Delegate and Conquer: An LP-based approximation algorithm for Minimum Degree MSTs*

R. Ravi and Mohit Singh

Tepper School of Business, Carnegie Mellon University, Pittsburgh PA 15213
{ravi,mohits}@andrew.cmu.edu

Abstract. In this paper, we study the minimum degree minimum spanning tree problem: Given a graph $G = (V, E)$ and a non-negative cost function c on the edges, the objective is to find a minimum cost spanning tree T under the cost function c such that the maximum degree of any node in T is minimized.
We obtain an algorithm which returns an MST of maximum degree at most $\Delta^* + k$ where Δ^* is the minimum maximum degree of any MST and k is the distinct number of costs in any MST of G. We use a lower bound given by a linear programming relaxation to the problem and strengthen known graph-theoretic results on minimum degree subgraphs [3, 5] to prove our result. Previous results for the problem [1, 4] used a combinatorial lower bound which is weaker than the LP bound we use.

1 Introduction

The minimum spanning tree problem is a fundamental problem in combinatorial optimization. It also has various applications, especially in network design. A favorable property of a connecting network is not only to have the lowest possible cost but also to have small load on all nodes. A natural way to formulate this problem is via the minimum degree minimum spanning tree (MDMST) problem. In an instance of the MDMST problem, we are given a graph $G = (V, E)$ and a non-negative cost function c on the edges, and the objective is to find a minimum cost spanning tree T under the cost function c such that the maximum degree of T is minimized. Here, the maximum degree of T is the maximum degree among all vertices in T.

The MDMST problem is closely related to the Hamiltonian path problem. If the maximum degree of an MST in an unweighted graph is at most 2, we get a Hamiltonian path. Since we do not assume that the costs are metric, no approximation is possible unless we relax the degree constraints [6]. Hence, for the MDMST problem, the natural criterion for approximation is the maximum degree of the minimum spanning tree.

* Tepper School of Business, Carnegie Mellon University. Supported by NSF ITR grant CCR-0122581 (The ALADDIN project) and NSF grant CCF-043075. Email: {ravi,mohits}@andrew.cmu.edu

1.1 Previous Work

For the MDMST problem, Fischer [4] gave a polynomial time algorithm which returns a minimum spanning tree with maximum degree $b\Delta^* + \log_b n$ for any $b > 1$ where Δ^* is the maximum degree of the optimal MST based on the techniques on Furer and Raghavachari [5]. A generalization of the MDMST problem is the bounded degree minimum spanning tree problem (BDMST) in which one is given degree bounds (B_v for vertex v) in an undirected graph with edge costs c and we demand a minimum cost tree satisfying the degree bounds. The BDMST problem is closely related to the well-studied Travelling Salesman Problem [8]. In particular, if we set $B_v = 2$ for each vertex v, the BDMST problem reduces to the Travelling Salesman Path Problem which has been studied by Lam and Newman [11].

For the BDMST problem, Konemann and Ravi [9, 10] gave bi-criteria approximation algorithms which return a spanning tree with $O(B_v + \log n)$ bound on the degree of vertex v and cost $O(c_{opt})$. Here n is the number of vertices in the input graph and c_{opt} is the minimum cost of a spanning tree obeying the degree bounds. Chaudhuri et al [1, 2] gave a quasi-polynomial time algorithm for the MDMST problem which returns a tree of maximum degree $O(\Delta^* + \frac{\log n}{\log \log n})$ and a polynomial time algorithm that returns a tree of maximum degree $O(\Delta^*)$. They also generalize both their algorithm for the BDMST problem giving algorithms with similar bounds on the degree as in the MDMST problem and cost $O(c_{OPT})$. All these results [9, 1, 2] for the BDMST problem are derived from results for the MDMST problem [4, 1, 2], thus motivating us to concentrate on the latter. Subsequent to our work, Goemans [7] has shown an algorithm for the BDMST problem which returns a tree of optimal cost and degree of vertex v at most $B_v + 2$ for each $v \in V$.

An interesting restriction of the MDMST problem arises when all costs are in $\{1, \infty\}$. Then, as every spanning tree of cost 1 edges is an MST, the MDMST problem reduces to finding a spanning tree in the undirected graph induced by the cost one edges with minimum maximum degree. Fürer and Raghavachari [5] gave an algorithm which returns a tree with maximum degree within $\Delta^* + 1$, where Δ^* is the degree of the optimal tree.

1.2 Our Work and Contributions

All previous algorithms for the MDMST problem worked with a combinatorial lower bound given by a *witness set*. The major contribution in this paper is working with a *stronger* lower bound given by a natural linear programming relaxation of the problem. Also, we strengthen the existing results of Fürer and Raghavachari [5] and Ellingham and Zha [3]. This helps us prove our main theorem below. Here, the maximum degree restriction can be generalized to specify separate bounds on individual nodes.

Theorem 1. *Given an instance of the minimum degree minimum spanning tree problem on a graph $G = (V, E)$ with a cost function c on the edges and a degree*

bound B_v on vertex v for each $v \in V$, there exists a polynomial time algorithm which shows either that the degree upper bounds are infeasible for any minimum spanning tree of G or returns an MST in which the degree of each vertex v is at most $B_v + k$ where k is the number of distinct costs in any MST.

Note that our Theorem 1 strictly generalizes the result of [5] since $k = 1$ in an unweighted graph. We introduce the following new ideas to prove Theorem 1:

- We use linear programming relaxation as a check for infeasibility instead of the witness set that has been used previously [4, 1]. If the degree bounds are feasible for a *fractional MST*, we use the optimal LP solution to divide the total degree bound of a vertex v into k parts, each assigned to a set of incident edges of a particular cost.
  Our strategy is to deal with edges of distinct costs separately. We use the known results for the unweighted case of the problem given by Fürer and Raghavachari [5] and its generalizations by Ellingham and Zha [3] to prove a weaker version of theorem 1 with degree guarantees of $B_v + 2k - 1$ instead of $B_v + k$ as claimed in the theorem. This we prove in Section 3.
- We strengthen the existing results of [5] in Theorem 4, by showing that when we do not find a witness for infeasibility we can obtain a solution where the degree bound is strictly satisfied for *one chosen vertex* while still ensuring that the violation of this bound is at most one for any other vertex. Similarly, we strengthen the results of Ellingham and Zha [3] in Theorem 5 (Section 4). We believe that these improvements are interesting in their own right.
- We use the strengthened guarantees in Theorem 4 and Theorem 5 to prove Theorem 1. We do this by applying the methods of Theorem 5 on different unweighted subgraphs, each naturally defined by edges of a particular cost that are used in an MST. This application proceeds in the top-down order by considering subgraph of progressively decreasing costs. At each step, we assign vertices to cost classes in which they can exceed their degree bound by at most one. We then inductively ensure that any such vertex does not exceed its bound in any other cost class. The resulting delegate-and-conquer algorithm is presented in Section 5, along with a proof of our main result.

2 Structure of MSTs

In this section, we prove some properties of MSTs. We then show the implication of these properties on the structure of the optimal solution to the linear programming relaxation to the MDMST problem.

2.1 Forest over Forest Problem

We define a new problem which will be used later for the MDMST problem.

Given a forest F of a graph G, we call H a F-tree of G if H does not contain any edge $e = \{u, v\}$ such that both u and v are in the same component of F and $F \cup H$ is a spanning tree over each connected component of G. Note that for

any F-tree H of G, $|H|$ = (number of connected components in F) − (number of connected components of G).

In an instance of the *forest over forest problem*, we are given an *unweighted* graph $G = (V, E)$ a forest F with connected components $\mathcal{C}(F) = \{C_1, \ldots, C_k\}$ and a degree bound B_v for each vertex $v \in V$. The problem is to find a F-tree H of G such that $deg_H(v) \leq B_v$.

We also define a notion of *witness set* which forms the basis of the algorithms of Ellingham and Zha [3] and Fürer and Raghavachari [5]. Given a set $W \subset V$ and partition $\mathcal{P}$ of connected components of F, we say $(W, \mathcal{P})$ is a witness if each edge e with endpoints in different sets of $\mathcal{P}$ must have at least one endpoint in W. The following lemma is straightforward and proved in [3].

Lemma 1. *[3] If $(W, \mathcal{P})$ is a witness, then $\sum_{w \in W} deg_H(w) \geq |\mathcal{P}| - \kappa(G)$ for any F-tree H of G, where $\kappa(G)$ is the number of connected components of G.*

The following theorem was proved by Ellingham and Zha [3] for the forest over forest problem.

Theorem 2. *[3] There exists a polynomial time algorithm which given an instance of the* forest over forest problem *over a graph $G = (V, E)$ and a forest F with degree bound B_v for each vertex $v \in V$, returns a F-tree H and a witness $(W, \mathcal{P})$ such that:*

1. *If $W \neq \phi$, then the witness $(W, \mathcal{P})$ shows that $\sum_{w \in W} deg_{H'}(w) \geq (\sum_{w \in W} B_w) + 1$ for each F-tree H' of G, i.e., the degree bounds are infeasible for any F-tree of G.*
2. *If $W = \phi$, then $deg_H(v) \leq B_v + 1$ for each $v \in V$.*

The MDST problem (unweighted MDMST problem) is a special case when $F = \phi$. Then the problem reduces to finding a spanning tree of G and the guarantees of the above theorem are exactly the same as those of Fürer and Raghavachari [5].

2.2 Laminar Structure of an MST

Given a graph $G = (V, E)$ with cost function c on the edges, let the cost function c take at most k different values on the edges of the MST. Without loss of generality, we can delete all edges of G of other costs since they do not occur in *any* MST. We also assume, without loss of generality, that the range of c is $\{1, \ldots, k\}$ as the particular values do not change the structure of any MST.

Let $G^{\leq i}$ denote the graph over $V(G)$ with only those edges of $E(G)$ that cost at most i. We let $G^{\leq 0}$ denote the graph over vertex set $V(G)$ with no edges. Let G^i denote the graph with vertex set $V(G)$ and edges in $E(G)$ which cost exactly i. The following lemma is a standard result about minimum spanning trees.

Lemma 2. *T is a minimum spanning tree of a graph G iff T^i is a $G^{\leq i-1}$-tree of $G^{\leq i}$ for each i.*

Hence, we also delete all edges e of cost i or higher which have both endpoint of $G^{\leq i-1}$ without affecting any MST T of G. More importantly, Lemma 2 implies that we can independently select the edges of each cost class one at a time and solve the appropriate unweighted forest over forest problem to form an MST. The main issue is to manage the degree of any vertex across different cost classes.

2.3 LP Relaxation

We formulate the following integer program MST_{IP} for the problem considered in Theorem 4 which is a generalization of the MDMST problem.

$$opt_B = min \quad \sum_{e \in E} c_e x_e \tag{1}$$
$$s.t.\ x(\delta(v)) \leq B_v \ \forall v \in V, \tag{2}$$
$$x \in SP_G, \tag{3}$$
$$x \ \ integer. \tag{4}$$

Here SP_G is the spanning tree polyhedron, i.e., a linear description of the convex hull of all spanning trees of G. It is well known that optimization over SP_G can be achieved in polynomial time [12]. We then relax the integrality conditions to obtain MST_{LP}. If the optimum value of MST_{LP} is more than the cost of an MST of G, then clearly the problem is infeasible.

Let x^* denote an optimal basic feasible solution to MST_{LP}. The following lemma follows directly from LP duality and is implicit in [9].

Lemma 3. *[9] The optimal basic feasible solution x^* to MST_{LP} can be written as a convex combination of spanning trees, i.e, there exists spanning trees $T_0, \ldots, T_n$ and constants $\lambda_0, \ldots, \lambda_n$ such that $x^* = \sum_{i=0}^{n} \lambda_i T_i$, $\sum_{i=0}^{n} \lambda_i = 1$ and $\lambda_i > 0$ for each $0 \leq i \leq n$. Here n is the number of vertices in the graph G.*

The following corollary to Lemma 3 is straightforward.

Corollary 1. *If $c(x^*) = c_{MST}$ then each of the spanning trees $T_0, \ldots, T_n$ obtained from Lemma 3 are minimum spanning trees.*

Proof. As $x^* = \sum_{i=0}^{n} \lambda_i T_i$, we have $c_{MST} = c(x^*) = \sum_{i=0}^{n} \lambda_i c(T_i) \leq \sum_{i=0}^{n} \lambda_i c_{MST} = c_{MST} \sum_{i=0}^{n} \lambda_i = c_{MST}$. Hence, each of the inequalities $c_{MST} \leq c(T_i)$ must hold at equality. □

2.4 LP relaxation for Forest over Forest problem

Given a forest over forest problem of constructing a F-forest of graph G with degree bound B_v for each vertex $v \in V$, we formulate the following natural IP formulation for the forest over problem which we call the $IP_{FOR}(F, G)$. Observe that this a feasibility problem as a forest over forest problem is over an unweighted graph.

$$opt = min \quad 0 \tag{5}$$
$$s.t. \quad x(\delta(v)) \leq B_v \quad \forall v \in V, \tag{6}$$
$$x \in SF(G/F), \tag{7}$$
$$x \in \{0,1\}, \tag{8}$$

Here G/F denotes the graph formed when we shrink components of each component of F in to a single vertex in G and $SF(G)$ denote the natural linear formulation for the incidence vectors of all maximal spanning forests of G generalized from [12, 9] (This is the same as the formulation for the incidence vectors of the bases of the graphic matroid of G). If we relax the integrality constraints, we get a LP relaxation which we denote by $LP_{FOR}(F, G)$. Later in Lemma 5, we show that if there exists a witness showing infeasibility of the degree bounds then the above LP relaxation is also infeasible.

2.5 Decomposing MSTs into Forests over Forests

To obtain any minimum spanning tree on an edge-weighted graph G, we need to solve the $LP_{FOR}(G^{\leq i}, G^{\leq i+1})$ for each $i = 0, \ldots, k-1$ where k is the number of distinct edge-costs in any minimum spanning tree of G. Now, we show that MST_{LP} actually solves each of these forest over forest problems with appropriate degree bounds.

Let $B_v^i = \sum_{e \in \delta(v), c(e)=i} x_e^*$. Observe that $\sum_{i=1}^{k} B_v^i \leq B_v$. Let $y_e^i = x_e^*$ if $c(e) = i$ else $y_e^i = 0$ for each $i = 1, \ldots, k$ and $e \in E$. Then we have the following lemma.

Lemma 4. *For each $1 \leq i \leq k$, y^i is a feasible solution to the linear programming relaxation of the forest over forest problem of finding a $G^{\leq i-1}$-tree of $G^{\leq i}$ with degree bound B_v^i for each vertex $v \in V$.*

Proof. Let $x^* = \sum_{j=0}^{n} \lambda_j T_j$ as in Lemma 3. Let H_j^i be the forest formed by cost-i edges in tree T_j. Clearly, each of the forests H_j^i is a valid $G^{\leq i-1}$-tree of $G^{\leq i}$ but may violate the degree bounds. By definition, $y^i = \sum_{j=0}^{n} \lambda_j H_j^i$ and hence is a valid fractional solution to $LP_{FOR}(G^{\leq i-1}, G^{\leq i})$. Also, it satisfies the degree constraints by definition as $\sum_{e \in \delta(v)} y^i(e) = \sum_{e \in \delta(v), c(e)=i} x_e = B_v^i$. □

The following lemma shows that the LP gives a stronger notion of infeasibility than any witness.

Lemma 5. *If there exists a witness $(W, \mathcal{P})$ showing that the degree bounds are infeasible then the $LP_{FOR}(F, G)$ is infeasible.*

Proof. If $(W, \mathcal{P})$ is a witness showing that the degree bounds are infeasible then for any F-tree H, $\sum_{v \in W} deg_H(v) \geq \sum_{v \in W} B_v + 1$. Hence, the above holds for F-trees $H_0, \ldots, H_n$. For any convex combination, we get

$$\sum_{v \in W} \sum_{i=0}^{n} \alpha_i deg_{H^i}(v) \geq \sum_{i=0}^{n} \alpha_i (\sum_{w \in W} B_w + 1) \geq \sum_{v \in W} (\sum_{i=0}^{n} \alpha_i B_v) + 1 = \sum_{v \in W} B_v + 1$$

since $\sum_{i=0}^{n} \alpha_i = 1$. Since any feasible solution to $LP_{FOR}(F, G)$ dominates a convex combination of F-trees (variant of Lemma 3), the degree constraint for at least one node in W must be violated. □

3 Weaker Algorithm

As a warm-up, we describe an algorithm which uses the linear programming relaxation for the MDMST problem and the algorithm of Ellingham and Zha [3] as stated in Theorem 2 to obtain a weaker guarantee than claimed in Theorem 1.

Given an instance of MDMST problem over $G = (V, E)$, cost function c and degree bound B_v on vertex v, the algorithm *Alg-Weak* is as follows:

1. Find x^* the optimum solution to the linear programming relaxation to the problem. If $c(x^*) > c_{MST}$, declare the problem infeasible.
2. Define $B_v^i = \sum_{e \in \delta(v), c(e)=i} x_e^*$. For each i, we construct $G^{\leq i-1}$-tree H^i of $G^{\leq i}$ with degree bounds $\lceil B_v^i \rceil$ for each vertex $v \in V$ using the algorithm described in Theorem 2.
3. Return the MST $T = \cup_i H^i$.

Theorem 3. *Algorithm Alg-Weak for the MDMST problem on graph G with a degree bound B_v on vertex v for each $v \in V$ and a cost function c returns an MST such that the degree of any vertex v is at most $B_v + 2k - 1$ or shows that the degree bounds are infeasible for any MST of G.. Here, k is the number of distinct edge costs in any MST.*

Proof. The algorithm declares the degree bounds infeasible only if $c(x^*) > c_{MST}$. Clearly, then the degree bounds are infeasible for any MST. We only need to argue that if the $c(x^*) = c_{MST}$, then the tree returned satisfies the claimed degree bounds and is an MST.

First, observe the tree T returned is an MST as it is a union of $G^{\leq i-1}$-tree H^i of $G^{\leq i}$ for each i (see Lemma 2).

We only need to show that the algorithm in Theorem 2 returns a F-tree and not a witness set showing infeasibility of the degree bounds. However, this directly follows from Lemma 5.

Observe that the degree of any vertex v in tree T is exactly $deg_T(v) = \sum_{i=1}^{k} deg_{H^i}(v) \leq \sum_{i=1}^{k} (\lceil B_v^i \rceil + 1) < \sum_{i=1}^{k} (B_v^i + 2) \leq B_v + 2k$. Here the last inequality follows from the fact that $\sum_{i=1}^{k} B_v^i \leq B_v$. which proves the degree bound as claimed. This proves Theorem 3. □

4 A refined characterization of witnesses

In this section, we strengthen Theorem 2 which will help us obtain improved guarantees for the MDMST problem. However, to illustrate the strengthening without getting mired in notation, we first state and prove the strengthening of the result of Fürer and Raghavachari [5] for spanning trees rather than for forests

over forests. Recall however that Theorem 2 is a generalization of the result of Fürer and Raghavachari [5], so the ideas in this strengthening generalize with some extra work allowing us to prove Theorem 5 in the the spirit of Theorem 4 (that is described in extended version of this paper [13]).

4.1 Improving Unweighted Minimum Degree Spanning Trees

Fürer and Raghavachari [5] present an algorithm which returns a tree of maximum degree $\Delta^* + 1$ where Δ^* is the minimum maximum degree of any tree.

We prove a stronger version of their theorem which is useful for the weighted version of the problem. The algorithm is similar to the algorithm of Fürer and Raghavachari [5] but our stopping criterion is more stringent.

Given a tree T and an edge $f \notin T$, let $Cycle(T, f)$ denote the set of vertices on the unique cycle in $T \cup f$.

Theorem 4. *Given a connected graph $G = (V, E)$, degree bound B_v for each vertex $v \in V$, there is a polynomial time algorithm which returns a spanning tree T and witness set $W \subset V$ (possibly empty) such that*

1. Infeasibility: *If $W \neq \phi$, then for any tree T', $\sum_{w \in W} \deg_{T'}(w) \geq \sum_{w \in W} B_w + 1$, i.e., the degree bounds are infeasible for any spanning tree of G.*
2. Solution: *If $W = \phi$, then for each node $v \in V$, $deg_T(v) \leq B_v + 1$.*
3. Strong Solution: *If $W = \phi$, then for each node in $v \in V$, there exists a tree T_v such that $deg_{T_v}(v) \leq B_v$ and for each $u \in V \setminus \{v\}$, $deg_{T_v}(u) \leq B_u + 1$.*

While the algorithm of Fürer and Raghavachari [5] results in a tree satisfying conditions 1 and 2 only, we continue to improve the solution until we satisfy condition 3 or find a new witness for infeasibility.

Algorithm Alg-Unweighted

1. Find any spanning tree T.
2. Initialize $Ugly(T) = \{v | deg_T(v) \geq B_v + 2\}$, $Bad(T) = \{v | deg_T(v) = B_v + 1\}$, $Good(T) = \{v | deg_T(v) \leq B_v\}$, $MakeGood(u) = (u)$ for each $u \in Good(T)$. Return (T, ϕ) if $Bad(T) \cup Ugly(T) = \phi$.
3. If there exists edges $e = (u_1, u_2) \in T$ and $f = (v_1, v_2) \in E \backslash T$, such that e and f are swappable (i.e., e lies in the cycle closed by f in T), $v_1, v_2 \in Good(T)$ and either u_1 or $u_2 \notin Good(T)$, then do for each $w \in Cycle(T, f) \cap (Ugly(T) \cup Bad(T))$:
   (a) $Good(T) \leftarrow Good(T) \cup \{w\}$
   (b) $Ugly(T) \leftarrow Ugly(T) \setminus \{w\}$, $Bad(T) \leftarrow Bad(T) \setminus \{w\}$.
   (c) $Makegood(w) \leftarrow (v_1, v_2)$.
4. If any w is shifted from $Ugly(T)$ to $Good(T)$ in Step 3, then $T \leftarrow Improve(w, T)$ and Return to Step 2.
5. Return $(T, W = Ugly(T) \cup Bad(T))$

The procedure $Improve(w, T)$ is implemented as follows:

1. If $MakeGood(w) = w$, then return T.
2. If $MakeGood(w) = (u, v)$, let T_u and T_v be the subtree containing u and v in $T \setminus W$ where $W = Bad(T) \cup Ugly(T)$. Here, $Bad(T)$ and $Ugly(T)$ are as defined by the algorithm before w is shifted to $Good(T)$. Let $T'_u = Improve(u, T_u)$ and $T'_v = Improve(v, T_v)$. Return $T' = T \cup T'_u \cup T'_v \cup \{u, v\} \setminus (T_u \cup T_v \cup e)$ where $e \in Cycle(\{u, v\}, T)$ and is incident at w.

The procedure $Improve(w)$ ensures that the degree of one ugly vertex reduces by at least 1 while no new ugly vertices are introduced in the resulting swaps. This is ensured by the following Lemma from [5]. A vertex v is called *non-blocking* in T if $deg_T(v) \leq B_v$.

Lemma 6. *[5] Suppose that $w \in Bad$ is marked Good in iteration i, when edge (u, v) is scanned in Step 3c of the algorithm. Then w can be made non-blocking by applying improvements to the components of F_i containing u and v where F_i is the subgraph of T generated by nodes marked good in iteration i.*

Now, we prove Theorem 4.

Proof. Suppose $W \neq \phi$. Let $C_1, \ldots, C_r$ be the components formed after removing W from T. Clearly, there does not exist any edge from C_i to C_j for any i, j else we would have found it in Step 3. Also number of components is at least $r \geq \sum_{w \in W} deg_T(w) - 2(|W|-1) \geq \sum_{w \in W}(B_w+1) - 2(|W|-1)$, since $deg_T(w) \geq B_w + 1$ for each $w \in Bad(T)$ and $deg_T(w) > B_w + 1$ for each $w \in Ugly(T)$. Let $W = \{w_1, w_2, \ldots, w_p\}$. Then, $(W, \mathcal{P} = \{C_1, \ldots, C_r\{w_1\}, \ldots, \{w_p\}\})$ is a witness as there is no swap edge wrt to W. Hence by Lemma 1 since G is connected, there must be at least $r + |W| - 1$ edges incident at vertices in W in any tree T',i.e., $\sum_{w \in W} deg_{T'}(w) \geq \sum_{w \in W} B_w + |W| - 2(|W| - 1) + |W| - 1 = \sum_{w \in W} B_w + 1$. This proves (1) in the theorem.

Suppose now that $W = \phi$. Algorithm $Alg - Unweighted$ returns a spanning tree T and set $W = Ugly(T) \cup Bad(T) = \phi$. Hence every vertex has been marked *Good* implying that for any vertex $v \in V$, $deg_T(v) \leq B_v + 1$, proving (2).

Now, we prove (3). Assume that $W = \phi$. Take any $v \in V$. Hence, $v \in Good(T)$ where T is the final tree returned by the algorithm. Either $deg_T(v) \leq B_v$ in which case $T_v = T$ suffices. Else $deg_T(v) = B_v + 1$ and v was shifted from $Bad(T)$ to $Good(T)$ in step 3b. Then by Lemma 6, there exist a series of swaps which do not increase the degree of any vertex $u \in V$ above $B_u + 1$ and make v non-blocking. The tree T_v obtained after performing these swaps by invoking $Improve(v, T_v)$ suffices for proving (3). □

4.2 Forests over Forests Revisited

In this section, we obtain strengthening of the results of Ellingham and Zha [3] on the lines of the results in Section 4.1. We are given an instance of forest over forest problem to construct a F-tree of G satisfying the degree bounds B_v for each vertex $v \in V$. We first begin with a few definitions.

Let $\mathcal{C}(F)$ be set of connected components of F. We will refer these connected components as *supernodes*. For any vertex v, we will denote F_v to be the supernode containing v.

We present the following theorem in the spirit of Theorem 4.

Theorem 5. *Given a graph $G = (V, E)$, a forest F , a degree bound B_v for each vertex $v \in V$, there exists a polynomial time algorithm StrongForest which returns a F-tree H of G and witness set $(W, \mathcal{C}_W)$ where $W \subset V$ and $\mathcal{C}_W$ is a partition of $\mathcal{C}(F)$ such that*

1. Infeasibility: *If $W \neq \phi$, then $\sum_{w \in W} deg_{H'}(w) \geq \sum_{w \in W} B_w + 1 \ \forall$ F-trees H', i.e, the degree bounds are infeasible for any F-tree of G.*
2. Solution: *If $W = \phi$, then for each $v \in V$, $deg_H(v) \leq B_v + 1$ and in each supernode $F_i \in \mathcal{C}(F)$ there is at most one vertex for which the above inequality is satisfied at equality.*
3. Strong Solution: *If $W = \phi$ then for each supernode $F_i \in \mathcal{C}(F)$ there exists a F-tree H_i which satisfies the condition (2) above. Moreover, for each vertex $u \in F_i$ we have $deg_{H_i}(u) \leq B_u$.*

Ellingham and Zha [3] prove the above theorem with conditions 1 and a weaker version of condition 2 and we strengthen it proving condition 3. Due to space considerations we omit the algorithm and the proof of the theorem since they are very similar to that for trees, only more notationally tedious. They are included in the technical report [13]. Note that the above theorem strictly generalizes Theorem 4, by setting $F = \emptyset$.

Another point to note here is that the strong solution guarantee can be applied to each connected component of G, i.e., we can choose one supernode from each connected component of G when obtaining the strong solution. This follows from the fact the F-tree problem over each connected component of G is independent and can be treated separately. We use this fact critically later in the algorithm for the MDMST problem.

5 Delegating Vertices using Refined Witnesses

We now describe an algorithm, which given an MDMST problem on graph $G = (V, E)$ with cost function c and degree bounds B_v gives a better guarantee than one in Section 2. We use the algorithm *StrongForest* of Theorem 5 instead of the algorithm *forest over forest* to obtain a improved guarantee for the MDMST problem.

Algorithm Delegate-and-Conquer:

1. **Step 1: Initialization**
   Solve the LP relaxation for the MDMST problem to obtain the optimal solution x^* and if $c(x^*) > c_{MST}$, we declare the instance infeasible. Let $B_v^i = \sum_{e \in \delta(v), c(e) = i} x_e^*$ for each $v \in V$ and for each $i = 1, \ldots, k$. Observe that $\sum_{i=1}^{k} B_v^i \leq B_v$ for each $v \in V$. Observe that by Lemma 5, invoking

the algorithm *StrongForest* of Theorem 5 to construct a $G^{\leq i-1}$-tree of $G^{\leq i}$ with degree bounds $\lceil B_v^i \rceil$ will always result in an empty witness set.

2. **Step 2: Using StrongForest**
   Find a $G^{\leq k-1}$-tree H^k of $G^{\leq k}$ with degree bounds $\lceil B_v^k \rceil$ for each vertex v using the algorithm described in Theorem 5. Let $S^k = \{v | deg_{H^k}(v) = \lceil B_v^k \rceil + 1\}$. Observe that at most one vertex of any connected component of $G^{\leq k-1}$ lies in S^k. This follows from condition (2) of Theorem 5 as each connected component of $G^{\leq k-1}$ is a supernode in the forest-over-forest problem solved. Also, let $M^k = H^k$.
3. **Step 3: Delegating the vertices to cost classes**
   For $i = k-1$ down to 1, repeat
   (a) From each connected component of $G^{\leq i}$ there is at most one vertex in S^{i+1} (proved in Lemma 7). Apply algorithm *StrongForest* of Theorem 5 to each component G_j^i of graph $G^{\leq i}$. Apply condition (3) of Theorem 5 by selecting from each component G_j^i the supernode containing v where $v \in S^{i+1}$ to obtain $G^{\leq i-1}$-tree M^i of $G^{\leq i}$.
   (b) Define $S^i = \{v | deg_{\cup_{r=i}^k M^r}(v) = (\sum_{j=i}^k \lceil B_v^j \rceil) + 1\}$
   Return $T = \cup_{i=1}^k M^i$.

Theorem 6. *Given an instance of the MDMST problem over a graph $G = (V, E)$, cost function c and degree bound B_v for vertex $v \in V$, Algorithm Delegate-and-Conquer returns either an MST T such that $deg_T(v) \leq B_v + k$ or shows that the degree bounds are infeasible for any MST. Here k is the number of different costs in any MST.*

Proof. We declare the problem infeasible when $c(x^*) > c_{MST}$ in which case the problem is clearly infeasible. The Step 2 of the algorithm returns $G^{\leq k-1}$-tree H^k of graph $G^{\leq k}$ satisfying the conditions of Theorem 5. First we prove the following claim.

Lemma 7. *There is at most one vertex of S^i in each connected component of $G^{\leq i-1}$ for each $1 \leq i \leq k$.*

Proof. The proof of the claim is by induction for $i = k$ down to 1. Clearly, this is true for S^k from condition (2) of Theorem 5. Suppose it is true for S^{i+1} such that $2 \leq i+1 \leq k$. We claim that it is true for S^i. Observe that the candidate vertices for S^i are vertices in S^{i+1} or vertices which exceed their corresponding degree bound $\lceil B_v^i \rceil$ in M^i.

Take any connected component G_j^{i-1} of $G^{\leq i-1}$. If there is some vertex v in G_j^{i-1} that is in S^{i+1}, then G_j^{i-1} is chosen in Step 3(a) of the algorithm as the selected supernode. Hence, no vertex in this connected component exceeds the degree bound $\lceil B_v^i \rceil$ in M^i by condition (3) of Theorem 5. Hence, v remains the only vertex that might exceed its total degree bound in $\cup_{r=i}^k M^i$. Else, if the connected component of $G^{\leq i-1}$ is such that there is no vertex of S^{i+1} in it, then we introduce at most one vertex which exceeds the degree bound in M^i by condition (2) of Theorem 5. In either case there is at most one vertex of S^i in each component of $G^{\leq i-1}$. Hence, the property holds for each $1 \leq i \leq k$. □

Hence, we obtain $deg_T(v) = \sum_{i=1}^{k} deg_{M^i}(v) \leq (\sum_{i=1}^{k} \lceil B_v^i \rceil) + 1 < \sum_{i=1}^{k} (B_v^i + 1) + 1 = B_v + k + 1$. This implies that $deg_T(v) \leq B_v + k$ for each $v \in V$. □

Observe that in the above proof, if each B_v^i were integral, then we would have obtained that $deg_T(v) \leq B_v + 1$ as we would "save" $k - 1$ in rounding of fractional values.

References

1. Kamalika Chaudhuri, Satish Rao, Samantha Riesenfeld, and Kunal Talwar. What would Edmonds do? Augmenting Paths and Witnesses for degree-bounded MSTs. In *Proceedings of 8th. International Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, 2005.
2. Kamalika Chaudhuri, Satish Rao, Samantha Riesenfeld, and Kunal Talwar. Push Relabel and an Improved Approximation Algorithm for the Bounded-degree MST Problem. In *To Appear in ICALP*, 2006.
3. Mark Ellingham and Xiaoya Zha. Toughness, trees and walks. *J. Graph Theory*, 33:125–137, 2000.
4. T. Fischer. Optimizing the degree of minimum weight spanning trees. *Technical report, Department of Computer Science, Cornell University*, 1993.
5. Martin Furer and Balaji Raghavachari. Approximating the minimum degree spanning tree to within one from the optimal degree. In *SODA '92: Proceedings of the third annual ACM-SIAM symposium on Discrete algorithms*, pages 317–324, Philadelphia, PA, USA, 1992. Society for Industrial and Applied Mathematics.
6. Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman & Co., New York, NY, USA, 1979.
7. Michel Goemans. Personal Communication.
8. G. Gutin and A P Punnen eds. Traveling salesman problem and its variations. *Kluwer Publications*, 2002.
9. J. Könemann and R. Ravi. A matter of degree: Improved approximation algorithms for degree-bounded minimum spanning trees. In *STOC '00: Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 537–546, New York, NY, USA, 2000. ACM Press.
10. Jochen Könemann and R. Ravi. Primal-dual meets local search: Approximating MST's with nonuniform degree bounds. In *STOC '03: Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 389–395, New York, NY, USA, 2003. ACM Press.
11. Fumei Lam and Alantha Newman. Traveling salesman path perfect graphs. *Preprint*, 2006.
12. George L. Nemhauser and Laurence A. Wolsey. *Integer and combinatorial optimization.* Wiley-Interscience, New York, NY, USA, 1988.
13. R. Ravi and Mohit Singh. Delegate and Conquer: An LP-based approximation algorithm for Minimum Degree MSTs. *Technical report, Tepper School of Business, Carnegie Mellon University*, 2006.

A Algorithm StrongForest

In this section, we give the algorithm claimed in Theorem 5 and prove its correctness.

We assume in the sequel that G is connected since the same argument can be used independently for each connected component of G.

A.1 Definitions

We first begin with a few definitions.

Given a forest F of graph G and a F-tree H and degree bounds B_v for each vertex v, a vertex v is called $Ugly$ if $deg_H(v) \geq B_v + 2$, Bad if $deg_H(v) = B_v + 1$ and $Tight$ if $deg_H(v) = B_v$ and $Good$ if $deg_H(v) \leq B_v - 1$. We will also use $Ugly$ to denote the set of vertices which are called $Ugly$. Also, we let $\mathcal{C}(F)$ denote the components of F. For each component $F_i \in \mathcal{C}(F)$(we will also refer them as "supernodes"). A super node F_i is called $Clean$ if $F_i \subset Good \cup Tight$, $Dangerous$ if $|F_i \cap Bad| = 1$ and $F_i \cap Ugly = \phi$. If a supernode is not $Clean$ or $Dangerous$, then it is called a Mob. Observe that, F_i is a Mob iff $F_i \cap Ugly \neq \phi$ or $|F_i \cap Bad| \geq 2$. Loosely, the definitions of Clean, Dangerous and Mob supernodes are intended to generalize those of Tight/Good, Bad and Ugly nodes. Let $\mathcal{C}(F)$ be set of connected components of F. We will refer these connected components as *supernodes*. Given a edge $e \notin H$ for F-tree H, we let $Cycle(H, e)$ denote the set of all the vertices which have an edge of H incident on them in the cycle formed by e in $H \cup F$. For any vertex v, we will denote F_v to be the component of F containing v.s

A vertex v in a F-tree H is called *non-blocking* if either $v \in Good$ or $v \in Tight$ and $F_v \in Clean$. In the first case, v is called non-blocking of Type I and in the latter case, v is called non-blocking of Type II.

A.2 Algorithm

The algorithm $StrongForest$ is as follows:

1. Initialize with any F-tree H.
2. Let $WitC = \{C_i \in \mathcal{C}(F) | excess_H(C_i) > 0\}$
   $Horrendous = \{v | v$ is Bad or Ugly and $F_v \in Mob\}$,
   $Awful = \{v | v$ is Bad and $F_v \in Dangerous$ or $v \in Tight$ *and* $F_v \in Dangerous \cup Mob\}$.
   $Fine = V \setminus (Horrendous \cup Awful)$. $Makefine(u) = \{u\}$ for each $u \in Fine$.
   If $Horrendous \cup Awful = \phi$ then return (H, ϕ, ϕ)
3. If there exist an edge $e = (u, v) \notin E(H)$ such that $w \in Cycle(H, e) \cap Horrendous$ and $u, v \in Fine$, then let $T'_u = Improve(H, T_u, u)$, $T'_v = Improve(H, T_v, v)$, $H \leftarrow H \cup T'_u \cup T'_v \cup \{u, v\} \setminus (T_u \cup T_v \cup \{f\})$ where f is the edge incident at w in $Cycle(H, e)$. Goto Step 2. Else
4. If there exist edges $e = (u, v) \notin H$ such that $w' \in Cycle(H, e) \cap Awful$ such that both u and $v \in Fine$, do for each $w \in Cycle(H, e) \cap Awful$:
   (a) $Awful \leftarrow Awful \setminus \{w\}$, $Fine \leftarrow Fine \cup \{w\}$
   (b) $Makefine(w) = \{u, v\}$.
   (c) If $F_w \in Dangerous$ and $w \in Bad$ then for each $x \in Awful \cap F_w$ do
   //Bad Vertex takes responsibility for all tight vertices in its supernode

i. $Awful \leftarrow Awful \setminus \{x\}$, $Fine \leftarrow Fine \cup \{x\}$
ii. $Makefine(x) = \{w\}$
and $WitC \leftarrow WitC \setminus \{F_w\}$
and goto Step 2. Else

5. Return $(H, W, \mathcal{C}_W)$ where $W = Horrendous \cup Awful$ and $\mathcal{C}_W$ is the set of connected components of G formed after removing all edges of H incident at vertices of W.

Here, the procedure, $Improve(H, T, w)$ is defined as follows:

1. *Good:* If $Makefine(w) = w$, return H.
2. *Tight:* If $Makefine(w) = x$, return $Improve(H, T_x, x)$ where T_x is the subtree containing x in $H \setminus (Awful \cup Horrendous)$ where $Awful$ and $Horrendous$ are as defined by the Algorithm before w was was included to $Fine$.
3. *Bad:* If $Makefine(w) = (u, v)$, then let $T'_u = Improve(H, T_u, u)$ and $T'_v = Improve(H, T_v, v)$ where T_u, T_v are the subtrees containing u and v in $H \setminus (Awful \cup Horrendous)$ at the time u and v were shifted to $Fine$. Return $H' \leftarrow H \cup T'_u \cup T'_v \cup \{u, v\} \setminus (T_u \cup T_v \cup \{e\})$ where $e \in \delta(w) \cap Cycle(T, \{u, v\})$

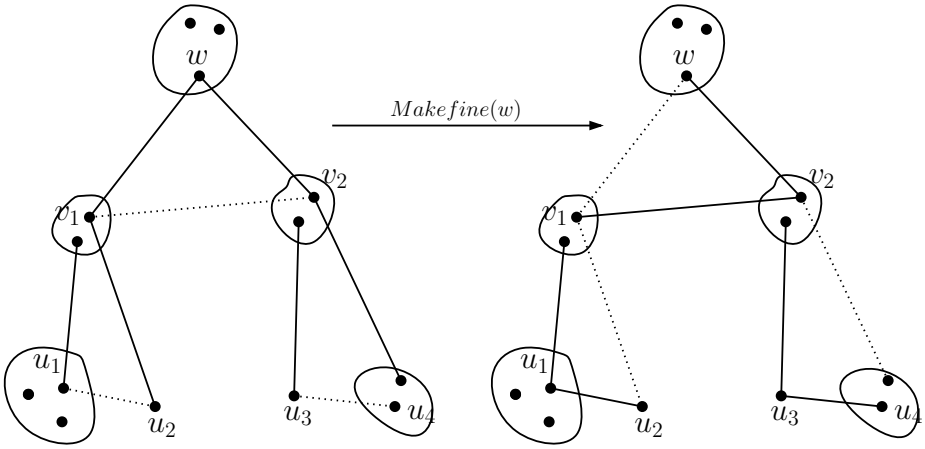

Fig. 1. In the above example, $B_v = 1$ for each $v \in V$. The algorithm returns $W = \phi$. $Improve(w)$ returns the tree such that $deg(w) = B_w$. Here, $Makefine(w) = \{v_1, v_2\}$, $Makefine(v_1) = \{u_1, u_2\}$, $Makefine(v_2) = \{u_3, u_4\}$ and $Makefine(u_i) = u_i$ for each i.

For the purpose of clarity, let us fix an iteration of the algorithm for the following definition and Lemma 8. For each vertex v that is included in $Fine$ in this iteration, let $Awful(v)$ and $Horrendous(v)$ denote the sets as defined by the algorithm exactly after v is included in $Fine$.

Before we prove Theorem 5, we prove the following lemma.

Lemma 8. *Let H be a F-tree and w be a vertex which is included in set $Fine$ by the algorithm in some iteration. Then, a call to $Improve(H, T, w)$ where T is the subtree containing v in $H \setminus (Awful(w) \cup Horrendous(w))$ returns a subtree T' such that w is non-blocking in H'. Moreover, if $w \in Bad$ then w is non-blocking of Type II. Also, no new supernodes are included in Mob.*

Proof. Structural Induction on $Makefine(w)$.

- $Makefine(w) = \{w\}$: Then $w \in Good$ in Step 2 of the algorithm and $deg_H(w) \leq B_w - 1$. Hence, w is non-blocking in H and the $Improve(H, T, w)$ return T. Clearly, H satisfies all the conditions of the Lemma.
- $Makefine(w) = \{v\}$: then $v \in Bad$ and $F_v = F_w$ is Dangerous.(This assignment takes place in Step 4c of the algorithm. Both v and w are in the same supernode with v the only vertex which exceeds its degree). Since $v \in Fine$, by induction hypothesis we have $T'_v = Improve(H, T_v, v)$ such that v is non-blocking of type II in $H' = H \cup T'_v \setminus T_v$. Hence, in H', F_v is clean. But as $F_v = F_w$, we have that w is non-blocking in H'. Also observe that T_v is a subset of T follows from the fact that v was included in $Fine$ before w in this iteration. Moreover, no new mob supernodes are introduced by induction hypothesis.
- $Makefine(w) = (u, v)$ and $w \in Tight$. Observe that $w \in Awful(u) \cup Awful(v)$. Using the fact that algorithm only decreases the sets $Awful$ in any iteration, we have that both T_u and T_v lie in different component of $H \setminus (Awful(w) \cup Horrendous(w) \cup \{w\})$. Hence, T'_u and T'_v do not share any nodes. If $H'' = H \cup T'_u \cup T'_v \setminus (T_u \cup T_v)$, then we have *both* u and v are non-blocking in H''. Then $H' = H \cup \{u, v\} \setminus e$ where e is an edge incident at w in $Cycle(H, \{u, v\})$ satisfies that $w \in Good$ and hence, non-blocking. Clearly, we did not introduce any new mob nodes in H'' by induction. Since, both u and v were non-blocking in H'', the inclusion of edge $\{u, v\}$ does not introduce any new mobs in H'.
- $Makefine(w) = (u, v)$ and $w \in Bad$. Observe that w is the only vertex from F_w which is Bad else F_w would have been a mob and we would have decreased the degree of w in Step (3). Following the same notation as in the previous case, we can argue that w is non-blocking of Type II in H'. Consider $H'' = H \cup T'_u \cup T'_v \setminus (T_u \cup T_v)$ as defined in the previous case. Clearly, $v \in Bad$ in H'' as $w \in Awful(v) \cup Awful(u)$. Since, no new mobs are introduced in H'', we have that F_w is Dangerous and w is the only vertex in F_w which exceeds its degree bound. Introduction of the edge $\{u, v\}$ and removal of e reduces the degree of w and w is a tight vertex in H'. Moreover, if any of u or v are in F_w then they must be $Good$ in H''. This implies that inclusion of edge $\{u, v\}$ in H'' does not introduce any Bad vertex in F_w. Hence, w is non-blocking of Type II in H'. The fact that no mobs are introduced in H' follows directly from induction hypothesis.

Proof of Theorem 5:

Proof of property (1): Suppose $W \neq \phi$ is returned along with forest H. Clearly, there does not exist any edge between two components of $H \setminus W$ else

it would have been found in Step 3 or 4. Root the forest H at any component C_i containing a vertex of W. Remove the edges incident at vertices of W in H. Now, we count the number of components generated after removing W from H. Traversing down the rooted tree, the number of extra components after removing vertices of W from a supernode F_i is exactly $(\sum_{v \in F_i \cap W} deg_H(w)) \geq (\sum_{v \in F_i \cap W} B_v)+1$ for the root node and at least for all other nodes $(\sum_{v \in F_i \cap W} deg_H(w))-1 \geq \sum_{v \in F_i \cap W} B_v$ for (An extra component for each edge incident at any of the nodes in $W \cap F_i$ except for one parent edge). Hence, the total number of components in $H \setminus W$ is at least $\sum_{w \in W} B_w + 2$. Hence, there must be at least $\sum_{w \in W} B_w + 1$ edges incident at any F-tree H' of G.

Proof of property (2): Suppose $W = \phi$ and forest H is returned. Then there are no Mob supernodes in H else we would have $Horrendous \neq \phi$. Hence, there can be at most one vertex in each supernode which exceeds its degree.

Proof of Property (3): Suppose $W = \phi$ and forest H is returned. Let F_i be any supernode. By (2), there is at most 1 vertex in F_i which exceeds its degree by one. If there is no such vertex then $H_i = H$ suffices. Else, let $v \in Bad \cap F_i$. Then, as $Awful = \phi$, v must be included in $Fine$ from $Awful$ in the last iteration of the algorithm. Hence, by Lemma 8, v can be made non-blocking in of Type II in H' without introducing any mobs. H'=H_i satisfies all the requirements of condition (3).

Now, we analyze the running time of the algorithm. For that we need to define an appropriate potential function which measures the progress of the algorithm.

Let $excess_H(v) = \max\{0, deg_H(v) - B_v\}$. Let excess of supernode F_i be $excess_H(F_i) = \sum_{v \in C_i} excess_H(v, \mathcal{B})$. Let normalized excess of a supernode F_i be $nexcess_H(F_i) = \max\{excess_H(F_i, \mathcal{B}) - 1, 0\}$. Let normalized excess of the F-tree H be

$$nexcess(H, \mathcal{B}) = \sum_{F_i \in \mathcal{C}(F)} nexcess_H(F_i, \mathcal{B}).$$

Normalized excess of the F-tree will be the measure of the progress of our algorithm. Clearly, $nexcess(H)$ in the beginning of the algorithm is at most n^2. In each improvement of Step 3, we reduce the normalized excess of the current F-tree by at least one. Hence, there can at most n^2 improvements in Step 3.

Between two consecutive improvements of Step 3, there can be at most n calls of Step 4, as we reduce the size of $Awful$ each time we call Step 4. Hence, there are at most n^3 calls of Step 3 or Step 4. Also, all these steps can be implemented in polynomial time proving Theorem 5.