A Matter of Degree: Improved Approximation Algorithms for Degree-Bounded Minimum Spanning Trees

J. Könemann [*]
Carnegie Mellon University
GSIA
Pittsburgh, PA 15213
jochen@cmu.edu

R. Ravi [†]
Carnegie Mellon University
GSIA
Pittsburgh, PA 15213
ravi@cmu.edu

ABSTRACT

In this paper, we present a new bicriteria approximation algorithm for the *degree-bounded minimum spanning tree* problem. In this problem, we are given an undirected graph, a nonnegative cost function on the edges, and a positive integer B^*, and the goal is to find a minimum cost spanning tree T with maximum degree at most B^*. In an n-node graph, our algorithm finds a spanning tree with maximum degree $O(B^* + \log n)$ and cost $O(\texttt{opt}_{B^*})$ where $\texttt{opt}_{B^*}$ is the minimum cost of any spanning whose maximum degree is at most B^*. Our algorithm uses ideas from Lagrangean duality in a novel way. We show how a set of optimum Lagrangean multipliers yields bounds on both the degree and the cost of the computed solution.

1. INTRODUCTION

1.1 Motivation and formulation

In the design of computer networks a fundamental problem is that of transmitting a single information packet from a given source-host to a set of recipient-hosts. This problem is widely known as the *broadcast* or *multicast* problem, depending on whether we want to transmit the packet to all other hosts or to a specific subset of recipients. Both problems have been widely studied [4; 6; 17]. In particular, it is believed that the multicast problem will play an increasingly important role in data networks.

A naive solution to the multicast problem would be to implement it as a series of unicasts. In other words, the source sends a single packet to every recipient host. The routing is done independently for each of the unicasts. However, the cost of this approach in terms of bandwidth consumption becomes unacceptable if the number of hosts in the multicast group grows.

Graph theoretic ideas have turned out to be essential in the design of efficient network routing protocols. A physical network can be modeled as a complete graph where each host is associated with a node and an edge uv represents the *virtual link* between the corresponding hosts. Usually, edges of that graph are annotated by the *transmission delay* of the corresponding virtual link. A standard solution to broadcasting and multicasting problems is then to send packets along the edges of a minimum spanning tree rooted at the source node [17]. Every internal node duplicates the incoming message and sends it to its children.

However, a spanning tree might have a high *fan-out* out at certain internal nodes. Switches in point-to-point networks may vary in their ability to support multicasting. That is, some routers may not support multicasting at all and others may only support a limited number of copies they can make of an incoming packet [18]. Bauer and Varma [1] show that it is natural to model switch capabilities by node degrees in graphs.

The preceding discussion suggests that a solution to the multicasting problem should minimize the total transmission delay *and* the maximum degree of a vertex in the computed solution. Traditional approaches for this kind of *bicriteria* problem often compute the minimum solution under a linear combination of the two cost measures [2]. However, in the case of very disparate objectives these techniques do usually not produce useful solutions.

In this paper, we address a natural budgeted version of the *degree-bounded minimum spanning tree problem.* Here, we are given an undirected graph $G = (V, E)$, a cost function $c : E \to \mathbb{R}^+$ and a positive integer $B \geq 2$. We would like to find a spanning tree T of maximum vertex degree at most B and minimum cost. This formulation was first introduced in [16] and can be modeled by the following integer linear program.

$$\texttt{opt}_B = \min \quad \sum_{e \in E} c_e x_e \qquad \text{(IP)}$$

$$\text{s.t} \quad x(\delta(v)) \leq B \quad \forall v \in V \qquad (1)$$

$$x \in \texttt{SP}_G$$

Here, $\delta(v)$ denotes the set of all edges of E that are incident to v and $\texttt{SP}_G$ is the spanning tree polyhedron, that is, the

[*]Research supported in part by NSF CAREER grant 96-25279

[†]Research supported in part by NSF CAREER grant 96-25279

convex hull of edge-incidence vectors of spanning trees of G.

1.2 Previous work and our result

Ravi et al. [16] showed how to compute a spanning tree T of maximum degree $O(B \log(\frac{n}{B}))$ and total cost at most $O(\log n)\ \mathtt{opt}_B$. They generalize their ideas to Steiner trees, generalized Steiner forests and the node-weighted case.

Another result that is related to our work is given in a paper by Khuller, Raghavachari and Young [12]. The authors show how to compute a spanning tree of n points in the plane that has degree at most 3 (4) and weight at most 1.5 (1.25) that of a minimum weight spanning tree (without any degree constraints).

While the approximation factor of $O(\log n)$ on the cost of the solution cannot be improved substantially (via reductions from the set covering problem [13]) in the node-weighted case, improvements for the edge-weighted case were left open in [16]. Our main result is such an improvement and is stated in the following theorem. We denote the degree of a node v in tree T by $\delta_T(v)$. Let the maximum node degree in a tree T be denoted by $\Delta(T)$.

THEOREM 1. *There is a polynomial-time approximation algorithm that, given a graph $G = (V, E)$, a nonnegative cost function $c : E \to \mathbb{R}^+$, a constant $B^* \geq 2$ and a parameter $\omega > 0$, computes a spanning tree T such that*

1. *$\Delta(T) \leq \nu B^* + \log_b n$ for any arbitrary constant $b > 1$ and $\nu > (1 + \omega)b$*
2. *$c(T) < (1 + 1/\omega)\ \mathtt{opt}_{B^*}$.*

For instance, choosing $\omega = 1-\epsilon$ and $b = 2$ would yield a tree with degree at most $4B^* + \log_2 n$ and cost at most $2\ \mathtt{opt}_{B^*}$.

1.3 Technique: Lagrangean Duality

Our algorithm builds on the Lagrangean dual of (IP) resulting from *dualizing* the degree constraints. We denote its value by $\mathtt{opt}_{LD(B)}$.

$$\max_{\lambda \geq 0} \min_{T \in \mathtt{SP}_G} \{c(T) + \sum_{v \in V} \lambda_v(\delta_T(v) - B)\}. \qquad \text{(LD(B))}$$

For any fixed $\lambda \geq 0$, an optimum integral solution to IP is a feasible candidate for attaining the inner minimum above. Since the maximum degree of such a solution is at most B and $\lambda \geq 0$, it follows that $\mathtt{opt}_{LD(B)}$ is a lower bound on $\mathtt{opt}_B$.

PROPOSITION 1. *[14] $\mathtt{opt}_{LD(B)} \leq \mathtt{opt}_B$*

The interesting fact is that $\mathtt{opt}_{LD(B)}$ can be computed in polynomial time [14]. The result is a vector λ^B of optimum Lagrangean multipliers on the nodes and a set of optimum trees $\mathcal{O}^B$, all of which achieve the inner minimum for this set of multipliers. In other words, every tree $T^B \in \mathcal{O}^B$ minimizes the following objective:

$$c(T^B) + \sum_{v \in V} \lambda_v^B(\delta_{T^B}(v) - B).$$

Given λ^B, the task of finding a tree T that minimizes the above objective function is called the Lagrangean subproblem of LD(B).

Using $c^{\lambda^B}(uv) = c(uv) + \lambda_u^B + \lambda_v^B$ the last expression can be restated as

$$c^{\lambda^B}(T^B) - B \sum_{v \in V} \lambda_v^B \qquad (2)$$

Notice that for a given λ^B and B, the second term is a constant. Hence, any minimum spanning tree of G under cost c^{λ^B}, denoted by $\mathtt{MST}(G, c^{\lambda^B})$, is a solution for T.

An important feature of our algorithm is to relax the degree constraints slightly from B to $(1+\omega)B$ for some fixed $\omega > 0$ and show that there is a tree $T \in \mathcal{O}^{(1+\omega)B}$ that satisfies the conditions of Theorem 1.

This paper is organized as follows: in Section 2 we review results on the related *minimum-degree spanning tree problem*. In particular, we present the fundamental ideas from [5; 7]. In Section 3, we state our algorithm. Finally, we give the analysis of our procedure in Section 4.

2. MINIMUM DEGREE SPANNING TREES

Related to the BMST problem is the problem of minimizing the maximum degree of a spanning tree in some graph G (MDST). This problem is NP-hard since the *Hamiltonian path* problem is a special case. In fact, it is NP-complete to decide for any $k \geq 2$ whether G contains a spanning tree of maximum degree k [8].

Fürer and Raghavachari presented an approximation algorithm with an additive performance guarantee of one [7]: i.e., they describe a polynomial time algorithm that finds a spanning tree T of G such that $\Delta(T) \leq \Delta^* + 1$, where Δ^* denotes the minimum possible maximum degree over all spanning trees. In the same paper the authors also give a local search algorithm for the MDST problem. This approach yields a tree with maximum degree at most $b\Delta^* + \log_b n$ for any constant $b > 1$ and is obviously worse than the previously mentioned result. However, this latter technique turns out to be useful for our algorithm. Finally, Fischer noted that the local search algorithm can be adapted to find a minimum-cost spanning tree of approximately minimum maximum degree in an edge-weighted graph [5].

2.1 A local improvement algorithm

In this section, we explain the basic ideas from the local search algorithm for the MDST problem. We state the algorithm since we use it later. The procedure starts with a spanning tree T and tries to improve it by *swapping* non-tree edges against tree edges.

DEFINITION 1. *Given a tree T and a non-tree edge uv. Let $\mathcal{C}(uv)$ be the unique cycle in $T \cup \{uv\}$ and let $wz \in \mathcal{C}(e)$. We call the swap $\langle uv, wz \rangle$ an* improvement *for w if*

$$\max\{\delta_T(u), \delta_T(v)\} + 1 < \delta_T(w).$$

If an edge swap $\langle uv, wz \rangle$ is an improvement step for either w or z then the maximum degree of the nodes u, v, w and z decreases as a result of the swap; We call such a swap simply an improvement.

The algorithm in [7] performs improvement steps as long as possible. In fact, it is not hard to see that starting with an arbitrary tree, the number of possible improvements is finite. We end up with a *locally optimal tree.*

DEFINITION 2. *A tree T is called* locally optimal *(LOT) if no vertex degree can be decreased by applying an improvement swap.*

Computing a locally optimal tree might be too ambitious a goal however. In fact, it is not known how to do this in polynomial time. However, the analysis in [7] shows that it is enough to compute a *pseudo-optimal* tree.

DEFINITION 3. *A tree T of maximum degree $\Delta(T)$ is called* pseudo-optimal *(POT) if for all vertices v with $\Delta(T) - \lceil \log_b n \rceil \leq \delta_T(v) \leq \Delta(T)$, no improvement step for v is applicable. Here b is an arbitrary constant bigger than one.*

Fischer's adaptation [5] of the algorithm from [7] computes a minimum-cost spanning tree of approximately minimum maximum degree. To obtain his algorithm we have to make two small changes to the procedure from [7]. First, instead of starting with an arbitrary spanning tree, we start with a minimum-cost spanning tree. Second, an improvement step must be *cost neutral*. That is, for an improvement step $\langle uv, wz \rangle$ to be applicable we must have $c_{uv} = c_{wz}$. Algorithm 1 states the procedure.

Algorithm 1 The algorithm `PLocal` computes a pseudo-optimal tree.

```
Given: graph G = (V, E) and cost function c : E → IR+
T ← MST(G, c)
while T is not pseudo optimal do
   Identify cost neutral improvement ⟨uv, wz⟩.
   T ← T − wz + uv
end while
```

2.2 Analysis and running time

In what follows we highlight the major ideas from the analysis from [7]. Let a set $W \subseteq V$ be such that for a given graph $G = (V, E)$, the graph $G[V - W]$ has t connected components. A spanning tree of G has to connect these t components *and* the nodes from W. We need exactly $t + |W| - 1$ edges going between the nodes of W and the t connected components to accomplish this. Each of these edges must be incident a node from W. Hence averaging yields a lower bound of $(t + |W| - 1)/|W|$ on the maximum degree Δ^* of T.

PROPOSITION 2. *[7] Let W be a set of size w whose removal splits G into t components. Then $\Delta^* \geq \lceil \frac{w+t-1}{w} \rceil$.*

The set W *bears witness* to the fact that $\Delta^* \geq \lceil \frac{w+t-1}{w} \rceil$ and is therefore referred to as a *witness set*. The following easy proposition will be used in the later analysis.

PROPOSITION 3. *[7] For any constant $b > 1$ and a tree T, let S_d be the set of nodes that have degree at least d in T. Then, there is a*

$$d \in \{\Delta_T - \lceil \log_b n \rceil + 1, \ldots, \Delta_T\}$$

such that $|S_{d-1}| \leq b|S_d|$.

The main theorem is the following.

THEOREM 2. *[7] If T is a pseudo-optimal MWST, then $\Delta_T < b\Delta^* + \lceil \log_b n \rceil$ for any constant $b > 1$. Moreover, a pseudo-optimal MWST can be computed in polynomial time.*

Proof: Given a constant $b > 1$, choose d as in Proposition 3. That is, we have $|S_{d-1}| \leq b|S_d|$. Recall that S_d contains the nodes of degree at least d in the tree T.
Removing S_d from T leaves us with a forest F. By the definition of S_d, we know that F has at least

$$|S_d|d - 2(|S_d| - 1)$$

trees. This is, since every node in S_d has degree at least d in T and there are at most $|S_d| - 1$ edges going between nodes of S_d.
This means, that we need at least

$$|S_d|d - 2(|S_d| - 1) + |S_d| - 1$$

edges to connect up the trees of F and the nodes of S_d. By the pseudo-optimality of T, each these edges must be incident to some node in S_{d-1}. Hence, the minimum maximum degree of any tree on $F \cup S_d$ is at least the average degree it induces on a node of S_{d-1}, viz.

$$\frac{|S_d|(d-1) + 1}{|S_{d-1}|}.$$

Using $|S_d| \geq |S_{d-1}|/b$ we derive

$$\Delta^* > \frac{d-1}{b}.$$

Using the range of d we obtain $\Delta(T) < b\Delta^* + \lceil \log_b n \rceil$.
For the running time, note that each improvement step can be implemented in polynomial time. We need to bound the number of iterations. The proof uses a potential function argument. Define the potential of a vertex v as

$$\Phi(v) = 3^{\delta_T(v)}$$

where T is the current tree. The total potential is the sum over all vertex potentials, that is

$$\Phi(T) = \sum_{v \in V} \Phi(v).$$

Now, an improvement step in Algorithm 1 improves the degree of a vertex $v \in S_d$ with $\delta_T(v) = d$ and $d \geq \Delta(T) - \lceil \log_b n \rceil + 1$. Hence, the reduction in the potential is going to be at least

$$(3^d + 2 \cdot 3^{d-2}) - 3 \cdot 3^{d-1} = 2 \cdot 3^{d-2}.$$

Using the range of d we can lower bound the right hand side of the last equality by

$$3^{\Delta(T) - \log_b n - 1} = \Omega\left(\frac{3^{\Delta(T)}}{n}\right).$$

The potential $\Phi(T)$ of the tree T is at most $n3^{\Delta(T)}$. This implies that the overall decrease of the potential due to the improvement step is

$$\Omega\left(\frac{\Phi(T)}{n^2}\right)$$

In other words, we reduce the potential by at least a polynomial factor in each iteration. In $O(n^2)$ iterations the reduction is by a constant factor. Hence, the algorithm needs $O(n^3)$ improvement steps in total. ∎

3. THE BMST-ALGORITHM

In this section, we describe our algorithm for the BMST problem. It uses the Lagrangean formulation LD(B) from the introduction and Algorithm 1.

The input to our algorithm consists of a graph G, a non-negative cost function c, a degree bound B^* and a positive constant ω. Let $B = (1+\omega)B^*$.

Algorithm 2 Our algorithm for the BMST problem

Given: graph $G = (V, E)$, a cost function $c : E \to \mathbb{R}^+$, a degree bound $B^* \geq 2$ and a parameter $\omega > 0$.
$B \leftarrow (1+\omega)B^*$
$\lambda^B \leftarrow \texttt{Solve}(LD(B))$
$T^a \leftarrow \texttt{PLocal}(G, c^{\lambda^B})$

Since the optimum Lagrange multipliers and pseudo-optimal MWSTs can be computed in polynomial time [7; 14], Algorithm 2 runs in polynomial time. In the sequel we use $\mathcal{O}^B$ to denote the set of all optimum spanning trees. That is, $\mathcal{O}^B$ contains all minimum-weight spanning trees of G for cost function c^{λ^B}.

4. ANALYSIS

In this section we prove Theorem 1. First we show that the cost $c(T^a)$ of our tree T^a is small. Then, we prove that T^a has low maximum degree in several steps. Our proofs will make heavy use of techniques from Lagrangean duality and of results on the spanning tree polyhedron from [3]. We provide the details on demand.

4.1 The cost of T^a

Recall that $\texttt{opt}_{LD(B)} \leq \texttt{opt}_B$ from Proposition 1. Unfortunately, $\texttt{opt}_{LD(B)} = \texttt{opt}_B$ is not true in general. There might be a *duality gap*. However, it can be shown that $\texttt{opt}_{LD(B)}$ equals the optimum objective function value of the relaxation of (IP). The proof is insightful. Hence we present it here.

THEOREM 3. *[14]* $\texttt{opt}_{LD(B)} = \min\{c(T) : T \in \texttt{SP}_G, \forall v \in V : \delta_T(v) \leq B\}$

Proof: We can restate (LD(B)) as the following linear program in variables η and λ. Recall that we denote its maximum objective function value by $\texttt{opt}_{LD(B)}$.

$$\begin{aligned} \max \quad & \eta \\ \text{s.t.} \quad & \eta \leq c(T) - \sum_{v\in V} \lambda_v (B - \delta_T(v)) \quad \forall T \in \texttt{SP}_G \\ & \lambda \geq 0 \end{aligned} \tag{3}$$

The first block of constraints states that η is the cost of any spanning tree T of G with respect to the Lagrangean function (2). The maximization objective of (3) forces η to attain the best possible cost. Writing down the dual of the last program yields

$$\begin{aligned} \min \quad & c(\sum_{T\in \texttt{SP}_G} \alpha_T T) \\ \text{s.t.} \quad & \sum_{T\in \texttt{SP}_G} \alpha_T = 1 \\ & \sum_{T\in \texttt{SP}_G} \alpha_T \delta_T(v) \leq B \sum_{T\in \texttt{SP}_G} \alpha_T = B \quad \forall v \in V \\ & \alpha \geq 0 \end{aligned} \tag{4}$$

Note that $T' = \sum_{T\in \texttt{SP}_G} \alpha_T T$ is a convex combination of trees in $\texttt{SP}_G$. Also, observe that $\sum_{T\in \texttt{SP}_G} \alpha_T \delta_T(v)$ is precisely the degree $\delta_{T'}(v)$ of this fractional tree at node v. These observations yield the theorem. ■

The theorem has a nice corollary that we use.

COROLLARY 1. *There exists a convex combination* $T^f = \sum_{T\in\mathcal{O}} \alpha_T T$ *such that* $\forall v \in V : \delta_{T^f}(v) \leq B$ *and* $\lambda^*_v > 0$ *only if* $\delta_{T^f}(v) = B$.

Proof: This follows from complementary slackness applied to the optimum solutions of (3) and (4). ■

We now prove that $c(T^a)$ is small.

LEMMA 1. *For all trees* $T \in \mathcal{O}^B$ *we have* $c(T) < (1 + 1/\omega)\, \texttt{opt}_{B^*}$.

Proof: Let λ^B be the vector of optimal Lagrangean multipliers for (LD(B)). Recall that $\mathcal{O}^B$ is the set of minimum-cost spanning trees under c^{λ^B} and that $B = (1+\omega)B^*$

The following inequality holds for every $T \in \mathcal{O}^B$:

$$\begin{aligned} \sum_{v\in V} \lambda^B_v (\delta_T(v) - B^*) & \leq c(T) + \sum_{v\in V} \lambda^B_v (\delta_T(v) - B^*) \\ & \leq \texttt{opt}_{LD(B^*)} \end{aligned} \tag{5}$$

In the first inequality we just added $c(T)$. Note, that the right hand side of the first line is just the Lagrangean objective function (2) for B^*. Recall that T is a minimum spanning tree with respect to c^{λ^B}. Moreover, λ^B is a feasible set of multipliers for (LD(B^*)). Hence, the second inequality follows.

We also have

$$\begin{aligned} c(T) & = c(T) + \sum_{v\in V} \lambda^B_v (\delta_T(v) - B^*) + \sum_{v\in V} \lambda^B_v (B^* - \delta_T(v)) \\ & \leq \texttt{opt}_{LD(B^*)} + \sum_{v\in V} \lambda^B_v (B^* - \delta_T(v)) \end{aligned}$$

where the inequality follows from (5). Applying Proposition 1 and the fact that $\delta_T(v) \geq 1$ for all $v \in V$ leads to

$$c(T) < \texttt{opt}_{B^*} + B^* \sum_{v\in V} \lambda^B_v.$$

In the remainder of this proof we will derive the inequality $B^* \sum_{v\in V} \lambda^B_v \leq \texttt{opt}_{B^*}/\omega$. This yields the lemma. From Corollary 1, we know that there is a convex combination

$$T^f = \sum_{T\in\mathcal{O}^B} \alpha_T T$$

such that $\lambda^B_v > 0$ only if $\delta_{T^f}(v) = B$.

We derive a new inequality by summing over all $T \in \mathcal{O}^B$, α_T times the inequality (5) for each T. We obtain

$$\sum_{T\in\mathcal{O}^B} \alpha_T \left(\sum_{v\in V} \lambda_v^B (\delta_T(v) - B^*) \right) \leq \texttt{opt}_{LD(B^*)} \sum_{T\in\mathcal{O}^B} \alpha_T \tag{6}$$

The right hand side is equivalent to $\texttt{opt}_{LD(B^*)}$ because $\sum_{T\in\mathcal{O}^B} \alpha_T = 1$. Reordering the left hand side yields

$$\sum_{v\in V} \lambda_v^B \left(\left(\sum_{T\in\mathcal{O}^B} \alpha_T \delta_T(v) \right) - B^* \right)$$

Instead of summing over all $v \in V$ it suffices to sum over v, where $\lambda_v > 0$. For such v, we have

$$\sum_{T\in\mathcal{O}^B} \alpha_T \delta_T(v) = B$$

by Corollary 1. Using $B = (1+\omega)B^*$ it follows that the left hand side of (6) is equivalent to

$$\omega B^* \sum_{v\in V} \lambda_v^B$$

and this finishes the proof of the lemma. ■

4.2 Background: Yet another MST algorithm

Before we present a proof for the upper bound on $\Delta(T^a)$, we have to discuss some results on the spanning tree polyhedron from [3]. We give a complete description of the convex hull of all spanning trees. Furthermore, we describe a dual based procedure from [3] to compute those trees. Ideas from this procedure will turn out to be useful in our later discussion. A *feasible partition* of vertex set V is a set $\pi = \{V_1, \ldots, V_k\}$ such that the V_i are subsets of V and pairwise disjoint. Moreover, $V = V_1 \cup \ldots \cup V_k$ and the induced subgraphs $G[V_i]$ are connected. Let G_π denote the (multi-) graph that has one vertex for each V_i and edge (V_i, V_j) occurs with multiplicity $|\{(v_i, v_j) : v_i \in V_i, v_j \in V_j\}|$. In other words, G_π results from G by contracting each of the V_i to single nodes. Define the *rank* $r(\pi)$ of π as the number of nodes of G_π. The main result from [3] is stated in the following theorem.

THEOREM 4. *[3]* $SP_G = \{x \in \mathbb{R}^m : \sum_{e\in E(G_\pi)} x_e \geq r(\pi) - 1 \quad \forall$ *feasible partitions* $\pi\}$.

Using this formulation, the dual of the minimum spanning tree problem is as follows.

$$\begin{aligned} \max \quad & \sum_{\pi} (r(\pi) - 1) y_\pi & \text{(DSP)} \\ \text{s.t.} \quad & \sum_{\pi: e\in E(G_\pi)} y_\pi \leq c_e & \\ & x \geq 0 & \end{aligned}$$

Intuitively, each partition π with $y_\pi > 0$ *loads* the edges which go between the components of π, i.e. the edges from $E(G_\pi)$. In a feasible dual solution, we *pack* partitions so that no edge is overloaded. By complementary slackness we know that $e \in E$ only if $\sum_{\pi: e\in E(G_\pi)} y_\pi = c_e$, i.e. $e \in T$ only if the dual constraint for edge e is *tight*. We will call edges e with tight corresponding dual constraints *tight edges*. The following algorithm of Chopra [3] produces a pair (x, y) of primal and dual solutions to the minimum spanning tree problem at the same time.

Algorithm 3 A minimum spanning tree algorithm

```
1: k ← 0, G_0 ← G, c_0(e) ← c(e)   ∀e ∈ E
2: repeat
3:    k ← k + 1
4:    e_k ← argmin_{e∈E(G_k)} c_k(e)
5:    c_{k+1}(e) = c_k(e) − c_k(e_k)
6:    G_{k+1} ← contract(G_k, e_k)
7: until |V(G_{k+1})| = 1
```

The algorithm regards the current active node set $V(G_k)$ as a partition π_k. Note, that a single node in $V(G_k)$ can stand for a set in V since we perform a series of contractions in the course of the algorithm. Notice, that step 4 does not necessarily return a unique edge e. There may be more than one edge $e \neq e_k$ in $E[G_k]$ with $c_k(e_k) = c_k(e)$. The algorithm then goes through a series of iterations contracting one of these equivalent edges at a time (if it continues to cross the current partition).

Let π_k denote the partition defined by G_k. Picking edge e_k with respect to the edge costs c_k and subtracting its cost from all other edge costs can be viewed as growing partition π_k by $c_k(e_k)$. In iteration k edge e_k becomes tight. Hence, the primal solution that Algorithm 3 computes is $T = \{e_k : 1 \leq k \leq n-1\}$. The dual solution computed by Algorithm 3 is

$$y_\pi = \begin{cases} c_k(e_k) & : \quad \pi = \pi_k \\ 0 & : \quad \text{otherwise.} \end{cases}$$

Observe that there may be tight non-tree edges. A tight edge $e \notin T$ forms a unique cycle $\mathcal{C}(e)$ in T. There is another edge $e' \in \mathcal{C}(e)$ that became tight at the same time as e. Hence, $c(e') = c(e)$ and $T - e' + e$ is also a minimum spanning tree. We say e is *swappable* against e' in T.

Finally, observe that we can use the latter dual to reformulate (LD(B)).

$$\begin{aligned} \max_{\lambda\geq 0} \quad \max \quad & \sum_{\pi} (r(\pi) - 1) y_\pi - \sum_{v\in V} \lambda_v B & (LD'(B)) \\ \text{s.t.} \quad & \sum_{\pi: uv\in E(G_\pi)} y_\pi \leq c_{uv} + \lambda_u + \lambda_v \quad \forall uv \in E & \\ & y \geq 0 & \end{aligned}$$

In the sequel, we use the abbreviation

$$z(\lambda, y) := \sum_{\pi} (r(\pi) - 1) y_\pi - \sum_{v\in V} \lambda_v B.$$

4.3 The Maximum Degree of T^a: Overview

The overview of the proof is as follows: We show, that the existence of a high-degree vertex leads to an improvement of the Lagrangean dual solution. However, we started off with an optimum Lagrangean solution. Hence we have derived a contradiction.

LEMMA 2. $\Delta_{T^a} \leq \nu B^* + \lceil \log_b n \rceil$ *for constants* ν, b *and* ω *such that* $\nu > (1+\omega)b$ *and* $b > 1$.

Proof: Assume for the sake of contradiction that $\Delta_{T^a} > \nu B^* + \lceil \log_b n \rceil$. Again, let S_d denote the set of nodes that has degree at least d in T^a. Then, choose a $d \in \{\Delta_{T^a} - \lceil \log_b n \rceil, \ldots, \Delta_{T^a}\}$ such that $|S_{d-1}| \leq b|S_d|$ (see Proposition 3). Also, assume that we are given the optimum dual solution (λ^B, y^B) for $(LD'(B))$. The idea of the following proof is to show that we can obtain a new feasible solution (λ, y) from (λ^B, y^B) such that

$$z(\lambda, y) > z(\lambda^B, y^B)$$

which would contradict the optimality of (λ^B, y^B).
Figure 1(a) shows the basic idea of the proof: By increasing the λ values of high degree nodes from S_d, we make the cost $c^{\lambda}_{uv} (= c_{uv} + \lambda_u + \lambda_v)$ of the edges uv incident to S_d nodes higher. For the sake of simplicity, assume that $S_d = \{v\}$. In terms of the dual solution, by increasing λ_v by ϵ, we create some *slack* in the edges incident to v. This allows further growth of the partition $\pi = \{\{v\}, T_1, \ldots, T_k\}$. The rank of this partition is $k+1$. Increasing the λ value of node v by ϵ decreases $z(\lambda, y)$ by $\epsilon B = \epsilon(1+\omega)B^*$. However, we should also be able to increase y_π by ϵ. This would yield an increase in $z(\lambda, y)$ of $\epsilon(r(\pi) - 1) = \epsilon k$. We show later that by our initial assumption that $\Delta_{T^a} > \nu B^* + \lceil \log_b n \rceil$ and by the choice of d, we can choose k to satisfy

$$k > (1+\omega)B^*.$$

This leads to a net-increase in the dual of $\epsilon(k - (1+\omega)B^*) > 0$, and hence to a better dual solution. That is a contradiction. The above discussion is meant to illustrate the proof idea and is informal and oversimplifying.

4.4 The Maximum Degree of T^a: Details

Call the tree edges $p_1, \ldots, p_k$ that are incident to S_d nodes *peer edges* and assume $c^{\lambda^B}(p_1) \leq \ldots \leq c^{\lambda^B}(p_k)$. Let T_j be the component of T^a that is connected to an S_d node by edge p_j. Observe, that T_j and T_i might be identical even for $i \neq j$. Also, T_j might be empty for an edge p_j that goes between two nodes in S_d. Now, let

$$\lambda_v = \begin{cases} \lambda^B_v + \epsilon & : \quad v \in S_{d-1} \\ \lambda^B_v & : \quad \text{otherwise} \end{cases}$$

for a small constant $\epsilon > 0$.
Call the non-tree edges $e = uv$ between the components $T_1, \ldots, T_k$ with

$$\sum_{\pi : e \in E(\pi)} y^B_\pi = c^\lambda(e)$$

cross edges. Note that the latter equality can only hold for $u, v \notin S_{d-1}$ since we increased the λ-values of nodes in S_{d-1}. Hence cross edges are tight non-tree edges that are not incident to S_{d-1} nodes. Let $\mathcal{R} = \{r_1, \ldots, r_l\}$ be all cross edges such that $c^{\lambda^B}(r_1) \leq \ldots \leq c^{\lambda^B}(r_l)$. Why do we consider those edges? The simplified explanation of the proof idea from above increases the dual variable corresponding to partition $\pi = \{\{v\}, T_1, \ldots, T_k\}$. However, this might not be possible! There may be *cross edges* linking different subtrees T_i. Increasing y^B_π would yield an infeasible dual solution, that is $\sum_{\pi : e \in \pi} y_\pi > c(e)$ for such cross edges (see figure 1(b)). Thus, the final version of the dual update hinted at in the proof idea will be more intricate.
Each cross edge r has a set of associated peer edges

$$P(r) = \{e \in \mathcal{C}(r) : e = p_i \text{ for some } 1 \leq i \leq k\}$$

where $\mathcal{C}(r)$ denotes the unique cycle in $T^a \cup \{r\}$. We also define sets $R(p)$ for each peer edge p:

$$R(p) = \{r \in \mathcal{R} : p \in P(r)\}.$$

In other words, $R(p)$ contains all cross edges r such that $p \in \mathcal{C}(r)$. Look at Figure 2 for an example.

PROPOSITION 4. *Let r be a cross edge in a pseudo-optimal tree. We must have*

$$c^{\lambda^B}(p) < c^{\lambda^B}(r)$$

for all $p \in P(r)$.

Proof: By definition, a cross edge is not incident to an S_{d-1} node. Hence, $c^{\lambda^B}(r) \leq c^{\lambda^B}(p)$ would contradict the pseudo-optimality of our tree T. ■
Alternatively, we must have $c^{\lambda^B}(p) < c^{\lambda^B}(r)$ for all $r \in R(p)$. The last proposition has one important corollary.

COROLLARY 2. *For each cross edge r, there is a partition π such that*

1. $r \in E[\pi]$
2. $P(r) \cap E[\pi] = \emptyset$

and $y^B_\pi > 0$.

4.5 Simplifying assumptions and consequences

For now, assume that

[A1] No two nodes in S_d are neighbors in T^a

[A2] Peer edges have pairwise different costs with respect to c^{λ^B}.

We relax these assumptions later but this simplifies the discussion for now. Note, that this implies that none of the T_j components are empty.
For each edge p_j we identify two partitions π^1_j and π^2_j such that

- π^1_j is the partition of lowest rank with $p_j \in E(\pi^1_j)$ and $y^B_{\pi^1_j} > 0$
- π^2_j is the partition of highest rank with $p_j \notin E(\pi^2_j)$ and $y^B_{\pi^2_j} > 0$

The two assumptions stated above and Corollary 2 ensure that π^1_j and π^2_j are indeed well defined (e.g., if we don't require [A1] we might have one partition for 2 different nodes from S_d).
If we think of y^B as being generated by Algorithm 3 then π^1_j corresponds to the partition just before edge p_j has been contracted and π^2_j is the partition just after the contraction. Recall, that in an iteration of algorithm 3, there can be many equivalent edges in a contraction step. The processing order of these edges, however, is arbitrary.
The following proposition is crucial for the proof.

PROPOSITION 5. *Suppose that for $e \in E$ and $1 \leq j \leq k$ we have that*

$$e \in E[\pi^1_j] \text{ and } e \notin E[\pi^2_j]. \tag{7}$$

Then, the following holds

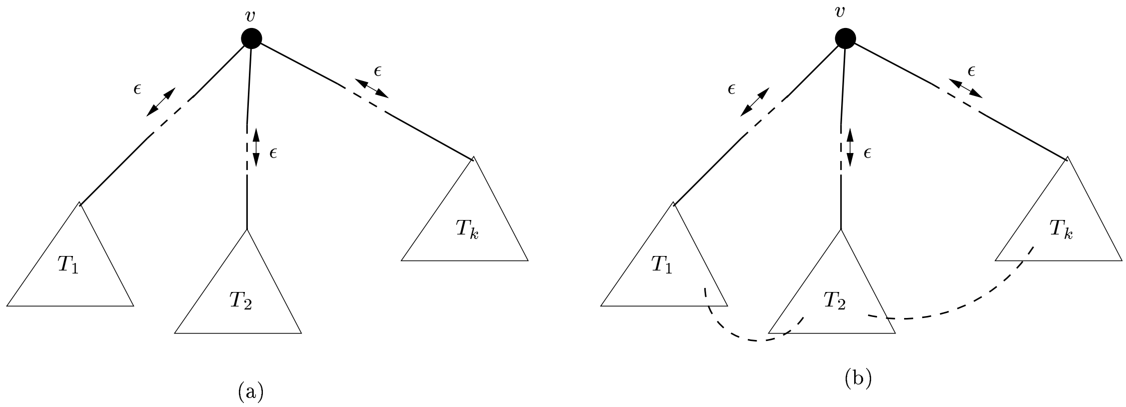

Figure 1: (a) We introduce some *extra space* in the edges incident to high degree nodes. (b) The dashed lines represent *cross edges*. Their tightness keeps us from increasing y_π for $\pi = \{\{v\}, T_1, \ldots, T_k\}$.

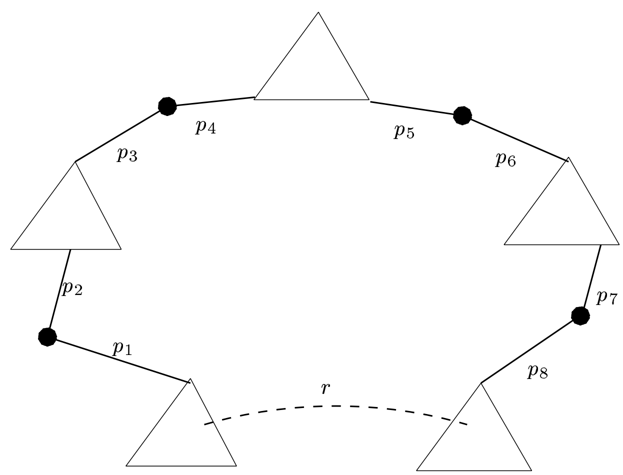

Figure 2: Each cross edge r has a set of associated peer edges $P(r)$ (here: $\{p_1, \ldots, p_8\} \subseteq P(r)$). Similarly, $r \in R(p_i)$ for $i = 1, \ldots, 8$.

1. $\forall$ *partitions* π *with* $r(\pi) \leq r(\pi_j^2)$ *and* $y_\pi^B > 0$ *we have* $e \notin E_\pi$
2. $\forall$ *partitions* π *with* $r(\pi) \geq r(\pi_j^1)$ *and* $y_\pi^B > 0$ *we have* $e \in E_\pi$.

Proof: To see this we can think of y^B as being generated by Algorithm 3. In the construction of y^B Algorithm 3 went through a series of partitions

$$V = \pi_1, \ldots, \pi_n \tag{8}$$

such that $r(\pi_i) = r(\pi_{i+1}) + 1$ for all $1 \leq i \leq n-1$, i.e. the above sequence is monotone with respect to the rank function r.
Also, for all π with $y_\pi^B > 0$ there exists a a $1 \leq j \leq n$ such that $\pi = \pi_j$. Let $\pi_j^1 = \pi_s$ and $\pi_j^2 = \pi_t$.
The fact that $e \in E(\pi_j^1)$ but $e \notin E(\pi_j^2)$ shows that edge e has been contracted in the process of moving from partition π_j^1 to π_j^2. It follows from the description of Algorithm 3 that $e \in \pi_j$ for all $1 \leq j \leq s$ and that $e \notin \pi_j$ for all $t \leq j \leq n$.
It follows from the rank-monotonicity of the sequence (8) that a partition π with $y^\pi > 0$ and $r(\pi) \leq r(\pi_j^2)$ corresponds to a partition π_j with $t \leq j \leq n$. Hence we have $e \notin E_\pi$. Similarly a partition π with $y_\pi^B > 0$ and $r(\pi) \geq r(\pi_j^1)$ must correspond to π_j with $1 \leq j \leq s$. Thus we have that $e \in E_\pi$. ■

The last proposition has two important corollaries.

COROLLARY 3. *Given an edge* $e \in E$. *Under assumption [A1] we can have*

$$e \in E_{\pi_j^1} \text{ and } e \notin E_{\pi_j^2}$$

for at most one $1 \leq j \leq k$.

Proof: Suppose we have $e \in E$ and $1 \leq j \leq k$ such that

$$e \in E_{\pi_j^1} \text{ and } e \notin E_{\pi_j^2}.$$

It follows from assumption [A1] and our choice of π_j^1 and π_j^2 that there cannot exist $1 \leq l \leq k, l \neq j$ with

$$r(\pi_j^1) \leq r(\pi_l^1) \leq r(\pi_j^2)$$

or

$$r(\pi_j^1) \leq r(\pi_l^2) \leq r(\pi_j^2).$$

That is, we have one of the two following cases:

1. $r(\pi_l^1) \leq r(\pi_j^2)$
2. $r(\pi_j^1) \leq r(\pi_j^2)$.

In case 1, we have that both $r(\pi_l^1)$ and $r(\pi_l^2)$ have rank at most $r(\pi_j^2)$. It follows from Proposition 5 that $e \notin E[\pi_l^1]$ and $e \notin E[\pi_l^2]$. Case 2 can be proven similarly. ■

COROLLARY 4. *For all $1 \leq j \leq k$ a cross edge $r \in \mathcal{R}$ is either in both $E(\pi_j^1)$ and $E(\pi_j^2)$ or in neither one.*

Proof: Assume for the sake of contradiction that there is a $1 \leq j \leq k$ and $r \in \mathcal{R}$ such that $r \in E(\pi_j^1)$ and $r \notin E(\pi_j^2)$ for some j. By the definition of π_j^2 we know that no π exists such that $r(\pi_j^2) < r(\pi) < r(\pi_j^1)$ and $y_\pi^B > 0$.
From Proposition 5, we know that for all partitions π with $y_\pi^B > 0$ and $r(\pi) \leq r(\pi_j^2)$ we must have that $r \notin E_\pi$ and $p_j \notin E_\pi$. We also know that for all π such that $r(\pi) \geq r(\pi_j^1)$ with $y_\pi^B > 0$ it must be true that $p_j, r \in E_\pi$. Intuitively, this means that the edges r and p_j are contained in exactly the same partitions. But this means that $c^{\lambda^B}(r) = c^{\lambda^B}(p_j)$. However, from the fact that $r \notin E[\pi_j^2]$, we know that $p_j \in P(r)$. This contradicts Proposition 4. ■
Observe that $p_j \in E[\pi_j^1]$ but $p_j \notin E[\pi_j^2]$. Thus, we have

$$r(\pi_j^1) - r(\pi_j^2) \geq 1. \tag{9}$$

Now, let

$$\epsilon = \min\{\min_{j=1,\ldots,k} y_{\pi_j^2}^B, \min_{e: s_e > 0} s_e\} \tag{10}$$

where s_e is the *slack* of edge e with respect to y^B is defined as follows:

$$s_e = c^{\lambda^B}(e) - \sum_{\pi: e \in E(\pi)} y_\pi^B. \tag{11}$$

4.6 A revised dual solution

It remains to define the new dual solution:

$$y_\pi = \begin{cases} y_\pi^B + \epsilon & : \quad \pi = \pi_j^1 \text{ for some } j \\ y_\pi^B - \epsilon & : \quad \pi = \pi_j^2 \text{ for some } j \\ y_\pi^B & : \quad \text{otherwise.} \end{cases}$$

We first prove feasibility of the new solution.

CLAIM 1. *For all edges $e \in E(G) : \sum_{\pi: e \in E(\pi)} y_\pi \leq c^\lambda(e)$.*

Proof: We prove the claim using two cases: either $e \cap S_{d-1} = \emptyset$ or $e \cap S_{d-1} \neq \emptyset$. We split the first case into 2 sub cases according to whether e is a cross edge or not.

1a $\underline{e \cap S_{d-1} = \emptyset \textit{ and } e \textit{ is not a cross edge.}}$ Suppose, for all j, either $e \in E(\pi_j^1)$ and $e \in E(\pi_j^2)$ or $e \notin E(\pi_j^1)$ and $e \notin E(\pi_j^2)$. In that case, the ϵ increase in the first partition is canceled out exactly by the ϵ decrease in the second one and we have

$$\sum_{\pi: e \in E(\pi)} y_\pi = \sum_{e \in E(\pi)} y_\pi^B. \tag{12}$$

The claim clearly holds. On the other hand, assume $e \in E(\pi_j^1)$ but $e \notin E(\pi_j^2)$ for some j. We know from Corollary 3 that this can happen for at most one such j.

Since e is not a cross edge we know that e is non-tight, i.e. $s_e > 0$. Now,

$$\begin{aligned} \sum_{\pi: e \in E(\pi)} y_\pi &= \epsilon + \sum_{e \in E(\pi)} y_\pi^B \\ &\leq s_e + \sum_{e \in E(\pi)} y_\pi^B \\ &= c^{\lambda^B}(e). \end{aligned}$$

The inequality follows from our choice of ϵ (see 10) and the last equality follows from (11). Since $e \cap S_d = \emptyset$, we know that $c^{\lambda^B}(e) = c^\lambda(e)$ and the claim follows.

1b $\underline{e \cap S_{d-1} = \emptyset \textit{ and } e \textit{ is a cross edge.}}$ By Corollary 4 we know that for all $1 \leq j \leq k$ either e is in both $E(\pi_j^1)$ and $E(\pi_j^2)$ or in none of them. Hence the argument from (12) applies also here.

2 $\underline{e \cap S_{d-1} \neq \emptyset.}$ In this case it might be the case that $e \in E(\pi_j^1)$ but $e \notin E(\pi_j^2)$ for exactly one j (see Corollary 3). That is

$$\sum_{\pi: e \in E(\pi)} y_\pi = \epsilon + \sum_{e \in E(\pi)} y_\pi^B = c_e^{\lambda^B} + \epsilon \leq c^\lambda(e).$$

The last equality follows from the fact that we have added ϵ to λ_v^B for all $v \in S_{d-1}$.

This finishes the proof of the claim. ■
We now compute $z(y, \lambda)$ and show that we can choose ν such that it is bigger than $z(y^B, \lambda^B)$. It follows from our previous discussion and from the definition of π_j^1 and π_j^2 that

$$\begin{aligned} z(y, \lambda) &= \sum_\pi (r(\pi) - 1) y_\pi - B \sum_{v \in V} \lambda_v \\ &= \sum_\pi (r(\pi) - 1) y_\pi^B - B \sum_{v \in V} \lambda_v^B + \\ &\quad \epsilon \sum_{j=1}^{k} (r(\pi_j^1) - r(\pi_j^2)) - \epsilon B |S_{d-1}| \qquad (13) \\ &\geq z(y^B, \lambda^B) + \epsilon(k - B|S_{d-1}|) \qquad (14) \end{aligned}$$

where k is the number of peer edges. We have to choose $k > B|S_{d-1}|$. How large is k? Since, by our assumption [A1], S_d is an independent set in T^a and by our choice of d, each node $v \in S_d$ has degree at least νB^*, we know that

$$k \geq \nu B^* |S_d|. \tag{15}$$

We also know that $|S_d| \geq |S_{d-1}|/b$. That is, we have to choose ν such that

$$\nu B^* \frac{|S_{d-1}|}{b} > B|S_{d-1}| = (1 + \omega) B^* |S_{d-1}|$$

which is equivalent to

$$\nu > (1 + \omega) b.$$

Hence, choosing $\nu > (1 + \omega) b$ suffices.

4.7 Taking care of the assumptions

It remains to relax the two assumptions [A1] and [A2]. First, we relax [A1].Assume we have $u, v \in S_d$ such that $(u, v) \in E(T^a)$. The problem in the above analysis is equation (15).

We are not any longer able to argue that $k \geq B|S_d|$. However, note that $c^{\lambda}(uv) = c_{uv}^{\lambda^B} + 2\epsilon$. The idea then is to use the extra ϵ to compensate for the decrease in k by one.
In fact, let $\mathcal{K}_1, \mathcal{K}_2$ be a partition of $\{1, \dots, k\}$ such that $j \in \mathcal{K}_1$ iff p_j has exactly one endpoint from S_d. We redefine y in the following way

$$y_\pi = \begin{cases} y_\pi^B + \epsilon & : \quad \pi = \pi_j^1, j \in \mathcal{K}_1 \\ y_\pi^B - \epsilon & : \quad \pi = \pi_j^2, j \in \mathcal{K}_1 \\ y_\pi^B + 2\epsilon & : \quad \pi = \pi_j^1, j \in \mathcal{K}_2 \\ y_\pi^B - 2\epsilon & : \quad \pi = \pi_j^2, j \in \mathcal{K}_2 \\ y_\pi^B & : \quad \text{otherwise.} \end{cases}$$

For Claim 1 to go through we need to replace the expression in (10) by

$$\epsilon = \frac{\min\{\min_{j=1,\dots,k} y_{\pi_j^2}^B, \min_{e:s_e>0} s_e\}}{2}$$

Finally, in (13), we can replace $\epsilon \sum_{j=1}^k (r(\pi_j^1) - r(\pi_j^2))$ by

$$\epsilon \sum_{j \in \mathcal{K}_1} (r(\pi_j^1) - r(\pi_j^2)) + 2\epsilon \sum_{j \in \mathcal{K}_2} (r(\pi_j^1) - r(\pi_j^2)).$$

Inequality (14) still holds because $|\mathcal{K}_1| + 2|\mathcal{K}_2| = k$.
Now we relax the second assumption, [A2]. Suppose we allow peer edges of the same weight. Let $P_1, \dots, P_q$ be a partition of $\{1, \dots, k\}$ such that

$$p_j, p_i \in P_s \text{ iff } c^{\lambda^B}(p_j) = c^{\lambda^B}(p_i)$$

and let $c^{\lambda^B}(P_j)$ be the cost of the edges from P_j. Intuitively, each part P_j collects all edges in the spanning tree T^a that are incident to nodes of S_d and that have weight $c^{\lambda^B}(P_j)$. Assume $c^{\lambda^B}(P_1) \leq \dots \leq c^{\lambda^B}(P_q)$. Instead of defining partitions π_j^1, π_j^2 for each peer edge p_j we define partitions π_j^1 and π_j^2 for each set P_j as follows

- π_j^1 is the partition of lowest rank with $P_j \subseteq E(\pi_j^1)$ and $y_{\pi_j^1}^B > 0$
- π_j^2 is the partition of highest rank with $P_j \cap E(\pi_j^2) = \emptyset$ and $y_{\pi_j^2}^B > 0$.

Notice, that we can now state something stronger than inequality (9):

$$r(\pi_j^1) - r(\pi_j^2) \geq |P_j|$$

for all $1 \leq j \leq q$ (recall that this corresponds to a sequence of contractions of 0-edges in Algorithm 3). In equation (13) we have to replace $\epsilon \sum_{j=1}^k (r(\pi_j^1) - r(\pi_j^2))$ by $\epsilon \sum_{j=1}^q (r(\pi_j^1) - r(\pi_j^2))$. Inequality (14) is still valid since

$$\begin{aligned} \epsilon \sum_{j=1}^q (r(\pi_j^1) - r(\pi_j^2)) &\geq \epsilon \sum_{j=1}^q |P_j| \\ &= \epsilon k. \end{aligned}$$

The last equality follows from $\{1, \dots, k\} = \bigcup_{j=1}^q P_j$.
We have proved that in the presence of nodes of degree bigger than $\nu B^* + \lceil \log_b n \rceil$ we are able to modify the optimum solution (λ^B, y^B) in order to obtain another feasible solution (λ, y) such that

$$z(\lambda^B, y^B) < z(\lambda, y).$$

This is a contradiction to the optimality of (λ^B, y^B). ■

5. CONCLUSIONS

5.1 Summary and remarks

In this paper we developed an improved approximation algorithm for the degree-bounded minimum spanning tree problem. For a positive constant B^* and an n-node graph, our method computes a spanning tree whose cost is at most a constant factor worse than the cost of the optimum degree-B^*-bounded minimum spanning tree. Additionally, we proved that the maximum degree of the resulting tree is $O(B^* + \log n)$. Our procedure solves a Lagrangean relaxation of the BMST integer program for slightly relaxed degree constraints ($(1 + \omega)B^*$ instead of B^*). We showed how this slack helps to prove low cost of the resulting tree. Our algorithm also makes use of a local search technique from [5; 7]. We used the local optimality and the fact that the final tree belonged to the set $\mathcal{O}^B$ of optimum trees for the Lagrangean program (LD(B)) to prove low maximum degree.
As a side note, the reader should notice that in Algorithm 2 we assumed the exact solution of (LD(B)). However, for practical purposes a reasonable approximation to the optimum Lagrangean multipliers is sufficient. To compute such an approximation, we could employ subgradient optimization techniques from [9; 10; 15].

5.2 Extensions and open questions

An interesting open question is whether our results extend to the case of Steiner trees and general Steiner networks. The central difficulty of such an extension stems from the fact that, in the Steiner case, the subproblem that arises from dualizing the degree constraints (the minimum cost Steiner tree problem) is NP-hard itself.
Another avenue for extending our work is to examine if our approach capable of handling individual node degrees? In the current version, node degrees are assumed to be uniform. Lemma 2 relies on the pseudo-optimality of tree T^a from Algorithm 2 and on results from [5; 7]. These results do not apply to non-uniform degrees. Is there an extention of the known MDST algorithms to handle individual degree bounds?
We believe that the techniques used in this paper can be generalized to apply to a broader class of multicriteria problems. A central point in the development of a more general framework is the identification of key properties of suitable optimization problems; in the BMST problem, the dualization of the degree constraints yields a tractable subproblem. Furthermore, the compact form of the objective function of this subproblem proved to be a key for the analysis.

6. REFERENCES

[1] F. Bauer and A. Varma. Degree-constrained multicasting in point-to-point networks. Technical Report UCSC-CRL-95-08, Computer Engineering Department, University of California, Santa Cruz, 1995.

[2] J. Cheriyan and R. Ravi. Approximation algorithms for network problems. Lecture Notes (http://www.gsia.cmu.edu/andrew/ravi), 1998.

[3] S. Chopra. On the spanning tree polyhedron. *Operations Research Letters*, 8:25–29, 1989.

[4] S. Deering, D. Estrin, and D. Farinacci. An architecture for wide-area multicast routing. In *Proceedings of SIGCOMM*, 1994.

[5] T. Fischer. Optimizing the degree of minimum weight spanning trees. Technical Report TR 93-1338, Dept. of Computer Science, Cornell University, Ithaca, NY 14853, 1993.

[6] S. Floyd, V. Jacobson, S. MacCanne, and L. Zhang. A reliable multicast framework for lightweight sessions and application level framing. In *Proceedings of SIGCOMM*, 1995.

[7] Martin Fürer and Balaji Raghavachari. Approximating the minimum-degree Steiner tree to within one of optimal. *Journal of Algorithms*, 17(3):409–423, November 1994.

[8] M. R. Garey and D. S. Johnson. *Computers and Intractability: A guide to the theory of NP-completeness.* W. H. Freeman and Company, San Francisco, 1979.

[9] M. Held and R. M. Karp. The traveling-salesman problem and minimum spanning trees: part II. *Math. Programming*, 1:6–25, 1971.

[10] Michael Held, Philip Wolfe, and Harlan P. Crowder. Validation of subgradient optimization. *Mathematical Programming*, 6(1):62–88, 1974.

[11] D. Hochbaum, editor. *Approximation Algorithms for NP-hard Problems.* PWS Publishing Company, 1997.

[12] Samir Khuller, Balaji Raghavachari, and Neal Young. Low-degree spanning trees of small weight. *SIAM Journal on Computing*, 25(2):355–368, April 1996.

[13] P.N. Klein and R. Ravi. A nearly best-possible approximation for node-weighted steiner trees. *J. Algorithms*, 19:104–115, 1995.

[14] G.L. Nemhauser and L.A. Wolsey. *Integer and Combinatorial Optimization.* Wiley, 1988.

[15] B.T. Polyak. A general method of solving extremum problems. *Doklady Akademmi Nauk SSSR*, 174(1):33–36, 1967.

[16] R. Ravi, M.V. Marathe, S.S.Ravi, D.J. Rosenkrantz, and H.B. Hunt. Many birds with one stone: Multi-objective approximation algorithms. In *Proceedings, ACM Symposium on Theory of Computing*, pages 438–447, 1993.

[17] A.S. Tanenbaum. *Computer Networks.* Prentice Hall, 1996.

[18] W. De Zhong. A copy network with shared buffers for large-scale multicast atm switching. *IEEE/ACM Transactions on Networking*, 1(2):157–165, 1993.