Improving Spanning Trees by Upgrading Nodes

S. O. Krumke,[1] M. V. Marathe,[2] H. Noltemeier,[1]
R. Ravi,[3] S. S. Ravi,[4] R. Sundaram,[5] H. C. Wirth[1]

[1] Dept. of Computer Science, University of Würzburg, Am Hubland, 97074 Würzburg, Germany.
Email: {krumke,noltemei,wirth}@informatik.uni-wuerzburg.de
[2] Los Alamos National Laboratory, P.O. Box 1663, MS K990, Los Alamos, NM 87545, USA. Email: madhav@c3.lanl.gov[§]
[3] GSIA, Carnegie Mellon University, Pittsburgh, PA 15213. Email: ravi+@cmu.edu[¶]
[4] Dept. of Computer Science, University at Albany – SUNY, Albany, NY 12222, USA. Email: ravi@cs.albany.edu
[5] Delta Trading Co. Work done while at MIT, Cambridge MA 02139. Email: koods@theory.lcs.mit.edu.[‖]

Abstract. We study *budget constrained optimal network upgrading problems*. We are given an edge weighted graph $G = (V, E)$ where node $v \in V$ can be upgraded at a cost of $c(v)$. This upgrade reduces the delay of each link emanating from v. The goal is to find a minimum cost set of nodes to be upgraded so that the resulting network has a good performance. We consider two performance measures, namely, the weight of a minimum spanning tree and the bottleneck weight of a minimum bottleneck spanning tree, and present approximation algorithms.

1 Introduction, Motivation and Summary of Results

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. There is a cost associated with upgrading a node, and there is often a budget on the total upgrading cost. Therefore, it is of interest to study the problem of upgrading a network so that the total upgrading cost obeys the budget constraint and the resulting network has the best possible performance among all upgrades that satisfy the budget constraint.

[§] Supported by the Department of Energy under Contract W-7405-ENG-36.
[¶] Supported by NSF CAREER grant CCR-9625297.
[‖] Supported by DARPA contract N0014-92-J-1799 and NSF CCR 92-12184.

The performance of the upgraded network can be quantified in a number of ways. In this paper, we consider two such measures, namely, the weight of a minimum spanning tree in the upgraded network and the bottleneck cost (i.e., the maximum weight of an edge) in a spanning tree of the upgraded network. Under either measure, the upgrading problem can be shown to be NP-hard. So, the focus of the paper is on the design of efficient approximation algorithms.

1.1 Background: Bicriteria Problems and Approximation

The problems considered in this paper involve two optimization objectives, namely, the upgrading cost and the performance of the upgraded network. A framework for such bicriteria problems has been developed in [7]. A generic bicriteria problem can be specified as a triple (A, B, Γ) where A and B are two objectives and Γ specifies a class of subgraphs. An instance specifies a budget on the objective A and the goal is to find a subgraph in the class Γ that minimizes the objective B for the upgraded network. As an example, the problem of upgrading a network so that the modified network has a spanning tree of weight at most D while minimizing the node upgrading cost can be expressed as (TOTAL WEIGHT, NODE UPGRADING COST, SPANNING TREE).

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

1. either produces a solution from the subgraph class Γ for which the value of objective A is at most α times the specified budget and the value of objective B is at most β 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 A.

1.2 Problem Definitions

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 with 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 $\sum_{v \in W} c(v)$.

For 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 $(u, v) \in E$

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

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

We also consider the node based upgrading problem to obtain a spanning tree with the bottleneck cost at most a given value. We denote the bottleneck weight (i.e., the maximum weight of an edge) of a minimum bottleneck spanning tree of G with respect to the weight function d_W by $\mathrm{MBOT}(G, d_W)$.

Definition 3. Given an edge and node weighted graph $G = (V, E)$ as above and a bound D, the *upgrading minimum bottleneck spanning tree problem*, denoted by (BOTTLENECK WEIGHT, NODE UPGRADING COST, SPANNING TREE), is to upgrade a set $W \subseteq V$ of nodes such that $\mathrm{MBOT}(G, d_W) \leq D$ and $c(W)$ is minimized.

Dual Problems The problem (TOTAL WEIGHT, NODE UPGRADING COST, SPANNING TREE) is formulated by specifying a budget on the weight of a tree while the upgrading cost is to be minimized. It is also meaningful to consider the corresponding *dual problem*, denoted by (NODE UPGRADING COST, TOTAL WEIGHT, 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.

Lemma 4. *If there exists an approximation algorithm for* (TOTAL WEIGHT, NODE UPGRADING COST, SPANNING TREE) *with a performance of* (α, β)*, then there is an approximation algorithm for* (NODE UPGRADING COST, TOTAL WEIGHT, SPANNING TREE) *with performance of* (β, α).

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

An instance of (NODE UPGRADING COST, TOTAL WEIGHT, 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 $(n-1)D_2 \leq \mathrm{OPT} \leq (n-1)D_0$ 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, BOTTLENECK 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 $D \leq \mathrm{OPT}$. The corresponding upgrading set W then satisfies $\mathrm{MST}(G, d_W) \leq \beta D \leq \beta \mathrm{OPT}$ and $c(W) \leq \alpha B$. □

A result similar to Lemma 4 can be shown for the bottleneck case. In view of these results, we express our results for the problems (TOTAL WEIGHT, NODE UPGRADING COST, SPANNING TREE) and (BOTTLENECK WEIGHT, NODE UPGRADING COST, SPANNING TREE).

1.3 Summary of Results

For the total weight MST upgrading problem, we derive our approximation results under the following assumption:

Assumption 5. *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.*

Theorem 6. *For any fixed $\varepsilon > 0$, there is a polynomial time algorithm which, for any instance of* (Total Weight, Node Upgrading Cost, Spanning Tree) *satisfying Assumption 5, provides a performance of* $(1,\ (1+\varepsilon)^2\mathcal{O}(\log n))$.

For the bottleneck case, we do not need any assumption about the edge weights.

Theorem 7. *There is an approximation algorithm for the* (Bottleneck Weight, Node Upgrading Cost, Spanning Tree) *problem with performance* $(1,\ 2\ln n)$.

Our approximation results are complemented by the following hardness results:

Theorem 8. *Unless* $\mathsf{NP} \subseteq \mathsf{DTIME}(n^{\mathcal{O}(\log\log n)})$, *there can be no polynomial time approximation algorithm for either* (Total Weight, Node Upgrading Cost, Spanning Tree) *or* (Bottleneck Weight, Node Upgrading Cost, Spanning Tree) *with a performance of* $(f(n),\ \alpha)$ *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 if Assumption 5 holds.*

Due to space limitations, the remainder of this paper discusses mainly the algorithm mentioned in Theorem 6 above. Proofs of other results will appear in a complete version of this paper.

1.4 Related Work

Some node upgrading problems have been investigated under a simpler model by Paik and Sahni [9]. In their model, the delay of an edge is decreased by constant factors of δ or δ^2, when one or two of its endpoints are upgraded, respectively. Clearly, this model is a special case of the model treated in our paper.

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.

Edge-based network upgrading problems have also been considered in the literature [1, 4, 5]. 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 [4] 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 [5]. Other references that address problems that can be interpreted as edge-based improvement problems include [3, 8, 10].

2 Upgrading Under Total Weight Constraint

In this section we develop our approximation algorithm for the (TOTAL WEIGHT, NODE UPGRADING COST, SPANNING TREE) problem. Without loss of generality we assume that for a given instance of (TOTAL WEIGHT, NODE UPGRADING COST, SPANNING TREE) the bound D on the weight of the minimum spanning tree after the upgrade satisfies $D \geq \mathrm{MST}(G, d_2)$, i.e., the weight of an MST with respect to d_2, since node upgrading cannot reduce the weight of the minimum spanning tree below this value. Thus, 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 5 (stated in Section 1.3) regarding the edge weights in the given instance.

2.1 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 $d_0(e)$. Now, during each iteration, we select a node and a subset of its neighbors and upgrade them. 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 on upgrading the 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 threshold D. To find a subset of node with the best ratio improvement in each iteration, we use an approximate solution to the *Two Cost Spanning Tree Problem* defined below.

Definition 9 *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.

The above problem can be expressed as the bicriteria problem (c-TOTAL WEIGHT, l-TOTAL WEIGHT, SPANNING TREE). This problem has been addressed by Ravi and Goemans [11] who obtained the following result.

Theorem 10. *For all $\varepsilon > 0$, there is a polynomial time approximation algorithm for the Two Cost Spanning Tree problem with a performance of $(1+\varepsilon, 1)$.*

2.2 Algorithm and Performance Guarantee

The steps of our algorithm are shown in Figure 1. This algorithm uses Procedure COMPUTE QC whose description appears in Figure 2.

ALGORITHM UPGRADE MST(Ω)

• *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" Ω 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 for the current tree T_{i-1} and the weight function $d_{W_{i-1}}$ we have: $d_{W_{i-1}}(T_{i-1}) \leq D$:
   (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 Ω.
   (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.

Fig. 1. Approximation algorithm for node upgrading under total weight constraint.

Before we embark on a proof of Theorem 6, 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 11. 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*.

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

$$q(C) := \frac{c(M(C))}{d_W(T) - \mathrm{MST}(T \cup C, d_{W \cup M(C)})} \quad , \text{if } M(C) \neq \emptyset,$$

and $+\infty$ otherwise. 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 essentially 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 the weight of an MST at the beginning of the iteration and W are the nodes upgraded so far. We can then use a potential function argument to show that this yields a logarithmic performance guarantee.

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 $K \in \{1, (1+\varepsilon), (1+\varepsilon)^2, \ldots, (1+\varepsilon)^s\}$ 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:

     $$c(h) := 0 \qquad l(h) := \begin{cases} d_2(v,w) & \text{if } w \in W \\ d_1(v,w) & \text{if } w \notin W \end{cases}$$

     Similarly, h_{up} models an upgrade of w:

     $$c(h_{\mathrm{up}}) := \begin{cases} 0 & \text{if } w \in W \\ c(w) & \text{if } w \notin W \end{cases} \qquad l(h_{\mathrm{up}}) := d_2(v, w).$$

   - 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 10, 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 $q(C) \leq 2(1+\varepsilon)^2 \frac{\mathrm{OPT}}{d(T)-D}$ and cost $c(M(C)) \leq (1+\varepsilon)\Omega$.

Fig. 2. Algorithm for computing a good claw.

2.3 Bounded Claw Decompositions

Definition 12. 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 $C_1, \ldots, C_r$ 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 belongs also to W.

Lemma 13. *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.* □

Lemma 14. *Let $T := T_{i-1}$ be an MST at the beginning of iteration i with $W := W_{i-1}$ being the nodes upgraded so far. Let $U \subseteq V$ be a set of nodes. Let $T' = \mathrm{MST}(G, d_{W \cup U})$ be a minimum spanning tree after the additional upgrade of the vertices in U. Then, there is a bijection $\varphi : T \to T'$ with the following properties: 1. For all edges $e \in T \cap T'$ we have $\varphi(e) = e$, 2. $d_{W \cup U}(\varphi(e)) \leq d_W(e)$ for all $e \in T$, 3. the "swaps" $e \mapsto \varphi(e)$ transform T into T', and 4. $\sum_{e \in T}(d_W(e) - d_{W \cup U}(\varphi(e))) = d_W(T) - d_{W \cup U}(T')$.* □

Lemma 15. *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*

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

Proof. Let $T' = \mathrm{MST}(G, d_{W \cup \mathrm{OPT}})$ be an MST after the additional upgrade of the vertices in OPT. Clearly, $d_{W \cup \mathrm{OPT}}(T') \leq D$. Apply Lemma 13 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

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

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. By Lemma 14, there exists a bijection $\varphi\colon T \to T'$ such that

$$\sum_{e \in T} \Big(d_W(e) - d_{W \cup \mathrm{OPT}}(\varphi(e)) \Big) = d_W(T) - d_{W \cup \mathrm{OPT}}(T') \geq d_W(T) - D. \tag{2}$$

For each of the claws C_j with $M(C_j) \neq \emptyset$ in the 2-bounded decomposition of T' its quotient cost $q(C_j)$ satisfies

$$q(C_j) \leq \frac{c(M(C_j))}{\sum_{\varphi(e)\in C_j}(d_W(e) - d_{W\cup \mathrm{OPT}}(\varphi(e)))}, \tag{3}$$

since we can exchange the edges $\varphi(e)$ ($e \in C_j$) for the corresponding edges e in the current tree T after the upgrade and thus decrease the weight of the tree by at least $\sum_{\varphi(e)\in C_j}(d_W(e) - d_{W\cup \mathrm{OPT}}(\varphi(e)))$.

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

$$q(C) \cdot \sum_{e\in C_j} \Big(d_W(e) - d_{W\cup \mathrm{OPT}}(\varphi(e))\Big) \leq c(M(C_j)) \quad \text{for } j = 1, \ldots, r. \tag{4}$$

Notice that the above equation holds, regardless of whether $M(C_j)$ is empty or not. Summing the inequalities in (4) over $j = 1, \ldots, r$, and using Equations (1) and (2), it can be seen that C is a claw with the desired properties. □

2.4 Finding a good claw in each iteration

Lemma 15 implies the existence of a marked claw with the required properties. We will now deal with the problem of finding such a claw.

Lemma 16. *Suppose that the bound Ω given to Algorithm* Upgrade MST *satisfies $\Omega \geq \mathrm{OPT}$. Then, for each stage i of the algorithm, it chooses a marked claw C' such that*

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

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 15, there is a marked claw C with quotient cost $q(C)$ at most $2\frac{\mathrm{OPT}}{d_W(T)-D}$. Let v be the center of this claw. By Lemma 15, v is marked. Let $c(C) := c(M(C))$ be the cost of the marked nodes in C and $L := \mathrm{MST}(T \cup C, d_{W\cup M(C)})$ be the weight of the MST in $T \cup C$ resulting from the upgrade of the marked vertices in C. Then, by definition of the quotient cost $q(C)$ we have

$$q(C) = \frac{c(C)}{d_W(T) - L} \leq 2\frac{\mathrm{OPT}}{d_W(T) - D}. \tag{5}$$

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 $\mathrm{MST}(T \cup C, d_{W\cup M(C)})$ 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 10 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 (5). □

2.5 Guessing an Upper Bound on the Improvement Cost

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

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

We then choose the best solution among all solutions produced. Our analysis shows that when $\mathrm{OPT} \leq \Omega < (1+\varepsilon) \cdot \mathrm{OPT}$, 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 $C_1, \ldots, C_f, C_{f+1}$ 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

$$c_i \leq (1+\varepsilon)\Omega \leq (1+\varepsilon)^2 \mathrm{OPT} \quad \text{for } i = 1, \ldots, f+1. \tag{6}$$

2.6 Potential Function Argument

We are now ready to complete the proof of the performance stated in Theorem 6. 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

$$\phi_{i+1} = \phi_i - (\mathrm{MST}_i - \mathrm{MST}_{i+1}) \overset{\text{Lemma 16}}{\leq} \left(1 - \frac{c_{i+1}}{\alpha \cdot \mathrm{OPT}}\right) \phi_i, \tag{7}$$

where $\alpha := 2(1+\varepsilon)^2$. We now use an analysis technique due to Leighton and Rao [6]. The recurrence (7) and the estimate $\ln(1-\tau) \leq -\tau$ give us

$$\sum_{i=1}^{f} c_i \leq \alpha \cdot \mathrm{OPT} \cdot \ln \frac{\phi_0}{\phi_f}. \tag{8}$$

Notice that the total cost of the nodes chosen by the algorithm is exactly the sum $\sum_{i=1}^{f+1} c_i$. By (8) and (6) we have

$$\sum_{i=1}^{f+1} c_i = c_{f+1} + \sum_{i=1}^{f} c_i \leq (1+\varepsilon)^2 \mathrm{OPT} + 2(1+\varepsilon)^2 \mathrm{OPT} \cdot \ln \frac{\phi_0}{\phi_f}. \tag{9}$$

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 5 that $\ln \phi_0 \in \mathcal{O}(\log(np(n))) \subseteq \mathcal{O}(\log n)$. Using this result in (9) yields

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

3 Concluding Remarks

Our algorithms produced solutions in which the budget constraints were 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 [7].

An open problem that arises immediately from our work is whether there is a good approximation algorithm for the (TOTAL WEIGHT, NODE UPGRADING COST, SPANNING TREE) problem even when Assumption 5 is not satisfied. It is also of interest to investigate whether our results for spanning trees can be extended to Steiner trees. Other open problems under the node-based upgrading model can be formulated using different performance measures for the upgraded network. Some measures which are of interest in this context include bottleneck weight, diameter and lengths of paths between specified pairs of vertices.

References

1. O. Berman, "Improving The Location of Minisum Facilities Through Network Modification," *Annals of Operations Research*, Vol. 40, 1992, pp. 1–16.
2. U. Feige, "A threshold of $\ln n$ for approximating set cover," *Proc. 28th Annual ACM Symposium on the Theory of Computing*, Philadelphia, PA, May 1996, pp. 314–318.
3. G. N. Frederickson and R. Solis-Oba, "Increasing the Weight of Minimum Spanning Trees", *Proc. 6th Annual ACM-SIAM Symposium on Discrete Algorithms*, January 1996, pp. 539–546.
4. S. O. Krumke, H. Noltemeier, M. V. Marathe, S. S. Ravi and K. U. Drangmeister, "Modifying Networks to Obtain Low Cost Trees," *Proc. Workshop on Graph Theoretic Concepts in Computer Science*, Cadenabbia, Italy, June 1996, pp. 293–307.
5. S. O. Krumke, H. Noltemeier, M. V. Marathe, R. Ravi and S. S. Ravi, "Improving Steiner Trees of a Network Under Multiple Constraints", Technical Report, LA-UR 96-1374, Los Alamos National Laboratory, Los Alamos, NM, 1996.
6. F. T. Leighton and S. Rao, "An Approximate Max-Flow Min-Cut Theorem for Uniform Multicommodity Flow Problems with Application to Approximation Algorithms", *Proc. 29th Annual IEEE Conference on Foundations of Computer Science*, Oct. 1988, pp. 422–431.
7. M. V. Marathe, R. Ravi, R. Sundaram, S. S. Ravi, D. J. Rosenkrantz and H. B. Hunt III, "Bicriteria Network Design Problems", In *Proc. 22nd International Colloquium on Automata, Languages and Programming*, July 1995, Vol. 944 of *Lecture Notes in Computer Science*, pp. 487–498.
8. J. Plesnik, "The Complexity of Designing a Network with Minimum Diameter", *Networks*, Vol. 11, 1981, pp. 77–85.
9. D. Paik and S. Sahni, "Network Upgrading Problems," *Networks*, Vol. 26, 1995, pp. 45–58.
10. C. Phillips, "The Network Inhibition Problem," *Proc. 25th Annual ACM Symposium on Theory of Computing*, San Diego,CA, May 1993, pp. 288–293.
11. R. Ravi and M. X. Goemans, "The Constrained Minimum Spanning Tree Problem", *Proc. Scandinavian Workshop on Algorithmic Theory*, Reykjavik, July 1996.

This article was processed using the LaTeX macro package with LLNCS style