PRIMAL-DUAL MEETS LOCAL SEARCH: APPROXIMATING MST'S WITH NONUNIFORM DEGREE BOUNDS

J. KÖNEMANN* AND R. RAVI *

Abstract. We present a new bicriteria approximation algorithm for the degree-bounded minimum-cost spanning tree problem: Given an undirected graph with nonnegative edge weights and a degree bound B find a spanning tree of maximum node-degree B and minimum total edge-cost. Our algorithm outputs a tree of maximum degree at most a constant times B and total edge-cost at most a constant times that of a minimum-cost degree-B-bounded spanning tree.

While our new algorithm is based on ideas from Lagrangean relaxation as is our previous work [8], it does not rely on computing a solution to a linear program. Instead it uses a repeated application of Kruskal's MST algorithm interleaved with a combinatorial update of approximate Lagrangean node-multipliers maintained by the algorithm. These updates cause subsequent repetitions of the spanning tree algorithm to run for longer and longer times (hence to "age" more and more), leading to overall progress and a proof of the performance guarantee. A second useful feature of our algorithm is that it can handle non-uniform degree bounds on the nodes: Given distinct bounds B_v for every node $v \in V$, the output tree has degree at most $O(B_v + \log |V|)$ for every $v \in V$. As before, the cost of the tree is at most a constant times that of a minimum-cost tree obeying the degree bounds.

Key words. approximation algorithms, network algorithms, bicriteria approximation, spanning trees, degree-bounded spanning trees, Lagrangean relaxation

AMS subject classifications. 68W25, 05C05, 05C85, 68R10, 90C29

1. Introduction.

1.1. Formulation. In this paper, we address a natural non-uniform budget version of the *degree-bounded minimum-cost spanning tree problem* (nBMST). Given an undirected graph $G = (V, E)$, a cost function $c : E \to \mathbb{R}^+$ and positive integers $\{B_v\}_{v \in V}$ all greater than 1, the goal is to find a spanning tree T of minimum total cost such that for all vertices $v \in V$ the degree of v in T is at most B_v.

1.2. Previous work and our result. In an n-node graph, Ravi et al. [10, 11] showed how to compute a spanning tree T of degree $O(B_v \log n)$ for all $v \in V$ and total cost at most $O(\log n)\,\texttt{opt}$ where `opt` is the minimum cost of any tree in which the degree of node v is bounded by B_v for all $v \in V$. The authors generalized their ideas to Steiner trees, generalized Steiner forests and the node-weighted case. Subsequently, Könemann and Ravi [7, 8] improved the results on the (edge-weighted) spanning tree version of the problem with uniform degree-bounds (i.e., $B_v = B$ for all v). The main theorem from [8] is as follows:

THEOREM 1.1 (see [8]). *There is an 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 $\omega > 1$, computes a spanning tree T such that*

1. $\Delta(T) \leq \frac{\omega}{\omega - 1} \cdot bB + \log_b n$ *for any arbitrary constant* $b > 1$, *and*
2. $c(T) < \omega \cdot \texttt{opt}$

where $b > 1$ is an arbitrary constant.

The contributions of this paper are two-fold: First, we present improved approximation algorithms for the minimum-cost degree-bounded spanning tree problem in the presence of non-uniform degree bounds. Second, our algorithm is *direct* in the

* GSIA, Carnegie Mellon University, Pittsburgh PA 15213. E-mail:{jochen,ravi}@cmu.edu. This material is based upon work supported by the National Science Foundation under Grant No. 0105548.

sense that we do not solve linear programs. The algorithm in [8] uses Lagrangean relaxation and thus needs to solve a linear program. The analysis in [8] relies crucially on the fact that we compute an exact solution to this LP.

On the other hand, our new algorithm integrates elements from the primal-dual method for approximation algorithms for network design problems with local search methods for minimum-degree network problems [4, 12]. The algorithm goes through a series of spanning trees and improves the maximum deviation of any vertex degree from its respective degree bound continuously. A practical consequence of this is that we can terminate the algorithm at any point in time and still obtain a spanning tree of the input graph (whose node-degrees, of course, may not meet the worst-case guarantees we prove).

THEOREM 1.2. *There is a primal-dual approximation algorithm that, given a graph $G = (V, E)$, a nonnegative cost function $c : E \to \mathbb{R}^+$, integers $B_v > 1$ for all $v \in V$ and a parameter $\omega > 1$ computes a tree T such that*

1. $\deg_T(v) \leq \frac{\omega}{\omega - 1} \cdot b \cdot B_v + 2 \cdot \log_b n$ *for all* $v \in V$, *and*
2. $c(T) < \omega \cdot \mathtt{opt}$

where $b > 1$ is an arbitrary constant. The running time is $O(|V|^6 \log |V|)$.

When $B = B_v$ for all $v \in V$, we can replace the additive term $2 \log_b n$ in the degree guarantee of Theorem 1.2 by $\log_b n$ matching Theorem 1.1 constructively.

The paper is organized as follows: First, we review the primal-dual interpretation of the well-known algorithm for minimum-cost spanning trees by Kruskal [9]. Subsequently, we show how to use this algorithm for the nBMST problem and present an analysis of performance guarantee and running time of our method.

2. A primal-dual algorithm to compute MST's. In this section we review Kruskal's minimum-cost spanning tree algorithm. More specifically, we discuss a primal-dual interpretation of this method that follows from [2]. We start by giving a linear programming formulation of the convex hull of incidence vectors of spanning trees.

2.1. The spanning tree polyhedron. In the following, we formulate the minimum-cost spanning tree problem as an integer program where we associate a $0,1$-variable x_e with every edge $e \in E$. In a solution x, the value of x_e is one if e is included in the spanning tree corresponding to x and 0 otherwise. Our formulation relies on a complete formulation of the convex hull of incidence vectors of spanning trees (denoted by $\mathtt{SP}_G$) given by Chopra [2].

Chopra's formulation uses the notion of a *feasible partition* of vertex set V. A feasible partition of V is a set $\pi = \{V_1, \ldots, V_k\}$ such that the V_i are pairwise disjoint subsets of V. 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 a single node. Define the *rank* $r(\pi)$ of π as the number of nodes of G_π and let Π be the set of all feasible partitions of V. Chopra then shows that

$$\mathtt{SP}_G = \{x \in \mathbb{R}^m : \sum_{e \in E(G_\pi)} x_e \geq r(\pi) - 1 \quad \forall \pi \in \Pi\}.$$

We now let $\delta(v)$ denote the set of edges $e \in E$ that are incident to node v and obtain an integer programming formulation for our problem:

$$\text{(IP)} \qquad \begin{aligned} \min \quad & \sum_{e\in E} c_e x_e \\ \text{s.t} \quad & \sum_{e\in E[G_\pi]} x_e \geq r(\pi) - 1 \quad \forall \pi \in \Pi \\ & x(\delta(v)) \leq B_v \quad \forall v \in V \qquad (2.1) \\ & x \text{ integer} \end{aligned}$$

The dual of the linear programming relaxation (LP) of (IP) is given by

$$\text{(D)} \qquad \begin{aligned} \max \quad & \sum_{\pi\in\Pi} (r(\pi) - 1) \cdot y_\pi - \sum_{v\in V} \lambda_v B_v \\ \text{s.t} \quad & \sum_{\pi : e\in E[G_\pi]} y_\pi \leq c_e + \lambda_u + \lambda_v \quad \forall e = uv \in E \qquad (2.2) \\ & y, \lambda \geq 0 \end{aligned}$$

We also let (IP-SP) denote (IP) without constraints of type (2.1). Let the LP relaxation be denoted by (LP-SP) and let its dual be (D-SP).

2.2. A primal-dual interpretation of Kruskal's MST algorithm. Kruskal's algorithm can be viewed as a continuous process over *time*: We start with an empty tree at time 0 and add edges as we go along. The algorithm terminates at time t^* with a spanning tree of the input graph G. In this section we show that Kruskal's method can be interpreted as a primal-dual algorithm (see also [6]). For that reason, at any time $0 \leq t \leq t^*$ we keep a pair (x_t, y_t), where x_t is a partial (possibly infeasible) primal solution for (LP-SP) and y_t is a feasible dual solution for (D-SP). Initially, we let $x_{e,0} = 0$ for all $e \in E$ and $y_{\pi,0} = 0$ for all $\pi \in \Pi$.

Let E_t be the forest corresponding to partial solution x_t, i.e. $E_t = \{e \in E : x_{e,t} = 1\}$. We then denote by π_t the partition induced by the connected components of $G[E_t]$. At time t, the algorithm then increases y_{π_t} until a constraint of type (2.2) for edge $e \in E \setminus E_t$ becomes tight. Assume that this happens at time $t' > t$. The dual update is

$$y_{\pi_t,t'} = t' - t.$$

We then include e into our solution, i.e. we set $x_{e,t'} = 1$. If more than one edge becomes tight at time t', we can process these events in any arbitrary order. Thus, note that we can pick any such tight edge first in our solution.

Observe that the above primal-dual algorithm is indeed Kruskal's algorithm: If the algorithm adds an edge e at time t then e is the minimum-cost edge connecting two connected components of $G[E_t]$. In the rest of this paper we use `MST` to refer to Kruskal's minimum-cost spanning tree algorithm.

The proof of the following lemma is folklore. We supply it for the sake of completeness.

LEMMA 2.1. *At time t^*, Algorithm* `MST` *finishes with a pair (x_{t^*}, y_{t^*}) of primal and dual feasible solutions to (IP-SP) and (D-SP), respectively, such that*

$$\sum_{e\in E} c_e x_{e,t^*} = \sum_{\pi\in\Pi} (r(\pi) - 1) \cdot y_{\pi,t^*}$$

Proof. Notice, that for all edges $e \in E_{t^*}$ we must have $c_e = \sum_{\pi : e \in E[G_\pi]} y_{\pi,t^*}$ and hence, we can express the cost of the final tree as follows:

$$c(E_{t^*}) = \sum_{e \in E_{t^*}} \sum_{\pi : e \in E[G_\pi]} y_{\pi,t^*} = \sum_{\pi \in \Pi} |E_{t^*} \cap E[G_\pi]| \cdot y_{\pi,t^*}.$$

By construction E_{t^*} is a tree and we must have that the set $E_{t^*} \cap E[G_\pi]$ has cardinality exactly $r(\pi) - 1$ for all $\pi \in \Pi$ with $y_{\pi,t^*} > 0$. We obtain that $\sum_{e \in E} c_e x_{e,t^*} = \sum_{\pi \in \Pi} (r(\pi) - 1) \cdot y_{\pi,t^*}$ and this finishes the proof of the lemma. □

3. Minimum-cost degree-bounded spanning trees. In this section, we propose a modification of the above algorithm for approximating degree-bounded spanning trees of low total cost (for suitably weakened degree bounds). Our algorithm goes through a sequence of spanning trees $E^0, \ldots, E^t$ and associated pairs of primal (infeasible) and dual feasible solutions x^i, (y^i, λ^i) for $0 \leq i \leq t$. The idea is to reduce the degree of nodes $v \in V$ whose degree is substantially higher than their associated bound B_v, as we proceed through this sequence, while keeping the cost of the associated primal solution (tree) bounded with respect to the corresponding dual solution.

To begin, our algorithm first computes a minimum-cost spanning tree using the Algorithm `MST`. This yields a feasible primal solution x^0 for (LP-SP) and a feasible dual solution y^0 for (D-SP). Notice that y^0 also induces a feasible solution for (D) by letting $\lambda_v^0 = 0$ for all $v \in V$ while x^0 potentially violates constraints of type (2.1).

We introduce the notion of *normalized degree* of a node v in a tree T and denote it by

$$\texttt{ndeg}_T(v) = \max\{0, \deg_T(v) - \beta \cdot B_v\} \tag{3.1}$$

where $\beta > 0$ is a constant to be specified later. Our algorithm successively computes pairs of spanning trees and associated dual solutions to (D), i.e.

$$(x^1, \{y^1, \lambda^1\}), (x^2, \{y^2, \lambda^2\}), \ldots, (x^t, \{y^t, \lambda^t\}).$$

From one such pair to the next, we try to reduce the degree of nodes of high normalized degree. Specifically, our algorithm runs as long as there is a node in the current tree with $\texttt{ndeg}(v) \geq 2 \log_b(n)$ for some constant $b > 1$.

Our algorithm keeps a cost $\widetilde{c}_e^i$ with each edge $e \in E$ and for each iteration $1 \leq i \leq t$. x^i corresponds to a minimum-cost spanning tree E^i for cost function $\widetilde{c}^i$ and y^i is the associated dual packing. Throughout the algorithm we maintain that

$$c_{uv} \leq \widetilde{c}_{uv}^i \leq c_{uv} + \lambda_u^i + \lambda_v^i \tag{3.2}$$

for all $uv \in E$. Hence, (y^i, λ^i) is a feasible solution for (D).

Let Δ^i be the maximum normalized degree of any node in the tree E^i. The central piece of our algorithm is a *recompute step* where we raise the λ values of a carefully chosen set S_d of nodes with high normalized degree. This introduces slack in many of the constraints of type (2.2).

We now increase the $\widetilde{c}$-cost of edges that are incident to nodes in S_d (while maintaining (3.2)) and rerun `MST` on G using the new edge-costs. The hope is that the increased $\widetilde{c}$-cost of edges incident to nodes of high normalized degree leads `MST` to use edges that are incident to nodes of lower normalized degree in their place. We are

able to show that the number of recompute steps is polynomial, by arguing that we make substantial progress in the normalized degree sequence of all nodes.

As mentioned, each recompute step takes a pair of primal infeasible and dual feasible solutions $(x^i, (y^i, \lambda^i))$ and computes a new pair of primal (infeasible) and dual feasible solutions $(x^{i+1}, (y^{i+1}, \lambda^{i+1}))$. In the following we use $\texttt{ndeg}^i(v)$ as a short for $\texttt{ndeg}_{E^i}(v)$. We then adapt the notation from [3, 4] and let

$$S^i_d = \{v \in V \,:\, \texttt{ndeg}^i(v) \geq d\}$$

be the set of all nodes whose normalized degree is at least d in the i^{th} solution.

Algorithm 1 The algorithm for the nBMST problem attempts to reduce the maximum normalized degree of any node in a given spanning tree.

1: Given: primal feasible solution x^0 to (LP-SP) and dual feasible solution y^0 to (D-SP)
2: $\lambda^0_v \leftarrow 0, \forall v \in V; \widetilde{c}^0_e \leftarrow c_e, \forall e \in E$
3: $i \leftarrow 0$
4: **while** $\Delta^i > 2\log_b(n)$ **do**
5: Choose d^i in $\{\Delta^i - 2\log_b(n) + 1, \ldots, \Delta^i\}$ s.t. $\sum_{v \in S^i_{d^i-1}} B_v \leq b \cdot \sum_{v \in S^i_{d^i}} B_v$
6: Choose $\epsilon^i > 0$
7: $\lambda^{i+1}_v \leftarrow \lambda^i_v + \epsilon^i$ for all $v \in S^i_{d^i-1}$ and $\lambda^{i+1}_v \leftarrow \lambda^i_v$ otherwise
8: $\widetilde{c}^{i+1}(e) \leftarrow \widetilde{c}^i(e) + \epsilon^i$ if either $e \in E^i$ and $e \cap S^i_{d^i} \neq \emptyset$ or $e \notin E^i$ and $e \cap S^i_{d^i-1} \neq \emptyset$
9: $(x^{i+1}, y^{i+1}) \leftarrow \texttt{MST}(G, \widetilde{c}^{i+1})$
10: $i \leftarrow i + 1$
11: **end while**

A detailed description of the procedure is given in Algorithm 1. In step 5 of Algorithm 1, we choose a suitable set of nodes whose λ-values we increase. A simple argument in [3] can be extended to guarantee the feasibility of the choice in Step 5 of the algorithm.

LEMMA 3.1. *There is a* $d^i \in \{\Delta^i - 2\log_b(n) + 1, \ldots, \Delta^i\}$ *such that*

$$\sum_{v \in S^i_{d^i-1}} B_v \leq b \cdot \sum_{v \in S^i_{d^i}} B_v$$

for a given constant $b > 1$.

Proof. Suppose for a contradiction that for all $d^i \in \{\Delta^i - 2\log_b(n) + 1, \ldots, \Delta^i\}$, we have

$$\sum_{v \in S^i_{d^i-1}} B_v > b \cdot \sum_{v \in S^i_{d^i}} B_v.$$

Note that since we may assume $B_v \leq (n-1)$ for all vertices, we must have $\sum_{v \in V} B_v \leq n(n-1)$. However, since $\sum_{v \in S^i_{\Delta^i}} B_v \geq 1$, we have in this case that

$$\sum_{v \in S^i_{\Delta^i - 2\log_b(n)}} B_v \geq b^{2\log_b(n)} = n^2$$

a contradiction. □

When $B_v = B$ for all $v \in V$, the $2\log_b n$ term in the above lemma can be improved to $\log n$ using the previous arguments in [3]. This in turn leads to the slight improvement of our results claimed right after the statement of Theorem 1.2.

Step 6 of Algorithm 1 hides the details of choosing an appropriate ϵ^i by which edges in the current tree that are incident to nodes of normalized degree at least d^i are lengthened. Our choice of ϵ^i and the following update of the $\widetilde{c}$ costs of the edges in G will ensure that Kruskal's algorithm computes a new tree in which at least one edge $e^i \in E^i$ that is incident to a node of $S^i_{d^i}$ is replaced by an edge $\overline{e}^i \notin E^i$ that is incident to nodes of low normalized degree.

In fact, we show that a careful choice of ϵ^i ensures that $\overline{e}^i$ is incident to nodes of normalized degree at most $d^i - 2$ while e^i is incident to at least one node of normalized degree d^i or higher. The main idea here is to increase λ_v for nodes $v \in S^i_{d^i-1}$ by ϵ^i and increase the $\widetilde{c}$-cost of non-tree edges that are incident to nodes of normalized degree at least $d^i - 1$ by ϵ^i as well. In other words, the cost of non-tree edges incident to nodes of normalized degree at least $d^i - 1$ increases by the same amount as the cost of tree edges incident to nodes of normalized degree at least d^i. This way, we enforce that the edge we swap in touches nodes of normalized degree at most $d^i - 2$. Once we accomplish this, adapting a potential function argument from [3] we can put a polynomial upper bound on the number of such iterations (see Section 3.3).

Lemma 3.1 plays a key role in the later analysis of the performance guarantee achieved by Algorithm 1. Notice that step 7 of the algorithm increases the node multipliers of all nodes in $S^i_{d^i-1}$. On the other hand, in step 8, we only increase the $\widetilde{c}$-cost of those tree edges that are incident to nodes in $S^i_{d^i}$. Roughly, Lemma 3.1 provides us with a bound on the number of tree edges that are incident to nodes of normalized degree exactly $d^i - 1$.

We now describe how to choose ϵ^i so that the above conditions are satisfied.

3.1. Choosing ϵ^i. In this section we elaborate on the choice of ϵ^i in step 6 of Algorithm 1. In step 8 of Algorithm 1, we increase $\widetilde{c}_{uv}$ by ϵ^i for all tree edges uv that are incident to nodes of degree at least d^i and for all non-tree edges that are incident to nodes of degree at least $d^i - 1$. We want to choose ϵ^i such that the subsequent update of $\widetilde{c}^i$ and the following run of `MST` yields a new tree E^{i+1} that differs from E^i by a single edge swap: $E^{i+1} = E^i \setminus \{e^i\} \cup \overline{e}^i$. Here, the edge $e^i \in E^i$ is a tree edge that is incident to a node from $S^i_{d^i}$. On the other hand we want $\overline{e}^i \in E \setminus E^i$ to be a non-tree edge that is not incident to any node from $S^i_{d^i-1}$.

As indicated in the previous section, we want E^{i+1} to be a minimum-cost spanning tree of G for cost function $\widetilde{c}^{i+1}$. In order to achieve this, e^i must be on the unique cycle in $E^i \cup \{\widetilde{e}^i\}$ and we also must have

$$\widetilde{c}^{i+1}(e^i) = \widetilde{c}^i(e^i) + \epsilon^i = \widetilde{c}^{i+1}(\overline{e}^i).$$

In other words, the update of λ_v for $v \in S^i_{d^i-1}$ creates one more beneficial swap.

We let $\mathcal{K}^i$ be the set of connected components of the forest $E^i \setminus S^i_{d^i}$, i.e. the forest that results from removing nodes of normalized degree at least d^i from E^i. We say that an edge $e = uv \in E$ is a *cross-edge* if

1. e is a non-tree edge, i.e. $e \in E \setminus E^i$,
2. $u \in K_1, v \in K_2$ for $K_1, K_2 \in \mathcal{K}^i$ and $K_1 \neq K_2$, and
3. $\{u, v\} \cap S^i_{d^i-1} = \emptyset$.

We denote the set of cross-edges in iteration i by $\mathcal{C}^i$. Observe that if $C^i = \emptyset$, then the set S_{d^i-1} provides a witness to the infeasibility of the degree bounds imposed on the nodes in this set.

It is now clear that $E^i + e$ contains a unique cycle C_e^i for each cross-edge $e \in \mathcal{C}^i$. Furthermore, there must be at least one vertex v on C_e^i that has normalized degree at least d^i.

For each cross-edge $e \in \mathcal{C}^i$, we now let

$$\epsilon_e^i = \min_{e' \in C_e^i, e' \cap S_{d^i}^i \neq \emptyset} \left(\widetilde{c}^i(e) - \widetilde{c}^i(e') \right).$$

Note that it follows from the fact that E^i is a minimum-cost spanning tree for cost function $\widetilde{c}^i$ that $\epsilon_e^i \geq 0$ for all $e \in \mathcal{C}^i$. Finally, we let $\epsilon^i = \min_{e \in \mathcal{C}^i} \epsilon_e^i$.

In the following, we let $\langle e^i, \overline{e}^i \rangle$ be the *witness pair* for ϵ^i. In other words, let $\langle f, \overline{f} \rangle$ be a pair of edges where $\overline{f} \in E \setminus E^i$ is a non-tree edge and $f \in C_{\overline{f}}^i$ is a tree edge that is incident to a node from $S_{d^i}^i$ and that lies on the unique cycle in $E^i + \overline{f}$. Then, we must have that

$$\widetilde{c}^i(f) + \epsilon^i \leq \widetilde{c}^i(\overline{f})$$

and equality holds for $f = e^i$ and $\overline{f} = \overline{e}^i$. Notice that $\overline{e}^i$ is incident to nodes of normalized degree at most $d^i - 2$ by the definition of cross-edges.

An important observation is that ϵ^i can be 0. Such a step can be viewed as a *local-improvement* step along the lines of [4]. We do not modify the dual solution but decrease the normalized degree of a node of high normalized degree.

3.2. Analysis: Performance guarantee. Assume that Algorithm 1 terminates after iteration t^*. Recall that Theorem 1.2 requires us to show

$$c(E^{t^*}) \leq \omega \cdot \texttt{opt}. \tag{3.3}$$

In this section prove

$$\sum_{e \in E^i} c_e \leq \omega \sum_{\pi \in \Pi} (r(\pi) - 1) \cdot y_\pi^i - \omega \cdot \sum_{v \in V} B_v \cdot \lambda_v^i \tag{3.4}$$

for all $1 \leq i \leq t^*$. Observe that (y^i, λ^i) is a feasible solution for (D) and that the right-hand side of (3.4) is ω times the dual objective function induced by (y^i, λ^i). Inequality (3.4) for $i = t^*$ together with weak duality implies (3.3). This line of proof extends that developed for the analysis of primal-dual algorithms for minimum-cos networks developed in [1, 5].

In order to facilitate the proof of (3.4) for all $1 \leq i \leq t^*$, we maintain the following invariant inductively for all $0 \leq i \leq t^*$:

$$\omega \cdot \sum_{v \in V} B_v \lambda_v^i \leq (\omega - 1) \cdot \sum_{\pi \in \Pi} (r(\pi) - 1) \cdot y_\pi^i. \tag{Inv}$$

Recall that we assumed $\omega > 1$ in Theorem 1.2.

The rest of this section is split into two parts: First, we show that Invariant (Inv) holds throughout the execution of Algorithm 1). The second part then uses this invariant and proves (3.4) for all $1 \leq i \leq t^*$.

3.2.1. Proof of Invariant (Inv). We prove the validity of (Inv) for all $1 \leq i \leq t^*$ by induction over i. First, notice that (Inv) holds for $i = 0$ since $\lambda_v^0 = 0$ for all $v \in V$. To see the induction step of (Inv) we use the following lemma that ultimately

yields (3.4). Recall the definition of normalized degree in (3.1) and the role of the parameter $\beta > 0$ in it.

LEMMA 3.2. *Let $b > 1$ be the constant chosen in Theorem 1.2. We must have*

$$\sum_{\pi \in \Pi} (r(\pi) - 1) \cdot y_\pi^{i+1} \geq \sum_{\pi \in \Pi} (r(\pi) - 1) \cdot y_\pi^i + \frac{\epsilon^i \beta}{b} \cdot \sum_{v \in S^i_{d^i - 1}} B_v$$

for all $0 \leq i \leq t^*$.

The lemma quantifies the increase in the dual objective function value as our algorithm moves from y^i to y^{i+1}. Intuitively, the lemma shows that the increase in the dual objective function value is proportional to the total slack created by lengthening tree edges of E^i that are incident to nodes of normalized degree at least $d^i - 1$.

Before presenting the proof of Lemma 3.2 we show how it helps to prove (Inv). Observe that the left hand side of (Inv) increases by $\omega \epsilon^i \sum_{v \in S^i_{d^i-1}} B_v$ as a consequence of increasing the λ values of nodes in $S^i_{d^i-1}$. Lemma 3.2 implies that the right hand side of (Inv) increases by at least

$$(\omega - 1) \cdot \frac{\beta \epsilon^i}{b} \cdot \sum_{v \in S^i_{d^i - 1}} B_v.$$

Choosing

$$\beta \geq b \cdot \frac{\omega}{\omega - 1} \tag{3.5}$$

completes the proof of Invariant (Inv).

We now prove Lemma 3.2.

Proof of Lemma 3.2: Let $E^i = \{e^i_1, \ldots, e^i_{n-1}\}$ and let t^i_j be the time at which `MST` included edge e^i_j. W.l.o.g., assume that $t^i_1 \leq \cdots \leq t^i_{n-1}$. From the description of `MST` we can rewrite

(3.6)

$$\sum_{\pi \in \Pi} (r(\pi) - 1) \cdot y^i_\pi = \sum_{j=1}^{n-1} (t^i_j - t^i_{j-1}) \cdot (n - j) = \sum_{j=1}^{n-1} t^i_j \left((n - j + 1) - (n - j)\right) = \sum_{j=1}^{n-1} t^i_j$$

where we define $t^i_0 = 0$.

Figure 3.1 illustrates the effect of iteration i. Observe that, by our choice of ϵ^i, E^i is a minimum-cost spanning tree even for cost function $\widetilde{c}^{i+1}$. However, recall that there is a way to break ties such that step 9 in iteration i correctly outputs $E^{i+1} = E^i \setminus \{e^i\} \cup \{\overline{e}^i\}$. This observation together with (3.6) enables us to quantify the change in dual in iteration i:

$$\sum_{\pi \in \Pi} (r(\pi) - 1) \cdot \left(y^{i+1}_\pi - y^i_\pi\right) = \sum_{j=1}^{n-1} \left(t^{i+1}_j - t^i_j\right) \tag{3.7}$$

In iteration i, we increase the $\widetilde{c}$-cost of all edges $e \in E^i$ that are incident to nodes of normalized degree at least d^i by ϵ^i while the $\widetilde{c}$-costs of all other tree edges remain unchanged. It is not hard to see that the time an edge becomes tight equals its $\widetilde{c}$-cost. In other words, all edges in E^i that are incident to nodes of normalized degree at least d^i become tight ϵ^i time units later. Together with (3.7) we obtain

$$\sum_{\pi \in \Pi} (r(\pi) - 1) \cdot \left(y^{i+1}_\pi - y^i_\pi\right) = \epsilon^i \cdot \left|E\left(S^i_{d^i}\right) \cap E^i\right| \tag{3.8}$$

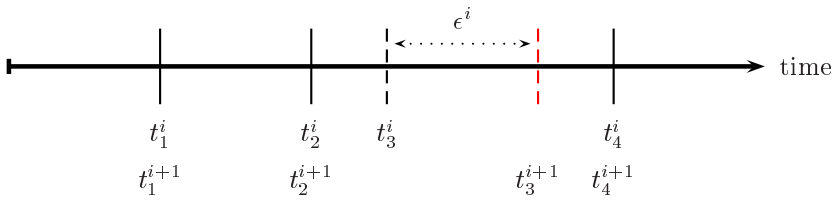

FIG. 3.1. *The figure shows two runs of* MST *in two consecutive iterations* i *and* $i+1$. *The horizontal line is the time axes while the vertical lines denote the times at which different edges become tight and are included. In the* i*th run edge* e_l *becomes tight at time* t_l^i. *In this example, edge* e_3 *is the only edge incident to a vertex of normalized degree at least* d^i. *Its length increases and the time at which it becomes tight during* MST*'s execution is postponed by* ϵ^i *time units.*

where $E\left(S^i_{d^i}\right)$ denotes the set of edges in E that are incident to nodes from $S^i_{d^i}$. (Note that we include in $E(S)$ edges with both endpoints in S).

Recall the definition of normalized degree in (3.1). Notice that it follows from the termination condition in step 4 of Algorithm 1 that $\Delta^i > 2\log_b(n)$ and hence $d^i > 0$. Therefore, the real degree of any node $v \in S^i_{d^i}$ must be at least

$$\beta \cdot B_v + d_i \geq \beta \cdot B_v + 1.$$

Finally, notice that it follows from the fact that E^i is a tree that there are at most $|S^i_{d^i}| - 1$ edges in $E\left(S^i_{d^i}\right)$ that are incident to two nodes from $S^i_{d^i}$. We can use these observations to lower-bound the right-hand side of (3.8):

$$\sum_{\pi\in\Pi}(r(\pi)-1)\cdot\left(y_\pi^{i+1}-y_\pi^i\right) \geq \epsilon^i\cdot\left(\left(\sum_{v\in S^i_{d^i}}\beta\cdot B_v+1\right)-\left(|S^i_{d^i}|-1\right)\right) \geq \epsilon^i\beta\cdot\sum_{v\in S^i_{d^i}}B_v$$

An application of Lemma 3.1 yields the lemma. □

We proceed with the proof of (3.4).

3.2.2. Bounding the cost of E^i. As MST finishes we obtain from Lemma 2.1 that

$$\widetilde{c}^{i+1}(E^{i+1}) = \sum_{e\in E^{i+1}}\widetilde{c}_e^{i+1} = \sum_{\pi\in\Pi}(r(\pi)-1)\cdot y_\pi^{i+1}. \tag{3.9}$$

Observe that the real cost of the spanning tree E^{i+1} is much smaller than $\widetilde{c}^{i+1}(E^{i+1})$. In fact, notice that we have

$$c(E^{i+1}) = c(\overline{e}^i) + c(E^i\setminus\{e_i\}) \leq \widetilde{c}^{i+1}(\overline{e}^i) + \widetilde{c}^i(E^i\setminus\{e^i\}) \tag{3.10}$$

where the first step follows from the fact that $E^{i+1} = E^i\setminus\{e^i\}\cup\{\overline{e}^i\}$ by the definition of ϵ^i and the way we break ties in MST. The second inequality uses the fact that we always have $c_e \leq \widetilde{c}_e^i$ for all $1 \leq i \leq t$ and for all $e \in E$.

Also, observe that

$$\widetilde{c}^{i+1}(E^i\setminus\{e^i\}) = \widetilde{c}^i(E^i\setminus\{e^i\}) + \left(\left|E\left(S^i_{d^i}\right)\cap E^i\right|-1\right)\cdot\epsilon^i \tag{3.11}$$

since exactly one edge from E^i that is incident to a node of normalized degree at least d^i is swapped out.

We can lower-bound $\left|E\left(S^i_{d^i}\right) \cap E^i\right|$ using the arguments from the proof of Lemma 3.2:

$$\left|E\left(S^i_{d^i}\right) \cap E^i\right| \geq \left(\sum_{v \in S^i_{d^i}} \beta \cdot B_v + 1\right) - (|S^i_{d^i}| - 1) \geq \frac{\beta}{b} \cdot \left(\sum_{v \in S^i_{d^i-1}} B_v\right) + 1$$

where the last inequality uses Lemma 3.1. Together with (3.11) we obtain

$$\widetilde{c}^{i+1}(E^i \setminus \{e^i\}) \geq \widetilde{c}^i(E^i \setminus \{e^i\}) + \frac{\epsilon^i \beta}{b} \cdot \sum_{v \in S^i_{d^i-1}} B_v. \tag{3.12}$$

We then have

$$\begin{aligned} c(E^{i+1}) &\leq \widetilde{c}^i(E^i \setminus \{e^i\}) + \widetilde{c}^{i+1}(\overline{e}^i) \\ &\leq \widetilde{c}^{i+1}(E^i) - \frac{\epsilon^i \beta}{b} \cdot \sum_{v \in S^i_{d^i-1}} B_v \\ &= \sum_{\pi \in \Pi} (r(\pi) - 1) \cdot y^{i+1}_\pi - \frac{\epsilon^i \beta}{b} \cdot \sum_{v \in S^i_{d^i-1}} B_v \end{aligned}$$

where the first inequality follows from (3.10), the second inequality follows from (3.12), and the last equality follows from (3.9).

Adding (Inv) for iteration $(i+1)$ to the last inequality we get

$$\begin{aligned} c(E^{i+1}) &\leq \omega \cdot \left(\sum_{\pi \in \Pi} (r(\pi) - 1) \cdot y^{i+1}_\pi - \sum_{v \in V} B_v \lambda^{i+1}_v\right) - \frac{\epsilon^i \beta}{b} \cdot \sum_{v \in S^i_{d^i-1}} B_v \\ &\leq \omega \cdot \left(\sum_{\pi \in \Pi} (r(\pi) - 1) \cdot y^{i+1}_\pi - \sum_{v \in V} B_v \lambda^{i+1}_v\right). \end{aligned}$$

This finishes the proof of the performance guarantee claimed in Theorem 1.2.

3.3. Analysis: Running time. In this section, we show that Algorithm 1 terminates in polynomial time. We accomplish this by showing that there will be only a polynomial number of iterations of the main loop in Algorithm 1.

LEMMA 3.3. *Algorithm 1 terminates after $O(n^4)$ iterations.*

Proof. Following [3], we define the potential of spanning tree E^i as

$$\Phi_i = \sum_{v \in V} 3^{\mathtt{ndeg}_i(v)}$$

where $\mathtt{ndeg}_i(v)$ denotes again the normalized degree of node v in the tree E^i.

Notice that an iteration of Algorithm 1 swaps out a single edge e^i that is incident to at least one node of normalized degree at least d^i. On the other hand we swap in one edge $\overline{e}^i$ that is incident to two nodes of normalized degree at most $d^i - 2$. The reduction in the potential hence is at least

$$(3^{d^i} + 2 \cdot 3^{d^i-2}) - 3 \cdot 3^{d^i-1} \geq 2 \cdot 3^{d^i-2}$$

Using the range of d^i, we can lower-bound the right hand side of the last inequality by

$$2 \cdot 3^{\Delta^i - 2\log_b(n) - 2} = \Omega\left(\frac{3^{\Delta^i}}{n^2}\right).$$

The initial potential Φ_0 is at most $n \cdot 3^{\Delta^0}$ and the decrease in the potential Φ_i in iteration i is at least $\Omega\left(\frac{\Phi_i}{n^3}\right)$.

In other words, in $O(n^3)$ iterations, we reduce Φ by a constant factor. Hence, the algorithm runs for $O(n^4)$ iterations total. Observing that each iteration can be implemented in time $O(n^2 \log n)$ [5], we see that the whole algorithm runs in time $O(n^6 \log n)$. □

Acknowledgments R. Ravi thanks IBM Almaden for hosting a stay when writing this paper.

REFERENCES

[1] A. Agrawal and P. Klein and R. Ravi. When trees collide: An approximation algorithm for the generalized Steiner problem in networks. *SIAM J. Comput.*, 24:440–456, 1995.

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

[3] M. Fürer and B. Raghavachari. An NC approximation algorithm for the minimum degree spanning tree problem. In *Proc. of the 28th Annual Allerton Conf. on Communication, Control and Computing*, pages 274–281, 1990.

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

[5] M. X. Goemans and D. P. Williamson. A general approximation technique for constrained forest problems. *SIAM J. Comput.*, 24:296–317, 1995.

[6] M. X. Goemans and D. P. Williamson. The primal-dual method for approximation algorithms and its application to network design problems. In Dorit S. Hochbaum, editor, *Approximation Algorithms for NP-hard Problems*, chapter 4. PWS, Boston, 1997.

[7] J. Könemann and R. Ravi. A matter of degree: Improved approximation algorithms for degree-bounded minimum spanning trees. In *Proceedings, ACM Symposium on Theory of Computing*, pages 537–546, 2000.

[8] J. Könemann and R. Ravi. A matter of degree: Improved approximation algorithms for degree-bounded minimum spanning trees. *SIAM J. Comput.*, 31(6):1783–1793, 2002.

[9] J. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings, American Mathematical Society*, 7:48–50, 1956.

[10] 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.

[11] R. Ravi, M. V. Marathe, S. S. Ravi, D. J. Rosenkrantz, and H. B. Hunt. Approximation algorithms for degree-constrained minimum-cost network-design problems. *Algorithmica*, 31, 2001.

[12] R. Ravi and B. Raghavachari and P. Klein. Approximation Through Local Optimality: Designing Networks with Small Degree. In *Proceedings, Foundations of Software Technology and Theoretical Computer Science*, pages 279–290, 1992.