LA-UR 97-476

CONF-970746--1

Los Alamos National Laboratory is operated by the University of California for the United States Department of Energy under contract W-7405-ENG-36

TITLE: **IMPROVING SPANNING TREES BY UPGRADING NODES**

AUTHOR(S): S. O. Krumke, M. V. Marathe, H. Noltemeier, R. Ravi, S. S. Ravi, R. Sundaram, H. C. Wirth

SUBMITTED TO: International Colloquium on Automata, Language and Programming
Bologna, Italy
July, 1997

Los Alamos

Los Alamos National Laboratory
Los Alamos New Mexico 87545

Improving Spanning Trees by Upgrading Nodes

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

January 16, 1997

Abstract

We study *budget constrained optimal network upgrading problems.* Such problems aim at finding optimal strategies for improving a network under some cost measure subject to certain budget constraints. A general problem in this setting is the following. We are given an edge weighted graph $G = (V, E)$ where nodes represent processors and edges represent bidirectional communication links. The processor at a node $v \in V$ can be upgraded at a cost of $c(v)$. Such an 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 the best performance with respect to some measure. We consider the problem under two measures, namely, the weight of a minimum spanning tree and the bottleneck weight of a minimum bottleneck spanning tree. We present approximation and hardness results for the problem. Our results are tight to within constant factors. We also show that these approximation algorithms can be used to construct good approximation algorithms for the dual versions of the problems where there is a budget constraint on the upgrading cost and the objectives are minimum weight spanning tree and minimum bottleneck weight spanning tree respectively.

[1] Department 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. The work is supported by the Department of Energy under Contract W-7405-ENG-36.

[3] Department of Computer Science, University at Albany – SUNY, Albany, NY 12222, USA. Email: ravi@cs.albany.edu.

[4] GSIA, Carnegie Mellon University, Pittsburgh, PA 15213. Email: ravi+@cmu.edu.

[5] Delta Trading Co. Work done while at MIT, Cambridge MA 02139. Email: koods@theory.lcs.mit.edu. Research supported by DARPA contract N0014-92-J-1799 and NSF CCR 92-12184.

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.

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 development of efficient approximation algorithms for upgrading problems under these two measures.

1.1 Background: Bicriteria Problems and Approximation

The problems considered in this paper involve two optimization measures, namely, the upgrading cost and the performance of the upgraded network. Such problems can be formulated using a framework for bicriteria problems developed in [8]. Using this framework, a generic bicriteria upgrading problem can be specified as a triple (**A**, **B**, **S**) where **A** and **B** are two minimization objectives and **S** specifies a class of subgraphs. The problem specifies a budget on the objective **A** and the goal is to find a subgraph in the class **S** 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 least possible weight subject to a budget constraint on the upgrading cost can be expressed as (NODE UPGRADING COST, TOTAL WEIGHT, SPANNING TREE). Similarly, the budget constrained upgrading problem where the goal is to minimize the bottleneck weight of a spanning tree can be expressed as (NODE UPGRADING COST, BOTTLENECK WEIGHT, SPANNING TREE).

We also use the notion of performance guarantee for bicriteria problems given in [8]. An (α, β)-approximation algorithm for a bicriteria problem (**A**, **B**, **S**) is a polynomial-time algorithm that produces a solution in 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 for any solution that satisfies the budget constraint. Furthermore, the solution produced by the approximation algorithm must belong to the subgraph class **S**. Reference [8] includes a discussion of known approximation results on bicriteria network design problems.

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 $e = (u, v) \in E$, we are given three nonnegative integers denoted by $d(e)$, $d_m(e)$ and $d_l(e)$. Here, $d(e)$ represents the *length* or *delay*

of the edge e. If exactly one of the endpoints u and v is upgraded, the delay of e decreases to $d_m(e)$, the "medium" delay. If both endpoints are upgraded, then the delay falls to $d_l(e)$, the "low" delay. It is assumed that $d_l(e) \leq d_m(e) \leq d(e)$.

Thus, the upgrade of a node v reduces the delay of each edge incident with v. For each node $v \in V$ the (integral) value $c(v)$ specifies how expensive it is to upgrade the node. For a subset W of V, the cost of upgrading all the nodes in W, 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 $e = (u, v) \in E$

$$d_W(u,v) := \begin{cases} d(u,v) & \text{if } |W \cap \{u,v\}| = 0, \\ d_m(u,v) & \text{if } |W \cap \{u,v\}| = 1, \\ d_l(u,v) & \text{if } |W \cap \{u,v\}| = 2. \end{cases}$$

We denote the total length of a minimum spanning tree (MST) with respect to the weight function d_W by $\mathrm{MST}_G(d_W)$. We are now ready to formulate the problems studied in this paper.

Definition 1.1 *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.*

The above problem is formulated by specifying a bound on the weight of an MST after the upgrade, 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 want to obtain the best possible weight of an MST while staying within the budget restrictions. It can be shown that an (α, β)-approximation algorithm for (TOTALWEIGHT, NODEUPGRADINGCOST, SPANNINGTREE) can be used as a subroutine to obtain a (β, α)-approximation algorithm for the dual problem.

We also consider the node based upgrading problem to obtain a spanning tree with the least possible bottleneck cost. In defining this problem, 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 $\mathrm{MBOT}_G(d_W)$. A formal definition of this problem is as follows.

Definition 1.2 *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.*

Again, it can be shown that an (α, β)-approximation algorithm for the above problem can be converted into a (β, α)-approximation algorithm for the corresponding dual problem, where we are given a bound on the node upgrading cost and want to minimize the bottleneck weight of the tree.

In view of the fact that approximation results carry over to the respective dual problems, the main focus of the remainder of this paper will be on the problems (TOTAL WEIGHT, NODE UPGRADING COST, SPANNING TREE) and (BOTTLENECK WEIGHT, NODE UPGRADING COST, SPANNING TREE).

1.3 Summary of Results

We present hardness results and approximation algorithms for the total cost and bottleneck cost spanning tree problems under the node based network improvement model. Specifically, our results are as follows.

1. We show that, unless NP$\subseteq$ DTIME($n^{\log\log n}$), there can be no polynomial time approximation algorithm with a performance guarantee of $(f(n),\ \alpha)$ for the problems (TOTAL WEIGHT, NODE UPGRADING COST, SPANNING TREE) and (BOTTLENECK WEIGHT, NODE UPGRADING COST, SPANNING TREE), for any $\alpha < \ln n$ and for any polynomial time computable function $f(n)$. Moreover, this result holds, with $f(n)$ being a polynomial in n, even when the difference between the maximum and the minimum edge weights in the problem instance is bounded by a polynomial function of the size of the graph.
2. We develop a polynomial time approximation algorithm with a performance of $(1,\ \mathcal{O}(\log n))$ for the (TOTAL WEIGHT, NODE UPGRADING COST, SPANNING TREE) problem (where n is the number of nodes in the graph), when the difference between the maximum and the minimum edge weights in the problem instance is bounded by a polynomial function of the size of the graph.
3. We develop a polynomial time approximation algorithm with a performance of $(1,\ 2\ln n)$ for the (BOTTLENECK WEIGHT, NODE UPGRADING COST, SPANNING TREE) problem. When all the nodes have unit costs, i.e., $c(v) = 1$ for all $v \in V$, we can improve our approximation result to a performance of $(1,\ 5 + 4\ln\Delta)$, where Δ is the maximum node degree in the input graph. The running time of the modified algorithm is $\mathcal{O}(n + m)$.

1.4 Related Work

Some node upgrading problems have been studied under a simpler model by Paik and Sahni [10]. In their model, upgrading a node causes the delay of each edge incident on that node to be reduced by a given constant factor $\delta < 1$. When both end points of an edge are upgraded, the delay of the edge is reduced by the factor δ^2. It is easy to see that 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 resulting 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. The approximation results presented here hold under our generalized model.

Network improvement problems where upgrading is carried out on edges rather than nodes have also been considered in the literature. For those results, we refer the interested reader to [1, 3, 5, 6, 9, 11].

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 $D \geq \mathrm{MST}_G(d_l)$, i.e., the weight of an MST with respect to d_l, since node upgrading cannot reduce the weight of the minimum spanning tree below this value. Thus, there exists always a subset of the nodes which, when upgraded, leads to an MST of weight at most D. We will also make the following assumption about the edge weights in a given instance:

Assumption 2.1 *The difference $d_{\max} - d_{\min}$ between the maximum edge weight $d_{\max} := \max_{e \in E} d(e)$ and the minimum edge weight $d_{\min} := \min_{e \in E} d_l(e)$ is bounded by a polynomial in the size of the input graph.*

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 $d(e)$ as the weight of each edge 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. Our algorithm finds a node with the best ratio improvement by using an approximate solution to the *Two Cost Spanning Tree Problem* defined below.

Definition 2.2 (*Two Cost Spanning Tree Problem*) *Given a connected undirected graph $G = (V, E)$, two cost values $c(e)$ and $l(e)$ for each edge $e \in E$ and a bound B, find a spanning tree T of G such that the total cost of T under the cost function c is at most B and the total cost of T under the cost function l is a minimum among all spanning trees that obey the budget constraint with respect to c.*

In the notation of [8], the above problem can be expressed as the bicriteria problem (c-TOTAL COST, l-TOTAL COST, SPANNING TREE). This problem has been addressed by Ravi and Goemans who obtained the following result [12].

Theorem 2.3 *For all fixed $\varepsilon > 0$, 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 after Figure 1. The description of the algorithm uses some terms whose definitions appear in the next subsection. The following theorem states the performance of our algorithm:

Theorem 2.4 *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 2.1, provides a performance guarantee of $(1,\ (1 + \varepsilon)^2 \mathcal{O}(\log n))$.*

Before we embark on a proof of the above theorem, 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.

ALGORITHM UPGRADE MST(Ω)

• *Input:* A graph $G = (V, E)$, three edge weight functions d, d_m, d_l, 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 with respect to 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 total weight no more than D, such that total cost of upgrading the nodes is no more than $(1+\varepsilon)\Omega \cdot \mathcal{O}(\log n)$, provided $\Omega \geq \mathrm{OPT}$, where 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.

Definition 2.5 *A graph $C = (V(C), E(C))$ is called a* claw, *if the edge set $E(C)$ is of the form $E(C) = \{ (v, w) : w \in V(C) \setminus \{v\} \}$ for some node $v \in V(C)$. If there are at most two nodes in the claw then we can choose any of the nodes as its* center. *Otherwise, the node with degree greater than one is the unique* center. *The vertices in the claw different from the center are said to be the* fingers *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 denote a minimum spanning tree with respect to d_W; that is, $T = \mathrm{MST}_G(d_W)$. Let C be a claw with nodes $M(C) \subseteq C$ marked. The *quotient cost* $q(C)$ of C is then

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

and to be $+\infty$ 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.

Our analysis essentially shows that in each iteration i, there exists a claw of quotient cost at most $\frac{2\mathrm{OPT}}{d_W(T)-D}$, where $T = T_{i-1}$ is the weight of an MST at the beginning of the iteration and $W = W_{i-1}$ are the nodes upgraded so far. We can then use a potential function argument to show that this yields a logarithmic performance guarantee. We now proceed with the details of the performance analysis.

PROCEDURE COMPUTE QC(Ω)

• *Input:* A graph $G = (V, E)$, a spanning tree T and an weight function d on E; $W \subseteq V$ is the set of upgraded nodes; a "guess value" 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.
   - 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 $c(u, w) = 0$.
   - For each edge (v, w) incident on v there are two parallel edges e_w^0 and e_w^1 between v and w in G_v. The edge e_w^0 has l-length $l(e_w^0)$ resulting from the upgrade of v and c-cost $c(e_w^0) = 0$. The edge e_w^1 has l-length resulting from the upgrade of *both* v *and* w and c-cost $c(w)$.
   - There is one edge (v, x) joining v to the dummy node x. This edge has length $l(v, x) = 0$ and cost $c(v, x) = c(v)$.
   - The bound B on the c-cost of the tree is set to K.

   (b) Using the algorithm mentioned in Theorem 2.3, 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 for *all* instances $I_{v,K}$ of *Two Cost Spanning Tree Problem* constructed above the algorithm from Theorem 2.3 fails to find a tree of cost at most $(1+\varepsilon)K$, 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 finger w is marked if and only if the edge in T_{v^*,K^*} between v^* and w has c-cost greater than zero.
6. Return the marked claw C.

• *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 $c(M(C)) \leq (1+\varepsilon)\Omega$.

2.3 Bounded Claw Decompositions

Definition 2.6 *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:*

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

Lemma 2.7 *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 2.8 *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:*

(i) For all edges $e \in T \cap T'$ we have $\varphi(e) = e$,
(ii) $d_{W \cup U}(\varphi(e)) \leq d_W(e)$ for all $e \in T$,
(iii) the "swaps" $e \longrightarrow \varphi(e)$ transforms T into T', and
(iv) $\sum_{e \in T}(d_W(e) - d_{W \cup U}(\varphi(e))) = d_W(T) - d_{W \cup U}(T')$. □

Lemma 2.9 *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*

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

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 2.7 to T' with the vertices in $\mathrm{OPT} \setminus W$ marked. The lemma shows that there is a 2-bounded claw decomposition of T' with respect to $\mathrm{OPT} \setminus W$. Let the claws be $C_1, \ldots, C_r$. In each claw C_j, the corresponding nodes $M(C_j) := C_j \cap (\mathrm{OPT} \setminus W)$ from $\mathrm{OPT} \setminus W$ are marked. Since the decomposition is 2-bounded with respect to $\mathrm{OPT} \setminus W$, 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 claw C_j does not exceed OPT, since we have marked only nodes from $\mathrm{OPT} \setminus W$. By Lemma 2.8, there exists a bijection $\varphi : T \to T'$ 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 this way 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 quotient cost $q(C)$. 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 for $j = 1,\ldots,r$, 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) now implies that C possesses the desired properties stated in the lemma. □

2.4 Finding a good claw in each iteration

Lemma 2.9 implies the existence of a marked with the required properties. We will now deal with the problem of finding a marked claw with a good quotient cost in each iteration.

Lemma 2.10 *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 2.9, 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 which by Lemma 2.9 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 2.3 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 2.4. 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 we have

$$\phi_{i+1} = \phi_i - (\mathrm{MST}_i - \mathrm{MST}_{i+1}) \overset{\text{Lemma 2.10}}{\leq} \left(1 - \frac{c_{i+1}}{\alpha \cdot \mathrm{OPT}}\right) \phi_i, \tag{7}$$

where $\alpha := 2(1+\varepsilon)^2$. We now use an analysis technique due to Leighton and Rao [7] to complete the proof. 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 $\phi_0 = \mathrm{MST}_0 - D \leq (n-1)(d_{\max} - d_{\min})$, where $d_{\max}$ and $d_{\min}$ denote the maximum and the minimum edge weight in the graph, defined in Assumption 2.1. It now follows from Assumption 2.1 that $\ln \phi_0 \in \mathcal{O}(\log n)$. Using this result in (9) yields

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

This completes the proof of Theorem 2.4. □

3 Upgrading Under Bottleneck Constraint

This section presents our approximation algorithm for (Bottleneck Weight, Node Upgrading Cost, Spanning Tree). Given a bound D on the bottleneck delay of a spanning tree, we partition the set of edges into four sets according to how many of the endpoints must be upgraded in order to make the delay of an edge fall below the threshold D. An edge of delay at most D

is called a *noncritical edge*. An edge e is said to be a *medium critical edge*, if $d(e) > D$ and $d_m(e) \leq D$. We say that e is a *highly critical edge*, if $d_m(e) > D$ and $d_l(e) \leq D$. Finally, any edge e with $d_l(e) > D$ is called a *useless edge*. Since node upgrading cannot reduce the delay of a useless edge to be below the threshold D, we assume without loss of generality that the set of useless edges is empty.

In the sequel we present our approximation algorithm for the (BOTTLENECK WEIGHT, NODE UPGRADING COST, SPANNING TREE) problem. In contrast to our algorithm from Section 2, we do *not* have to make any assumptions about the edge weights being polynomially bounded in the input size.

3.1 Overview of the Algorithm

The algorithm maintains a set W of nodes, a set F of edges and a set $\mathcal{G}$ of clusters which partition the vertex set V of the given graph G. The set $\mathcal{G}$ of clusters is initialized to be the set of connected components of the bottleneck graph bottleneck(G, d, D), containing only those edges e which have a delay $d(e)$ of at most D. The set W of upgrading nodes is initially empty.

The algorithm iteratively merges clusters until only one cluster remains. To this end, in each iteration it determines a node v of minimum *bottleneck quotient cost*. The bottleneck quotient cost of a node v is the ratio whose numerator is the cost of v plus the costs of some nodes adjacent to v in different clusters via highly critical edges, and whose denominator is the number of clusters which have nodes adjacent to v. A precise definition of the bottleneck quotient cost appears in Equation (10) in the algorithm depicted in Figure 2. This quotient cost measures the "average upgrading cost" of v and the vertices that are adjacent to v through highly critical edges. The algorithm then adds v and the nodes mentioned above to the solution set W and merges the corresponding clusters. The iteration stops, when only one cluster is left.

3.2 Analysis and Performance Guarantee

In the sequel, we use OPT to denote the minimum node upgrading cost to obtain a bottleneck spanning tree of given delay at most D. A key lemma used in establishing the performance guarantee of our algorithm is the following. A proof of this lemma is included in the appendix.

Lemma 3.1 *Let v be a node chosen in Step 5b of algorithm* UPGRADE BOTTLENECK SPANNING TREE *and let C denote the total cost of the nodes added to the solution set W in this iteration. Let there be q clusters before v is chosen and assume that in this iteration r clusters are merged. Then $C/r \leq \mathrm{OPT}/q$.* □

Using the above lemma, the following theorem, which gives the performance guarantee and the running time of our algorithm, can be proven.

Theorem 3.2 *The algorithm in Figure 2 provides a performance guarantee of* $(1, 2\ln n)$ *for the* (BOTTLENECK WEIGHT, NODE UPGRADING COST, SPANNING TREE) *problem. The algorithm can be implemented to run in time $\mathcal{O}(nm\,\alpha(m,n))$, where α is the inverse of Ackerman's function.* □

Proof (Performance): The proof uses a potential function argument similar to the proof of Theorem 2.4. Thus, we only highlight the main steps. Assume that the algorithm uses f iterations of the loop and denote by $v_1, \ldots v_f$ the vertices chosen in Step 5b of the algorithm.

ALGORITHM UPGRADE BOTTLENECK SPANNING TREE

• *Input:* A graph $G = (V, E)$, three edge weight functions d, d_m, d_l, a node weight function c, and a number D which is a bound on the bottleneck weight of a minimum bottleneck spanning tree in the upgraded graph.

1. $G' :=$ bottleneck(G, d, D).
2. $G_1, \ldots, G_q :=$ connected components of G'.
3. Initialize the set of upgraded nodes: $W := \emptyset$.
4. $F :=$ set of edges of G'.
5. While $G' = (V, F)$ has more than one connected component
   (a) Assume that $\mathcal{G} = \{G_1 \ldots, G_q\}$ is the set of components.
   (b) Find a node $v \in V$ in the graph G minimizing the ratio

$$\min_{2 \le r \le q} \min_{\{G_1, \ldots, G_r\} \subseteq \mathcal{C}} \frac{c(v) + \sum_{j=1}^{r} c(v, G_j)}{r}. \tag{10}$$

   Here, the cost $c(v, G_j)$ is defined in the following way: If $v \in G_j$ or v is adjacent to a node in G_j via a medium critical edge, then $c(v, G_j) := 0$. If all the edges from v to G_j are highly critical, then $c(v, G_j)$ is defined to be the minimum cost of a vertex in C_j adjacent to v. If there is no edge between v and any node in G_j, then $c(v, G_j) := +\infty$.
   (c) Let v be the node and $G_1, \ldots, G_r$ be the components in $\mathcal{C}$ chosen in Step 5b above, where w.l.o.g. $v \in G_1$.
   (d) Let $e_2, \ldots, e_r$ be a set of edges in G connecting v to $G_2, \ldots, G_r$, respectively.
   (e) $F := F \cup \{e_2, \ldots, e_r\}$, i.e., merge $G_1, G_2, \ldots, G_r$ into one component.
   (f) $W := W \cup \{v\}$.
   (g) For $i := 2, \ldots, r$: If $e_i = (v, v_i)$ is highly critical, then $W := W \cup \{v_i\}$.
   (h) $G_1, \ldots, G_{q'} :=$ connected components of $G' = (V, F)$
6. Output W.

• *Output:* A spanning tree with bottleneck weight no more than D, such that total cost of upgrading the nodes is no more than $2 \ln n \cdot \mathrm{OPT}$, where OPT denotes the optimal upgrading cost to reduce the weight of a bottleneck spanning tree to be at most D.

Figure 2: Approximation algorithm for node upgrading under a bottleneck constraint.

Let ϕ_j denote the number of clusters *after* choosing vertex v_j in this iteration. Thus, for instance, $\phi_0 = t$, the number of components at the beginning of the whole algorithm and $\phi_f = 1$, since we end up with one cluster. Let the number of clusters merged using vertex v_j be r_j and the total cost of the vertices added in that iteration be c_j.

Using Lemma 3.1 one can show the following inequality for the potential function ϕ_i:

$$\phi_f \leq \phi_0 \prod_{i=1}^{f} \left(1 - \frac{c_i}{2 \cdot \mathrm{OPT}}\right). \tag{11}$$

Taking natural logarithms on both sides and simplifying using the estimate $\ln(1-\tau) \leq -\tau$, we obtain

$$2 \cdot \mathrm{OPT} \cdot \ln \frac{\phi_0}{\phi_f} \geq \sum_{i=1}^{f} c_i. \tag{12}$$

Note that $\phi_0 \leq n := |V|$ and $\phi_f = 1$ and hence from (12) we get

$$\sum_{i=1}^{f} c_i \leq 2 \cdot \mathrm{OPT} \cdot \ln n. \tag{13}$$

Notice that the total cost of the nodes chosen by the algorithm is exactly the sum $\sum_{i=1}^{f} c_i$. This completes the proof of Theorem 3.2. □

When the upgrading cost of each node is 1, we can prove the following result.

Theorem 3.3 *There is an approximation algorithm for the* (BOTTLENECK WEIGHT, NODE UPGRADING COST, SPANNING TREE) *problem with unit upgrading costs (i.e., $c(v) = 1$ for all $v \in V$), with a performance of $(1,\ 5 + 4\ln\Delta)$, where Δ is the maximum degree in the input graph. This algorithm can be implemented to run in time $O(n+m)$.* □

4 Concluding Remarks

We considered minimum cost network upgrading problems under two performance measures, namely, the weight of a minimum spanning tree and the bottleneck weight of a minimum bottleneck weight spanning tree in the upgraded network. We presented polynomial time approximation algorithms for these problems. For these problems, our algorithms produced solutions in which the budget constraints were satisfied. This is unlike many bicriteria network design problems where it is necessary to violate the budget constraint by a small factor in order to efficiently obtain a solution that is near-optimal with respect to the objective function [8].

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 2.1 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," *Proceedings of the 28th Annual ACM Symposium on the Theory of Computing* (STOC'96), Philadelphia, PA, May 1996, pp. 314–318.

[3] G. N. Frederickson and R. Solis-Oba, "Increasing the Weight of Minimum Spanning Trees", *Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms* (SODA'96), January 1996, pp. 539–546.

[4] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman and Co., San Francisco, CA, 1979.

[5] 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 (WG'96)*, Cadenabbia, Italy, June 1996, pp. 293–307.

[6] 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.

[7] 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 Conference on Foundations of Computer Science* (FOCS'88), Oct. 1988, pp. 422–431.

[8] M. V. Marathe, R. Ravi, R. Sundaram, S. S. Ravi, D. J. Rosenkrantz and H. B. Hunt III, "Bicriteria Network Design Problems", In *Proceedings of the 22nd International Colloquium on Automata, Languages and Programming* (ICALP'95), July 1995, Vol. 944 of *Lecture Notes in Computer Science*, pp. 487–498.

[9] J. Plesnik, "The Complexity of Designing a Network with Minimum Diameter", *Networks*, Vol. 11, 1981, pp. 77–85.

[10] D. Paik and S. Sahni, "Network Upgrading Problems," *Networks*, Vol. 26, 1995, pp. 45–58.

[11] C. Phillips, "The Network Inhibition Problem," *Proc. 25th Annual ACM Symposium on Theory of Computing* (STOC'93), San Diego,CA, May 1993, pp. 288–293.

[12] R. Ravi and M. X. Goemans, "The Constrained Minimum Spanning Tree Problem", *Proc. Scandinavian Workshop on Algorithmic Theory* (SWAT'96), Reykjavik, Iceland, July 1996.

Appendix

A Hardness Results

The following result states the complexity of obtaining near-optimal solutions to the problems (TOTAL WEIGHT, NODE UPGRADING COST, SPANNING TREE) and (BOTTLENECK WEIGHT, NODE UPGRADING COST, SPANNING TREE).

Theorem A.1 *Unless* NP $\subseteq$ DTIME($n^{\log\log n}$), *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. *Moreover, this result holds with* $f(n) = n^k$ *being any polynomial, even if the difference between the maximum edge weight and the minimum edge weight in the problem instance is bounded by a polynomial function of the size of the graph.* □

We prove the above result by using a reduction from the MIN SET COVER problem and Feige's result [2] on the non-approximability of MIN SET COVER. It also follows from our proof of Theorem A.1 that similar hardness results hold for the problems (NODE UPGRADING COST, TOTAL WEIGHT, SPANNING TREE) and (NODE UPGRADING COST, BOTTLENECK WEIGHT, SPANNING TREE).

Recall that an instance of MIN SET COVER consists of a finite set Q of ground elements $\{q_1, \ldots, q_n\}$, a collection $\mathcal{F} = \{Q_1, \ldots, Q_m\}$ of subsets of Q and the requirement is to pick a minimum number of sets from $\mathcal{F}$ such that their union equals Q. The proof of Theorem A.1 uses Feige's result concerning the nonapproximability of the MIN SET COVER problem stated below.

Theorem A.2 (Feige [2]) *Unless* NP $\subseteq$ DTIME($n^{\log\log n}$), *the* MIN SET COVER *problem, with a ground set* Q *of size* $|Q|$, *cannot be approximated in polynomial time within a performance factor of* $\beta < \ln |Q|$.

Proof of Theorem A.1: Suppose A is a polynomial time algorithm that provides a performance guarantee of $(f(n),\ \alpha)$, for some polynomial time computable function $f(n)$ and some value $\alpha < \ln n$, for the (TOTAL WEIGHT, NODE UPGRADING COST, SPANNING TREE) problem. We will show that A can be used to obtain a polynomial time approximation algorithm with a performance guarantee of α for MIN SET COVER. Theorem A.1 would then follow in view of Feige's result indicated above.

Given an instance of MIN SET COVER, we produce an instance of (TOTAL WEIGHT, NODE UPGRADING COST, SPANNING TREE) as follows. We first construct the natural bipartite graph, one side of the partition for set nodes Q_j, $j = 1, \ldots, m$, and the other for element nodes q_i, $i = 1, \ldots, n$. We insert an edge (Q_j, q_i) if $q_i \in Q_j$. We now add one more node R (the "root") and connect R to all the set nodes.

Note that the resulting graph is bipartite (with R and the element nodes on one side and the set nodes on the other side).

The current weight, medium weight and the low weight values for each edge are $f(n)\cdot(n+m)+1$, 1 and 1 respectively. The cost of upgrading R is set to zero. For every set node, the cost of upgrading is set to 1; for every element node, the cost of upgrading is set to $\lceil m \ln n \rceil + 1$. The budget on the weight of a spanning tree in the upgraded network is set to $n + m$. The resulting

instance of (TOTAL WEIGHT, NODE UPGRADING COST, SPANNING TREE) satisfies the following property.

Claim A.3 *Let* OPT *denote the number of sets in an optimum solution to the* MIN SET COVER *instance. Then there is a solution to the* (TOTAL WEIGHT, NODE UPGRADING COST, SPANNING TREE) *instance constructed above with an upgrading cost of at most* OPT.

Proof of Claim A.3: Let $\mathcal{F}^*$ be an optimal solution to the MIN SET COVER instance. Let us upgrade the node R and the nodes corresponding to the sets in $\mathcal{F}^*$. The cost of this upgrade is OPT. A spanning tree T in which each edge has weight 1 can now be constructed by attaching each set node to R and for each upgraded set node, attaching the nodes corresponding to the elements in that set. Clearly, the weight of T is $n + m$. □

Now, suppose we run A on the (TOTAL WEIGHT, NODE UPGRADING COST, SPANNING TREE) instance constructed above and examine the spanning tree T' produced by A for the upgraded network. Since A violates the budget constraint only by a factor $f(n)$, by the above claim, the weight of T' is at most $f(n) \cdot (m + n)$. Since the weight of each edge in the original (unupgraded) network was chosen as $f(n) \cdot (m + n) + 1$, T' cannot include any such edge. Therefore, by our construction, the weight of each edge of T' is exactly 1.

We also note that the cost of the upgrading set chosen by A is at most $\alpha \cdot \text{OPT} < m \ln n$. Since the upgrading cost of any element node (namely, $\lceil m \ln n \rceil + 1$) exceeds this bound, A could not have upgraded any of the element nodes.

Thus, the upgrading set chosen by A consists of some set nodes and possibly the node R. Therefore, the number of upgraded set nodes is at most $\alpha \cdot \text{OPT}$. In T', each element node must be attached to an upgraded set node; otherwise, the weight of such an edge would be greater than 1. In other words, the sets corresponding to the upgraded set nodes form a cover of size at most $\alpha \cdot \text{OPT}$ for the MIN SET COVER instance. Thus, using A, we have constructed a polynomial time approximation algorithm with a performance guarantee $\alpha < \ln n$ for the MIN SET COVER problem. This completes the proof for the (TOTAL WEIGHT, NODE UPGRADING COST, SPANNING TREE) problem.

When $f(n)$ is a polynomial in n, we obtain the non-approximability result for (TOTAL WEIGHT, NODE UPGRADING COST, SPANNING TREE) for the case when the difference between the maximum edge weight and the minimum possible edge weight is polynomially bounded.

We note that the above construction also yields the hardness result for approximating the (BOTTLENECK WEIGHT, NODE UPGRADING COST, SPANNING TREE) problem. The reason is that our transformation leads to a spanning tree in which each edge has a weight of 1 in the upgraded network; that is, the minimum bottleneck cost of the tree is 1. □

By a slightly different transformation from SET COVER, it can be shown that the (TOTAL WEIGHT, NODE UPGRADING COST, SPANNING TREE) and (BOTTLENECK WEIGHT, NODE UPGRADING COST, SPANNING TREE) problems remain NP-hard even when each node has an upgrading cost of 1.

B Proofs for Upgrading under Bottleneck Constraint

B.1 Claw Partitions

While in Section 2 our main tool for decomposing the optimal solution was the notion of a bounded claw decomposition, for the bottleneck upgrading problem, we introduce the notion of a *claw partition*, which is formally defined below.

Definition B.1 *Let $G = (V, E)$ be a graph with node set V. A* claw partition *of V in G is a collection of node-disjoint nontrivial claws, which are all subgraphs of G and whose vertices form a partition of V.*

Thus, a claw partition can be viewed as a relaxation of a 1-bounded claw decomposition where we do not require each edge of the graph G to appear in one of the claws. The following lemma can be proved by an easy induction on $|V|$.

Lemma B.2 *Let G be a connected graph with node set V, where $|V| \geq 2$. Then there is a claw partition of V in G.* □

B.2 Proof of Lemma 3.1

Proof: Let T^* be an optimal tree with the nodes W^* be the upgraded nodes. Let $\mathrm{OPT} := c(W^*)$ be the cost of the optimal solution. Let $\mathcal{G} = G_1, \ldots, G_q$ be the clusters when the node v was chosen and let $T^*(v)$ be the graph obtained from T^* by contracting each G_j to a supernode. $T^*(v)$ is connected and contains all supernodes. We then remove edges (if necessary) from $T^*(v)$ so as to make it a spanning tree. Note that all the edges in this tree are either medium or highly critical.

Let $A \subseteq W^*$ be the set of nodes in the optimal solution that are adjacent to another cluster in $T^*(v)$. Clearly, the cost of these nodes is no more than OPT. Take a claw partition of $T^*(v)$. We now obtain a set of claws in the graph G itself in the following way: Initialize E' to be the empty set. For each claw in the partition with center G'_1 and fingers $G'_2, \ldots, G'_l$ we do the following: For each edge (G'_1, G'_j) the optimal tree T^* must have contained an edge (u, w) with $u \in G'_1$ and $w \in G'_i$. Notice that since this edge was either medium or highly critical, at least one of the vertices u and w must belong to $A \subseteq W^*$. We add (u, w) to E'.

It is easy to see that the subgraph of G induced by the edges in E' consists of disjoint nontrivial claws. Also, all edges in the claws were medium or highly critical and the total number of nodes in the claws is at least q. We need one more useful observation: If a claw center is not contained in A, then *all* the fingers of the claw must be contained in A, since the edges in the claw were medium or highly critical.

Let A_c be the set of nodes from A acting as centers in the just generated claws. Let A^{med} denote the fingers of the claws contained in A which are connected to their claw center via a medium critical edge, whereas A^{high} stands for the set of fingers adjacent to the center via a highly critical edge and also contained in A. For each claw with exactly two nodes we designate an arbitrary one of the nodes to be the center. Then by construction, A_c, A^{med} and A^{high} are disjoint. Therefore,

$$\mathrm{OPT} \geq \sum_{u \in A_c \cup A^{high}} c(u) + \sum_{u \in A^{med}} c(u). \tag{14}$$

For a node $u \in A_c$, let N_u denote the number of vertices in the claw centered at u. We have seen that if a center is not in A, then *all* the fingers belong to the optimal solution. Clearly, this can only happen, if the claw centered at u does not contain a highly critical edge. Thus, we can estimate the total number of nodes in the claws from above by summing up the cardinalities of the claws with centers in A and for all other claws adding twice the number of fingers. Hence

$$\sum_{u \in A_c} N_u + 2|A^{med}| \geq |\{\, w : w \text{ belongs to some claw } \}| \geq q, \tag{15}$$

since the total number of nodes in the claws is at least q.

We now estimate the first sum in (14). If $u \in A_c$, then the bottleneck quotient cost of u is at most the cost of u plus the cost of the fingers in the claw that are in A^{high} divided by the total number of nodes in the claw. This in turn is at least C/r by the choice of the algorithm in Step 5b. By summing up over all those centers, this leads to

$$\sum_{u \in A_c \cup A^{high}} c(u) \geq \frac{C}{r} \sum_{u \in A_c} N_u. \tag{16}$$

Now, for a node u in A^{med}, its bottleneck quotient cost is at most $c(u)/2$, which again is at least C/r. Thus

$$\sum_{u \in A^{med}} c(u) \geq \sum_{u \in A^{med}} 2\frac{C}{r} = 2|A^{med}| \cdot \frac{C}{r}. \tag{17}$$

Using (16) and (17) in (14) yields

$$\begin{aligned} \text{OPT} &\geq \sum_{u \in A_c \cup A^{high}} c(u) + \sum_{u \in A^{med}} c(u) \\ &\geq \frac{C}{r} \left(\sum_{u \in A_c} N_u + 2|A^{med}| \right) \\ &\overset{(15)}{\geq} \frac{C}{r} \cdot q. \end{aligned}$$

This proves the claim. □
