq 0$. The value $d_i(e)$ represents the *length* or *delay* of the edge e if exactly i of its endpoints are upgraded.

Thus, the upgrade of a node v reduces the delay of each edge incident on v. The (integral) value $c(v)$ specifies how expensive it is to upgrade the node v. The cost of upgrading all vertices in $W \subseteq V$, denoted by $c(W)$, is equal to $\sum_{v \in W} c(v)$.

Given a set $W \subseteq V$ of vertices, denote by d_W the edge weight function resulting from the upgrade of the vertices in W; that is, for an edge $(u, v) \in E$

$$d_W(u, v) := d_i(u, v) \qquad \text{where } i = |W \cap \{u, v\}|.$$

Our model is a generalization of the node upgrade model introduced by Paik and Sahni in [PS95]. In their model, the reduction in edge weight resulting from an upgrade of nodes is determined by a constant $0 < \alpha < 1$ in the following way: if exactly one endpoint of an edge is upgraded, then its weight is reduced by the factor α; if both endpoints are upgraded, the weight is reduced by the factor α^2. Clearly, the Paik-Sahni model is a special case of the node upgrade model used in this paper.

1.2.2 Background: Bicriteria Problems and Approximation

The problem considered in this paper involves two optimization objectives, namely, the upgrading cost and the weight of a minimum spanning tree in the upgraded network.

A framework for such bicriteria problems has been developed in [MR$^+$95]. Since this framework is used throughout this paper, we briefly review the relevant definitions from [MR$^+$95].

A generic bicriteria problem can be specified as a triple (f, g, Γ) where f and g are two objectives and Γ specifies a class of subgraphs. An instance of a bicriteria problem specifies a budget on the objective g. A subgraph in the class Γ is a *valid* solution if it satisfies this budget constraint. The goal is to find a valid solution that minimizes the objective f.

Using this notation, the problem treated in this paper can be expressed as (NODE UPGRADING COST, TOTAL WEIGHT, SPANNING TREE). The interpretation of this notation is that the budgeted objective is the weight of a minimum spanning tree in the upgraded network, and the goal is to minimize the upgrade cost.

Definition 1 (Approximation Algorithm) *A (polynomial time) algorithm for a bicriteria problem* (f, g, Γ) *is said to have* performance (α, β), *if it has the following property: For any instance of* (f, g, Γ), *the algorithm*

1. *either produces a solution from the subgraph class* Γ *for which the value of objective* g *is at most* β *times the specified budget and the value of objective* f *is at most* α *times the minimum value of a solution from* Γ *that satisfies the budget constraint, or*
2. *correctly provides the information that there is no subgraph from* Γ *which satisfies the budget constraint on* g.

1.3 Problem Definition

We denote the total length of a minimum spanning tree (MST) in G with respect to the weight function d_W by $\text{MST}(G, d_W)$.

Definition 2 (Upgrading MST Problem) *Given an edge and node weighted graph* $G = (V, E)$ *as above and a bound* D, *the* upgrading minimum spanning tree problem, *denoted by* (NODE UPGRADING COST, TOTAL WEIGHT, SPANNING TREE), *is to upgrade a set* $W \subseteq V$ *of nodes such that* $\text{MST}(G, d_W) \leq D$ *and* $c(W)$ *is minimized.*

The problem (NODE UPGRADING COST, TOTAL WEIGHT, SPANNING TREE) is formulated by specifying a budget on the weight of a tree while the upgrading cost is to be minimized. We will refer to this problem as the *primal problem*. It is also meaningful to consider the corresponding *dual problem*, denoted by (TOTAL WEIGHT, NODE UPGRADING COST, SPANNING TREE), where we are given a budget on the upgrading cost and the goal is to minimize the weight of a spanning tree in the resulting graph.

Definition 3 (Dual Upgrading MST Problem) *Given an edge and node weighted graph* $G = (V, E)$ *as above and a bound* B *on the upgrading cost, the problem* (TOTAL WEIGHT, NODE UPGRADING COST, SPANNING TREE), *is to upgrade a set* $W \subseteq V$ *of nodes such that* $c(W) \leq B$ *and* $\text{MST}(G, d_W)$ *is minimized.*

There is a close relationship between the approximabilities of the primal and the dual problems. We will show in Section 3 that a good bicriteria approximation algorithm for one of the problems can be used to design a good approximation algorithm for the other one in a generic way; that is, given an (α, β)-approximation algorithm for the problem (NODE UPGRADING COST, TOTAL WEIGHT, SPANNING TREE), one can obtain a (β, α)-approximation algorithm for the problem (TOTAL WEIGHT, NODE UPGRADING COST, SPANNING TREE).

2 Summary of Results and Related Work

2.1 Summary of Results

We derive our approximation results under the following assumption:

Assumption 4 *There is a polynomial* p *such that* $D_0 - D_2 \leq p(n)$, *where* $D_0 := \max_{e \in E} d_0(e)$ *and* $D_2 := \min_{e \in E} d_2(e)$ *are the maximum and minimum edge weight respectively, and* n *denotes the number of nodes in the graph.*

The main results of this paper are as follows.

1. We present a polynomial time approximation algorithm, which for any fixed $\varepsilon > 0$, provides a performance guarantee of $((1+\varepsilon)^2 \mathcal{O}(\log n), 1)$ for any instance of (NODE UPGRADING COST, TOTAL WEIGHT, SPANNING TREE) satisfying Assumption 4.

2. In contrast, we show that Unless $\mathsf{NP} \subseteq \mathsf{DTIME}(n^{\mathcal{O}(\log \log n)})$, there can be no polynomial time approximation algorithm for (NODE UPGRADING COST, TOTAL WEIGHT, SPANNING TREE) with a performance of $(\alpha, f(n))$ for any $\alpha < \ln n$ and any polynomial time computable function f. This result continues to hold, with $f(n) = n^k$ being any polynomial, even when Assumption 4 holds.

3. We also show that using a simple binary search over the set of admissible values, an approximation algorithm with a performance guarantee of $(1, (1+\varepsilon)^2 \mathcal{O}(\log n))$ can be obtained for any instance of the dual problem (TOTAL WEIGHT, NODE UPGRADING COST, SPANNING TREE) satisfying Assumption 4.

It should be noted that our approximation algorithm for the problem (NODE UPGRADING COST, TOTAL WEIGHT, SPANNING TREE) produces solutions in which the budget constraint is strictly satisfied. This is unlike many bicriteria network design problems where it is necessary to violate the budget constraint to obtain a solution that is near-optimal with respect to the objective function [MR+95].

2.2 Related Work

As mentioned earlier, a simpler node upgrading model has been considered by Paik and Sahni [PS95]. Under their model, Paik and Sahni studied the upgrading problem for several performance measures including the maximum delay on an edge and the diameter of the network. They presented NP-hardness results for several problems. Their focus was on the development of polynomial time algorithms for special classes of networks (e.g. trees, series-parallel graphs) rather than on the development of approximation algorithms. Our constructions can be modified to show that all the problems considered here remain NP-hard even under the Paik-Sahni model.

While in this paper we choose the total weight of a minimum spanning tree as a measure of the performance of the upgraded network, there are other useful performance measures. One of these measures, namely the bottleneck weight of a minimum bottleneck spanning tree, leads to the problem (NODE UPGRADING COST, BOTTLENECK WEIGHT, SPANNING TREE). This bottleneck problem has been investigated in [KM+97].

Edge-based network upgrading problems have also been considered in the literature [Ber92, KN+96b, KN+96a]. There, each edge has a current weight and a minimum weight (below which the edge weight cannot be decreased). Upgrading an edge corresponds to decreasing the weight of that particular edge, and there is a cost associated with such an upgrade. The goal is to obtain an upgraded network with the best performance. In [KN+96b] the authors consider the problem of edge-based upgrading to obtain the best possible MST subject to a budget constraint on the upgrading cost and present a $(1+\varepsilon, 1+1/\varepsilon)$-approximation algorithm. Generalized versions where there are other constraints (e.g. bound on maximum node degree) and the goal is to obtain a good Steiner tree, are considered in [KN+96a]. Other references addressing problems that can be interpreted as edge-based improvement problems include [FSO96, HT97, Phi93].

3 Dual Problems and Approximability

In this section we formally state and prove our claim from Section 1.3 that the dual problems defined in this paper are closely related with respect to their approximability.

Lemma 5 *If there exists an approximation algorithm for the problem* (NODE UPGRADING COST, TOTAL WEIGHT, SPANNING TREE) *with a performance of* (α, β)*, then there is an approximation algorithm for the problem* (TOTAL WEIGHT, NODE UPGRADING COST, SPANNING TREE) *with performance of* (β, α).

Proof. Let A be an (α, β)-approximation algorithm for (NODE UPGRADING COST, TOTAL WEIGHT, SPANNING TREE). We will show how to use A to construct a (β, α)-approximation algorithm for the dual problem.

An instance of (TOTAL WEIGHT, NODE UPGRADING COST, SPANNING TREE) is specified by a graph $G = (V, E)$, the node cost function c, the weight functions d_i, $i = 0, 1, 2$, on the edges and the bound B on the node upgrading cost. We denote by OPT the optimum weight of an MST after upgrading a vertex set of cost at most B. Observe that OPT is an integer such that $(n-1)D_2 \leq \text{OPT} \leq (n-1)D_0$ where $D_2 := \min_{e\in E} d_2(e)$ and $D_0 := \max_{e\in E} d_0(e)$.

We use binary search to find the minimum integer D such that $(n-1)D_2 \leq D \leq (n-1)D_0$ and algorithm A applied to the instance of (NODE UPGRADING COST, TOTAL WEIGHT, SPANNING TREE) given by the weighted graph G as above and the bound D on the weight of an MST after the upgrade, outputs an upgrading set of cost at most αB. It is easy to see that this binary search indeed works and terminates with a value $D \leq \text{OPT}$. The corresponding upgrading set W then satisfies $\text{MST}(G, d_W) \leq \beta D \leq \beta\text{OPT}$ and $c(W) \leq \alpha B$. □

Using a similar technique, one can also establish the following result:

Lemma 6 *If there exists an approximation algorithm for the problem* (TOTAL WEIGHT, NODE UPGRADING COST, SPANNING TREE) *with a performance of* (α, β)*, then there is an approximation algorithm for the problem* (NODE UPGRADING COST, TOTAL WEIGHT, SPANNING TREE) *with performance of* (β, α). □

In view of Lemma 5, the next section focuses on the development of an approximation algorithm for the problem (NODE UPGRADING COST, TOTAL WEIGHT, SPANNING TREE).

4 The Algorithm

In this section we develop our approximation algorithm for the (NODE UPGRADING COST, TOTAL WEIGHT, SPANNING TREE) problem. Without loss of generality, we assume that for a given instance of (NODE UPGRADING COST, TOTAL WEIGHT, SPANNING TREE), the bound D on the weight of the minimum spanning tree after the upgrade satisfies $D \geq \mathrm{MST}(G, d_2)$, since no upgrade strategy can shorten an edge $e \in E$ below $d_2(e)$, and therefore it is impossible to obtain a minimum spanning tree of weight strictly lower than $\mathrm{MST}(G, d_2)$ in our upgrading model. Thus, we can assume that there always exists a subset of the nodes which, when upgraded, leads to an MST of weight at most D. We remind the reader that our algorithm also uses Assumption 4 (stated in Section 2) regarding the edge weights in the given instance.

4.1 Overview of the Algorithm

Our approximation algorithm can be thought of as a *local improvement* type algorithm. To begin with, we compute an MST in the given graph with edge weights given by $d_0(e)$. This value equals $d_W(e)$ for the initial case $W = \varnothing$, where $W \subseteq V$ is the set of upgraded nodes maintained by the algorithm. During each iteration, we select a node and a subset of its neighbors and upgrade them by adding them to the set W. The policy used in the selection process is that of finding a set which gives us the best ratio improvement, which is defined as the ratio of the improvement in the total weight of the spanning tree to the total cost spent for upgrading the chosen nodes. Having selected such a set, we recompute the MST and repeat our procedure. The procedure is halted when the weight of the MST is at most the required bound D. To find a subset of nodes with the best ratio improvement in each iteration, we use an approximate solution to the *Two Cost Spanning Tree Problem* defined below.

Definition 7 (*Two Cost Spanning Tree Problem*) *Given a connected undirected graph $G = (V, E)$, two edge weight functions, c and l, and a bound B, find a spanning tree T of G such that the total cost $c(T)$ is at most B and the total cost $l(T)$ is a minimum among all spanning trees that obey the budget constraint.*

In the framework of bicriteria problems, the above problem can be expressed as (l-TOTAL WEIGHT, c-TOTAL WEIGHT, SPANNING TREE). This problem has been addressed by Ravi and Goemans [RG96] who obtained the following result.

Theorem 8 *For all $\varepsilon > 0$, there is a polynomial time approximation algorithm for the Two Cost Spanning Tree problem with a performance of $(1, 1 + \varepsilon)$.* □

4.2 Algorithm and Performance Guarantee

The remainder of Section 4 is devoted to a proof of the following theorem.

Theorem 9 *For any fixed $\varepsilon > 0$, there is a polynomial time approximation algorithm that provides a performance guarantee of $((1 + \varepsilon)^2 \mathcal{O}(\log n), 1)$ for any instance of* (NODE UPGRADING COST, TOTAL WEIGHT, SPANNING TREE) *satisfying Assumption 4.*

```
ALGORITHM UPGRADE MST($\Omega$)
• Input: A graph $G = (V, E)$, three edge weight functions $d_0 \geq d_1 \geq d_2$, a node weight
function $c$, and a number $D$, which is a bound on the weight of an MST in the upgraded
graph; a "guess value" $\Omega$ for the optimal upgrading cost.

  1. Initialize the set of upgraded nodes: $W_0 := \emptyset$.
  2. Let $T_0 := \mathrm{MST}(G, d_{W_0})$.
  3. Initialize the iteration count: $i := 1$.
  4. Repeat the following steps until the current tree $T_{i-1}$ and the weight function $d_{W_{i-1}}$
     satisfy the condition $d_{W_{i-1}}(T_{i-1}) \leq D$:
     (a) Let $T_{i-1} := \mathrm{MST}(G, d_{W_{i-1}})$ be an MST w.r.t. the weight function $d_{W_{i-1}}$.
     (b) Call Procedure COMPUTE QC to find a marked claw $C$ with "good" quotient cost
         $q(C)$. Procedure COMPUTE QC is called with the graph $G$, the current MST $T_{i-1}$,
         the current weight function $d_{W_{i-1}}$ and the bound $\Omega$.
     (c) If Procedure COMPUTE QC reports failure, then report failure and stop.
     (d) Upgrade the marked vertices $M(C)$ in $C$: $W_i := W_{i-1} \cup M(C)$.
     (e) Increment the iteration count: $i := i + 1$.

• Output: A spanning tree with weight at most $D$, such that total cost of upgrading the
nodes is no more than $(1 + \varepsilon)\Omega \cdot \mathcal{O}(\log n)$, provided $\Omega \geq$ OPT. Here, OPT denotes the
optimal upgrading cost to reduce the weight of an MST to be at most $D$.
```

Figure 1: Approximation algorithm for node upgrading under total weight constraint.

The algorithm referred to Theorem 9 is obtained by executing ALGORITHM UPGRADE MST (shown in Figure 1) for a polynomial number of values of the parameter Ω. (Details regarding the values of Ω used by the algorithm appear in Section 4.5.) ALGORITHM UPGRADE MST uses Procedure COMPUTE QC whose description appears in Figure 2.

Before we embark on a proof of the performance guarantee stated in Theorem 9, we give the overall idea behind the proof. Recall that each basic step of the algorithm consists of finding a node and a subset of neighbors to upgrade.

Definition 10 (Claw) *A graph* $C = (V, E)$ *is called a* claw, *if* E *is of the form* $E = \{(v, w) : w \in V \setminus \{v\}\}$ *for some node* $v \in V$. *The node* v *is said to be the* center *of the claw. A claw with at least two nodes is called a* nontrivial claw.

Notice that a claw's center is not uniquely determined if the claw contains less than three nodes.

Let W be a subset of the nodes upgraded so far and let T be an MST with respect to d_W; that is, $T = \mathrm{MST}(G, d_W)$. For a claw C with nodes $M(C) \subseteq C$ marked, we define its *quotient cost* $q(C)$ to be

$$q(C) := \begin{cases} \frac{c(M(C))}{d_W(T) - \mathrm{MST}(T \cup C, d_{W \cup M(C)})}, & \text{if } M(C) \neq \emptyset, \\ +\infty, & \text{otherwise.} \end{cases}$$

In other words, $q(C)$ is the cost of the vertices in $M(C)$ divided by the decrease in the weight of the MST when the vertices in $M(C)$ are also upgraded and edges in the current tree T can be exchanged for edges in the claw C. Notice that this way the real profit of upgrading the vertices $M(C)$ is underestimated, since the weight of edges outside of C might also decrease.

Our analysis shows that in each iteration, there exists a claw of quotient cost at most $\frac{2\,\mathrm{OPT}}{d_W(T)-D}$, where T is an MST at the beginning of the iteration and W are the nodes upgraded so far. Essentially this means, that in each iteration there is a claw whose quotient cost is bounded by the ratio of twice the optimum cost and the remaining effort. We can then use a potential function argument to show that this yields a logarithmic performance guarantee.

4.3 Bounded Claw Decompositions

Definition 11 *Let $G = (V, E)$ be a graph and $W \subseteq V$ a subset of marked vertices. Let $\kappa \geq 1$ be an integer constant. A* κ-bounded claw decomposition of G with respect to W *is a collection $C_1, \dots, C_r$ of nontrivial claws, which are all subgraphs of G, with the following properties:*

1. *$\bigcup_{i=1}^{r} V(C_i) = V$ and $\bigcup_{i=1}^{r} E(C_i) = E$.*
2. *No node from W appears in more than κ claws.*
3. *The claws are edge-disjoint.*
4. *If a claw C_i contains nodes from W, then its center also belongs to W.*

Lemma 12 *Let F be a forest in $G = (V, E)$ and let $W \subseteq V$ be a set of marked nodes. Then there is a 2-bounded claw decomposition of F with respect to W.*

Proof. We show how to decompose each tree T in the forest F to get a 2-bounded decomposition.

If each node in T has degree one, then T consists of a single edge which is a nontrivial claw. Otherwise, let v be an arbitrary vertex of degree at least two where at least one of its neighbors is of degree one. If all neighbors of v are of degree one, then T is again already a nontrivial claw and we are done.

Let N_v be the neighbors of v in T which are of degree one. Construct a claw C_v by selecting as its vertex set $N_v \cup \{v\}$. Remove the vertices in N_v from T. Call the resulting tree T'. Observe that T' consists of at least two vertices.

Repeat the above procedure with T' until we end up with a single claw. Add this claw to the collection of claws. Since each vertex appears in at least one and in at most two of the claws C_v, the claim now follows. □

In the sequel we will make heavy use of the following simple lemma.

Lemma 13 *Let T_H be a minimum spanning tree of a graph H with edge weights given by d. For any additional edge e to be added to H, if $T' = \mathrm{MST}(T_H \cup \{e\})$ is a minimum spanning tree in $T_H \cup \{e\}$, then T' is also a minimum spanning tree in $H \cup \{e\}$.* □

Lemma 14 *Let H be a spanning subgraph of a graph $G = (V, E)$ with edge weights given by d. Let $S \subseteq E$ be a set of edges, and let $f \notin H \cup S$ be an additional edge. Then,*

$$\mathrm{MST}(H \cup S) - \mathrm{MST}(H \cup S \cup f) \leq \mathrm{MST}(H) - \mathrm{MST}(H \cup f), \tag{1}$$

that is, the net profit from using an edge f in terms of decreasing the weight of an MST can never be increased with the help of a set S of edges added together with f.

PROCEDURE COMPUTE QC(Ω)

• *Input:* A graph $G = (V, E)$, a spanning tree T and a weight function d on E; $W \subseteq V$ is the set of upgraded nodes; a "guess" Ω for the optimal upgrading cost.

1. Let $s := \lceil \log_{1+\varepsilon} \Omega \rceil$.
2. For each node $v \notin W$ and all $K \in \{1, (1+\varepsilon), (1+\varepsilon)^2, \dots, (1+\varepsilon)^s\}$ do
   (a) Set up an instance $I_{v,K}$ of the *Two Cost Spanning Tree Problem* as follows:
      - The vertex set of the graph G_v contains all the vertices in G and an additional "dummy node" x.
      - There is an edge (v, x) joining v to the dummy node x of length $l(v, x) = 0$ and cost $c(v, x) = c(v)$ thus modeling the upgrading cost of v.
      - For each edge $(v, w) \in E$, G_v contains two parallel edges h and h_{up}. The edge h models the situation where w is not upgraded, while h_{up} models an upgrade of w:

$$\begin{array}{llll} c(h) := 0 & & c(h_{\text{up}}) := 0, & \text{if } w \in W \\ l(h) := d_2(v,w), & \text{if } w \in W & c(h_{\text{up}}) := c(w), & \text{if } w \notin W \\ l(h) := d_1(v,w), & \text{if } w \notin W & l(h_{\text{up}}) := d_2(v,w). & \end{array}$$

      - For each edge $(u, w) \in T$, there is one edge $(u, w) \in E$ which has length $l(u, w) = d(u, w)$ and cost $c(u, w) = 0$.
      - The bound B on the c-cost of the tree is set to K.

   (b) Using the algorithm mentioned in Theorem 8, find a tree of c-cost at most $(1+\varepsilon)K$ and l-cost no more than that of a minimum budget K bounded spanning tree (if one exists). Let $T_{v,K}$ be the tree produced by the algorithm.
3. If the algorithm fails for *all* instances $I_{v,K}$ then report failure and stop.
4. Among all the trees $T_{v,K}$ find a tree T_{v^*,K^*} which minimizes the ratio $c(T_{v^*,K^*})/(d(T) - l(T_{v^*,K^*}))$.
5. Construct a marked claw C from T_{v^*,K^*} as follows:
   - The center of C is v^* and v^* is marked.
   - The edge (v^*, w) is in the claw C if T_{v^*,K^*} contains an edge between v^* and w. The node w is marked if and only if the edge in T_{v^*,K^*} between v^* and w has c-cost greater than zero.

• *Output:* A marked claw C (with its center also marked) with quotient cost $q(C)$ satisfying $q(C) \le 2(1+\varepsilon)^2 \frac{\mathrm{OPT}}{d(T)-D}$ and cost $c(M(C)) \le (1+\varepsilon)\Omega$.

Figure 2: Algorithm for computing a good claw.

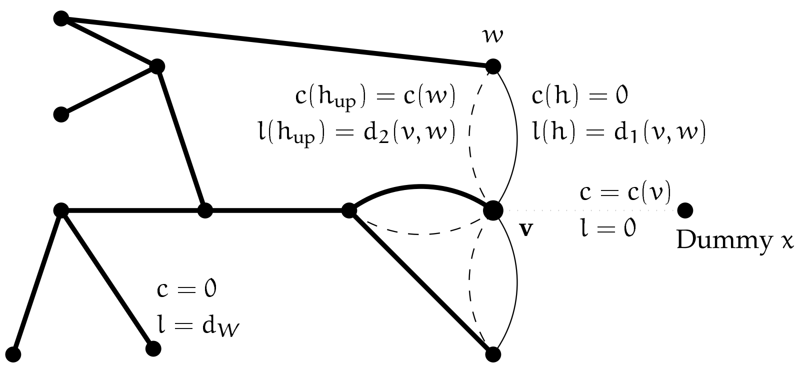

Figure 3: Example of an instance $I_{v,K}$ of *Two Cost Spanning Tree Problem* constructed by Procedure COMPUTE QC. Edges in the current MST are indicated by thick lines. The vertex w is not contained in the current upgrading set.

Proof. We show the claim by induction on $|S|$. The claim is trivial if S is empty. Now let $|S| = k + 1$, $S = S' \cup \{e\}$, and let the claim hold for all sets of cardinality k.

By induction hypothesis, we have for the set S' that

$$\mathrm{MST}(H \cup S') - \mathrm{MST}(H \cup S' \cup f) \leq \mathrm{MST}(H) - \mathrm{MST}(H \cup f). \tag{2}$$

Thus it suffices to show that $\mathrm{MST}(H \cup S) - \mathrm{MST}(H \cup S \cup f)$ is bounded by the left hand side of (2).

Let $T_{S'} := \mathrm{MST}(H \cup S')$ and $T_S := \mathrm{MST}(H \cup S)$. Notice that by Lemma 13 we can also write T_S as $T_S = \mathrm{MST}(T_{S'} \cup e)$. Hence T_S differs from $T_{S'}$ by at most one edge swap; that is, we can obtain T_S from $T_{S'}$ by inserting edge e and removing a heaviest edge e' on the cycle C'_e induced by e in $T_{S'}$. (Here, we explicitly allow the possibility that $e = e'$, in which case $T_S = T_{S'}$).

We now estimate the weight difference between $T_{S'}$ and $\mathrm{MST}(H \cup S' \cup f) = \mathrm{MST}(T_{S'} \cup f)$. Let C'_f be the cycle induced by the additional edge f in $T_{S'}$. If e' is not on C'_f, then C'_f is also exactly that cycle which edge f induces in T_S. Thus, in this case we get that

$$\mathrm{MST}(H \cup S) - \mathrm{MST}(H \cup S \cup f) = \mathrm{MST}(H \cup S') - \mathrm{MST}(H \cup S' \cup f) \tag{3}$$

since the weights of both $T_{S'}$ and T_S decrease by the difference of the weight of the heaviest edge on C_f and $d(f)$.

In the other case, the edge e' swapped out of $T_{S'}$ in exchange for e is on C'_f. Clearly, e' must be a heaviest edge on the cycle C'_e that is induced by e in $T_{S'}$. Thus, all edges on C'_e have weight at most that of e'.

The cycle which f induces in T_S contains only edges from $C'_e \cup C'_f$. Since all the edges on C'_e have weight at most $d(e')$, this implies that the edge f can only replace edges in T_S of weight at most that of a heaviest edge in C'_e. But this means that

$$d(T_S) - \mathrm{MST}(T_S \cup f) \leq \max_{h \in C'_e} d(h) - d(f) = d(T'_S) - \mathrm{MST}(T_S \cup f).$$

In other words, we have that

$$\mathrm{MST}(H \cup S) - \mathrm{MST}(H \cup S \cup f) \leq \mathrm{MST}(T \cup S') - \mathrm{MST}(T \cup S' \cup f). \tag{4}$$

Now using (3) or (4), respectively, together with the induction hypothesis stated in (2) yields the lemma. □

Lemma 15 *Let* T *be a spanning tree of* G *and let* T' *be an MST with respect to some weight function* d *on the edges. Let* $\Delta := \mathsf{T}' \setminus \mathsf{T}$ *be the edge difference set of the trees. Then there is a nonnegative function* $e \mapsto u(e)$, $e \in \Delta$, *with the following properties:*

1. $\sum_{e\in\Delta} u(e) = d(T) - d(T')$.
2. *For any subset* $A \subseteq \Delta$,
$$d(T) - \mathrm{MST}(T \cup A) \geq \sum_{e\in A} u(e).$$

Proof. Let $\Delta = \{e_1, \dots, e_k\}$, and define function u by

$$u(e_j) := \mathrm{MST}(T \cup \{e_1, \dots, e_{j-1}\}) - \mathrm{MST}(T \cup \{e_1, \dots, e_j\}), \quad j = 1, \dots, k.$$

The first claim of the lemma is now obvious. Thus, we need to consider only the second claim.

Let $A = \{e_{i_1}, \dots, e_{i_r}\} \subseteq \Delta$, with $i_1 \leq \cdots \leq i_r$. Write $\Delta_j := \{e_i \in \Delta \mid i \leq i_j\}$ and $A_j := \Delta_j \cap A$. Notice that $A_0 = \varnothing$ and $A_r = A$. With these definitions, we have

$$u(e_{i_j}) = \mathrm{MST}(T \cup (\Delta_j \setminus e_{i_j})) - \mathrm{MST}(T \cup \Delta_j).$$

Furthermore,

$$\begin{aligned}
&d(T) - \mathrm{MST}(T \cup A_r)\\
&\quad= \sum_{j=1}^{r} \Big(\mathrm{MST}(T \cup A_{j-1}) - \mathrm{MST}(T \cup A_j)\Big)\\
&\quad= \sum_{j=1}^{r} \Big(\mathrm{MST}(T \cup A_{j-1}) - \mathrm{MST}(T \cup A_{j-1} \cup e_{i_j})\Big)
\end{aligned}$$

and by Lemma 14 applied to $H := T \cup A_{j-1}$, $S := (\Delta_j \setminus e_{i_j})$ and $f := e_{i_j}$

$$\begin{aligned}
&\quad\geq \sum_{j=1}^{r} \Big(\mathrm{MST}\big(T \cup A_{j-1} \cup (\Delta_j \setminus e_{i_j})\big) - \mathrm{MST}\big(T \cup A_{j-1} \cup (\Delta_j \setminus e_{i_j}) \cup e_{i_j}\big)\Big)\\
&\quad= \sum_{j=1}^{r} \Big(\mathrm{MST}\big(T \cup (\Delta_j \setminus e_{i_j})\big) - \mathrm{MST}(T \cup \Delta_j)\Big)\\
&\quad= \sum_{j=1}^{r} u(e_{i_j}),
\end{aligned}$$

which is what we wanted to show. □

Lemma 16 *Let* $T := T_{i-1}$ *be an MST at the beginning of iteration* i, *i.e.,* $T = \mathrm{MST}(G, d_W)$, *where* $W := W_{i-1}$ *is the upgrading set constructed so far. Then there is a marked claw* C *(where its center* v *is also marked and* $v \notin W$*) with quotient cost* $q(C)$ *satisfying*

$$q(C) \leq \frac{2\,\mathrm{OPT}}{d_W(T) - D} \qquad \textit{and} \qquad c(M(C)) \leq \mathrm{OPT}.$$

Proof. Let $T' = \mathrm{MST}(G, d_{W\cup \mathrm{OPT}})$ be an MST after the additional upgrade of the vertices in OPT. Clearly, $d_{W\cup \mathrm{OPT}}(T') \le D$. Apply Lemma 12 to T' with the vertices in $Z := \mathrm{OPT} \setminus W$ marked. The lemma shows that there is a 2-bounded claw decomposition of T' with respect to Z. Let the claws be $C_1, \ldots, C_r$. In each claw C_j, the corresponding nodes $M(C_j) := C_j \cap Z$ from Z are marked. Since the decomposition is 2-bounded with respect to Z, it follows that

$$\sum_{j=1}^{r} c(M(C_j)) \le 2 \cdot \mathrm{OPT}. \tag{5}$$

Moreover, the cost $c(M(C_j))$ of the marked nodes in each single claw C_j does not exceed OPT, since we have marked only nodes from Z.

Let H be the graph obtained from G by inserting each edge e twice, on the one hand with weight $d_W(e)$ and on the other hand with the new weight $d_{W\cup \mathrm{OPT}}(e)$. Then, T is a spanning tree of H and T' can be identified with a minimum spanning tree of H.

Let $\Delta := T' \setminus T$ be the edge difference set of trees T' and T in H, where we consider two parallel edges with different weights as different edges. By Lemma 15, there exist numbers $u(e)$, $e \in \Delta$, such that

$$\sum_{e\in\Delta} u(e) = d_W(T) - d_{W\cup \mathrm{OPT}}(T') \ge d_W(T) - D. \tag{6}$$

Moreover, for any edge subset of Δ, and in particular, for any claw C_j, we have

$$d_W(T) - \mathrm{MST}(T \cup C_j, d_{W\cup \mathrm{OPT}}) \ge \sum_{e\in C_j} u(e). \tag{7}$$

In fact, Lemma 15 provides the stronger result that the difference of $d_W(T)$ and the minimum spanning tree in $T \cup C_j$ where each edge e of T still considered to have the old weight $d_W(e)$ is at least the right hand side of (7).

Now, since the weight of edges in claw C_j is only affected by the upgrade of the vertices in $M(C_j) = Z \cap C_j$, we obtain from (7)

$$d_W(T) - \mathrm{MST}(T \cup C_j, d_{W\cup M(C_j)}) \ge \sum_{e\in C_j} u(e), \tag{8}$$

that is, the decrease in the weight of the MST obtained by upgrading the vertices in $M(C_j)$ and possibly exchanging edges from the current MST for edges from C_j is at least the sum $\sum_{e\in C_j} u(e)$.

Therefore, for each of the claws C_j with $M(C_j) \ne \emptyset$ in the 2-bounded decomposition of T', the quotient cost $q(C_j)$ satisfies

$$q(C_j) \le \frac{c(M(C_j))}{\sum_{e\in C_j} u(e)}. \tag{9}$$

Let C be a claw among all the claws C_j with minimum $q(C)$. Then,

$$q(C) \cdot \sum_{e\in C_j} u(e) \le c(M(C_j)) \quad \text{for } j = 1, \ldots, r. \tag{10}$$

Notice that the above inequality holds, regardless of whether $M(C_j)$ is empty or not. Summing the inequalities in (10) over $j = 1, \ldots, r$, and using Equations (5), (6), and (8), it can be seen that C is a claw with the desired properties. □

4.4 Finding a good claw in each iteration

Lemma 16 implies the existence of a marked claw with the required properties. We will now deal with the problem of finding such a claw.

Lemma 17 *Suppose that the bound Ω given to Algorithm* UPGRADE MST *satisfies* $\Omega \geq \text{OPT}$. *Then, during each iteration* i, *the algorithm chooses a marked claw* C' *such that*

$$q(C') \leq 2(1+\varepsilon)^2 \frac{\text{OPT}}{d_W(T) - D} \qquad \textit{and} \qquad c(M(C')) \leq (1+\varepsilon)\Omega,$$

where $T := T_{i-1}$ *is an MST at the beginning of iteration* i *and* $W := W_{i-1}$ *is the set of nodes upgraded so far.*

Proof. By Lemma 16, there is a marked claw C with quotient cost $q(C)$ at most $2\frac{\text{OPT}}{d_W(T)-D}$. Let v be the center of this claw. By Lemma 16, v is marked. Let $c(C) := c(M(C))$ be the cost of the marked nodes in C and $L := \text{MST}(T \cup C, d_{W \cup M(C)})$ be the weight of the MST in $T \cup C$ resulting from the upgrade of the marked vertices in C. Then, by the definition of the quotient cost $q(C)$ we have

$$q(C) = \frac{c(C)}{d_W(T) - L} \leq 2\frac{\text{OPT}}{d_W(T) - D}. \tag{11}$$

Consider the iteration of Procedure COMPUTE QC when it processes the instance $I_{v,K}$ of *Two Cost Spanning Tree Problem* with graph G_v and $c(C) \leq K < (1+\varepsilon) \cdot c(C)$. The tree $\text{MST}(T \cup C, d_{W \cup M(C)})$ induces a spanning tree in G_v of total c-cost at most $c(C)$ (which is at most K) and of total l-length no more than L. Thus, the algorithm from Theorem 8 will find a tree $T_{v,K}$ such that its total c-cost $c(T_{v,K})$ is bounded from above by $(1+\varepsilon)K \leq (1+\varepsilon)^2 c(C)$ and of total l-length $l(T_{v,K})$ no more than L.

By construction, the marked claw C' computed by PROCEDURE COMPUTE QC from $T_{v,K}$ has quotient cost at most $c(T_{v,K})/(d_W(T) - l(T_{v,K}))$, which is at most $(1+\varepsilon)^2 c(C)/(d_W(T) - L)$. The lemma now follows from (11). □

4.5 Guessing an Upper Bound on the Improvement Cost

We run our Algorithm UPGRADE MST depicted in Figure 1 for all values of

$$\Omega \in \{1, (1+\varepsilon), (1+\varepsilon)^2, \dots, (1+\varepsilon)^t\}, \qquad \text{where } t := \lceil \log_{1+\varepsilon} c(V) \rceil.$$

We then choose the best solution among all the solutions produced. Our analysis shows that when $\text{OPT} \leq \Omega < (1+\varepsilon) \cdot \text{OPT}$, the algorithm will indeed produce a solution. In the sequel, we estimate the quality of this solution. Assume that the algorithm uses $f+1$ iterations and denote by $C_1, \dots, C_f, C_{f+1}$ the claws chosen in Step 4b of the algorithm. Let $c_i := c(M(C_i))$ denote the cost of the vertices upgraded in iteration i. Then, by construction

$$c_i \leq (1+\varepsilon)\Omega \leq (1+\varepsilon)^2 \text{OPT} \quad \text{for } i = 1, \dots, f+1. \tag{12}$$

4.6 Potential Function Argument

We are now ready to complete the proof of the performance stated in Theorem 9. Let MST_i denote the weight of the MST at the end of iteration i, i.e., $\mathrm{MST}_i := d_{W_i}(T_i)$. Define $\phi_i := \mathrm{MST}_i - D$. Since we have assumed that the algorithm uses $f+1$ iterations, we have $\phi_i \geq 1$ for $i = 0, \ldots, f$ and $\phi_{f+1} \leq 0$. As before, let $c_i := c(M(C_i))$ denote the cost of the vertices upgraded in iteration i. Then

$$\phi_{i+1} = \phi_i - (\mathrm{MST}_i - \mathrm{MST}_{i+1}) \overset{\text{Lemma 17}}{\leq} \left(1 - \frac{c_{i+1}}{\alpha \cdot \mathrm{OPT}}\right) \phi_i, \tag{13}$$

where $\alpha := 2(1+\varepsilon)^2$. We now use an analysis technique due to Leighton and Rao [LR88]. The recurrence (13) and the estimate $\ln(1-\tau) \leq -\tau$ give us

$$\sum_{i=1}^{f} c_i \leq \alpha \cdot \mathrm{OPT} \cdot \ln \frac{\phi_0}{\phi_f}. \tag{14}$$

Notice that the total cost of the nodes chosen by the algorithm is exactly the sum $\sum_{i=1}^{f+1} c_i$. By (14) and (12) we have

$$\sum_{i=1}^{f+1} c_i = c_{f+1} + \sum_{i=1}^{f} c_i \leq (1+\varepsilon)^2 \mathrm{OPT} + 2(1+\varepsilon)^2 \mathrm{OPT} \cdot \ln \frac{\phi_0}{\phi_f}. \tag{15}$$

We will now show how to bound $\ln \frac{\phi_0}{\phi_f}$. Notice that $\phi_f = \mathrm{MST}_f - D \geq 1$, since the algorithm uses $f+1$ iterations and does not stop after the fth iteration. We have $\phi_0 = \mathrm{MST}_0 - D \leq (n-1)(D_0 - D_2)$, where D_0 and D_2 denote the maximum and the minimum edge weight in the graph. It now follows from Assumption 4 that $\ln \phi_0 \in \mathcal{O}(\log(np(n))) \subseteq \mathcal{O}(\log n)$. Using this result in (15) yields

$$\begin{aligned}\sum_{i=1}^{f+1} c_i &\leq (1+\varepsilon)^2 \cdot \mathrm{OPT} + 2(1+\varepsilon)^2 \mathcal{O}(\log n) \cdot \mathrm{OPT} \\ &\in (1+\varepsilon)^2 \mathcal{O}(\log n) \cdot \mathrm{OPT}.\end{aligned}$$

This completes the proof. □

5 Hardness Result

In this section we prove the hardness result stated in Section 2. The proof relies on the following lemma.

Lemma 18 *Let α and f be two polynomial time computable functions. Let α be nondecreasing, and let there be constants $c > 1$ and $N \in \mathbb{N}$ such that $\alpha(n+1) \leq c \cdot \alpha(n)$ for all $n \geq N$. Then the existence of an $(\alpha(n), f(n))$-approximation algorithm for* (NODE UPGRADING COST, TOTAL WEIGHT, SPANNING TREE) *implies the existence of a $c \cdot \alpha(n)$-approximation algorithm for* MINIMUM DOMINATING SET. *Here, n denotes the number of vertices in the input graphs.*

Note that requiring the existence of c is not a serious restriction, since we can always assume $\alpha(n) \leq n$.

Proof. We perform a reduction from MINIMUM DOMINATING SET [GJ79, Problem GT2]. An instance of MINIMUM DOMINATING SET is given by a undirected graph $G = (V, E)$. A node set $D \subseteq V$ is a minimum dominating set, if each node in $V \setminus D$ is incident to a node in D, and D is of minimum cardinality among all node sets with the domination property.

Given an instance $G = (V, E)$ of MINIMUM DOMINATING SET, add a new node r (the root) to the graph and connect r to all the nodes in V. Let $n' = |V|+1$ be the resulting number of nodes. For all edges, set the initial weights to $l_0 := n \cdot f(n') + 1$, and the weights in the upgrading case to $l_1 := l_2 := 1$. The upgrade cost of the root is set to $c(r) := \lceil n \cdot c \cdot \alpha(n) \rceil + 1$, all remaining nodes have upgrading cost 1. The constraint on the total weight is $n := |V|$.

Now suppose there is an $(\alpha(n'), f(n'))$-approximation algorithm for (NODE UPGRADING COST, TOTAL WEIGHT, SPANNING TREE). Observe that for the instance of this problem constructed above, there is always a feasible solution, namely, the upgrading set consisting of all vertices in the graph. Thus, if applied to this instance, the algorithm must output an upgrading set of cost at most $\alpha(n')$ times the optimum upgrading cost such that the upgraded network contains an MST of weight at most $f(n') \cdot n$.

It is easy to see that upgrading a dominating set of size u in G yields a minimum spanning tree in G' which fulfills the weight constraint and has upgrade costs equal to u. Thus the optimum upgrading cost OPT is at most the size of the minimum dominating set.

Conversely, each upgrading set in G' not containing the root and resulting in an MST of weight at most n is also a dominating set in G. Now observe that any spanning tree of weight more than n has weight at least $l_0 = n \cdot f(n') + 1 > n \cdot f(n')$. Thus, to satisfy the weight constraint within a factor of $f(n')$, the algorithm must output a spanning tree consisting of edges of weight 1 only. Moreover, due to the high cost of upgrading the root, the algorithm can never choose the root for upgrading: let u be the size of a smallest dominating set, then $\text{OPT} \leq u$ by our observations from above. The algorithm produces a solution of cost at most $\alpha(n') \cdot \text{OPT} \leq \alpha(n') \cdot u \leq c \cdot \alpha(n) \cdot u < c(r)$.

Thus, an $(\alpha(n'), f(n'))$-approximation algorithm can be used to obtain a dominating set in the original graph G whose size is at most $c \cdot \alpha(n)$ times the cardinality of an optimum dominating set. □

Corollary 19 (Non-Approximability) *Let f be any (polynomial time computable) function, and $\alpha(n) < (1 - \varepsilon) \ln n$ for fixed $\varepsilon > 0$. Unless $\mathsf{NP} \subseteq \mathsf{DTIME}(n^{O(\log\log n)})$, there can be no polynomial time approximation algorithm for* (NODE UPGRADING COST, TOTAL WEIGHT, SPANNING TREE) *with performance guarantee $(\alpha(n), f(n))$, where n denotes the number of vertices in the input graph.*

Proof. Feige [Fei96] has shown that, unless $\mathsf{NP} \subseteq \mathsf{DTIME}(n^{O(\log\log n)})$, there can be no $\alpha(n)$-approximation algorithm for MINIMUM DOMINATING SET when $\alpha(n) < \ln n$.

For some $\varepsilon > 0$, assume that there is a $((1 - \varepsilon) \ln n, f(n))$-approximation algorithm for (NODE UPGRADING COST, TOTAL WEIGHT, SPANNING TREE). Then there are constants ε', c, and N, such that for $n \geq N$

$$\ln(n + 1) \leq c \cdot \ln n \qquad \text{and} \qquad (1 - \varepsilon) \ln n \leq \frac{1 - \varepsilon'}{c} \ln n.$$

With help of Lemma 18 we can conclude that there exists a $((1 - \varepsilon') \ln n)$-approximation algorithm for MINIMUM DOMINATING SET, which contradicts Feige's result. □

References

[Ber92] O. Berman, *Improving the location of minisum facilities through network modification*, Annals of Operations Research **40** (1992), 1–16.

[Fei96] U. Feige, *A threshold of* $\ln n$ *for approximating set cover*, Proceedings of the 28th Annual ACM Symposium on the Theory of Computing (STOC'96), 1996, pp. 314–318.

[FSO96] G. N. Frederickson and R. Solis-Oba, *Increasing the weight of minimum spanning tree*, Proceedings of the 7th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'96), March 1996.

[GJ79] M. R. Garey and D. S. Johnson, *Computers and intractability (a guide to the theory of* NP-*completeness)*, W.H. Freeman and Company, New York, 1979.

[HT97] S. E. Hambrusch and H. Y. Tu, *Edge weight reduction problems in acyclic graphs*, J. Algorithms **24** (1997), 66–93.

[KM$^+$97] S. O. Krumke, M. V. Marathe, H. Noltemeier, R. Ravi, S. S. Ravi, R. Sundaram, and H. C. Wirth, *Improving spanning trees by upgrading nodes*, Proceedings of the 24nd International Colloquium on Automata, Languages and Programming (ICALP'97), Lecture Notes in Computer Science, vol. 1256, 1997, pp. 281–291.

[KN$^+$96a] S. O. Krumke, H. Noltemeier, M. V. Marathe, R. Ravi, and S. S. Ravi, *Improving Steiner trees of a network under multiple constraints*, Tech. Report LA-UR 96-1374, Los Alamos National Laboratory, Los Alamos, New Mexico, USA, 1996.

[KN$^+$96b] S. O. Krumke, H. Noltemeier, S. S. Ravi, M. V. Marathe, and K. U. Drangmeister, *Modifying networks to obtain low cost trees*, Proceedings of the 22nd International Workshop on Graph-Theoretic Concepts in Computer Science, Cadenabbia, Italy., Lecture Notes in Computer Science., vol. 1197, June 1996, pp. 293–307. A Complete version to appear in Theoretical Computer Science, 1998.

[LR88] F. T. Leighton and S. Rao, *An approximate max-flow min-cut theorem for uniform multicommodity flow problems with application to approximation algorithms*, Proceedings of the 29th Annual IEEE Symposium on the Foundations of Computer Science (FOCS'88), 1988, pp. 422–431.

[MR$^+$95] M. V. Marathe, R. Ravi, R. Sundaram, S. S. Ravi, D. J. Rosenkrantz, and H. B. Hunt III, *Bicriteria network design problems*, Proceedings of the 22nd International Colloquium on Automata, Languages and Programming (ICALP'95), Lecture Notes in Computer Science,, vol. 944, 1995, pp. 487–498. A complete version to appear in J. Algorithms, 1998.

[Phi93] C. Phillips, *The network inhibition problem*, Proceedings of the 25th Annual ACM Symposium on the Theory of Computing (STOC'93), May 1993, pp. 288–293.

[PS95] D. Paik and S. Sahni, *Network upgrading problems*, Networks **26** (1995), 45–58.

[RG96] R. Ravi and M. X. Goemans, *The constrained minimum spanning tree problem*, Proceedings Scandinavian Workshop on Algorithmic Theory (SWAT'96), Reykjavik, Iceland, Lecture Notes in Computer Science, vol. 1097, July 1996, pp. 66–75.