

What Would Edmonds Do? Augmenting Paths and Witnesses for Degree-Bounded MSTs

Kamalika Chaudhuri^{1,*}, Satish Rao^{1,**},
Samantha Riesenfeld^{1,***}, and Kunal Talwar^{2,†}

¹ UC Berkeley

{kamalika,satishr,samr}@cs.berkeley.edu

² Microsoft Research, Redmond, WA

kunal@microsoft.com

Abstract. Given a graph and degree upper bounds on vertices, the BDMST problem requires us to find the minimum cost spanning tree respecting the given degree bounds. Könemann and Ravi [10, 11] give bicriteria approximation algorithms for the problem using local search techniques of Fischer [5]. For a graph with a cost C , degree B spanning tree, and parameters $b, w > 1$, their algorithm produces a tree whose cost is at most wC and whose degree is at most $\frac{w}{w-1}bB + \log_b n$. We give a polynomial-time algorithm that finds a tree of *optimal* cost and with maximum degree at most $bB + 2(b+1)\log_b n$. We also give a quasi-polynomial algorithm which produces a tree of *optimal* cost C and maximum degree bounded by $B + O(\log n / \log \log n)$. Our algorithms work when there are upper as well as lower bounds on the degrees of the vertices.

1 Introduction

We study the problem of finding a minimum cost spanning tree in a graph $G = (V, E)$ such that the tree meets degree bounds on the vertices in V . A solution to this problem, called a degree-bounded minimum spanning tree (BDMST), can be used to find minimum cost travelling salesman path (TSPP) between two endpoints by finding the minimum cost spanning tree where the endpoints have degree bounds of one and the internal nodes have degree bounds of two. Since the Hamiltonian path problem can be solved given even a polynomial factor approximation algorithm for TSPP, and a Hamiltonian path is NP-hard to approximate to within any computable factor, we expect that it is necessary to relax the degree bounds to get any reasonable approximation algorithm. (Note the implicit assumption that the edge costs of the input graph do not define a metric.)

* Research partially supported by the Berkeley Fellowship and NSF Grant CCR-0121555

** Research partially supported by NSF Grant 0013128000

*** Research partially supported by NSF Graduate Fellowship

† Part of this research was performed when the author was a student at UC Berkeley and was supported by the NSF via grants CCR-0121555 and CCR-0105533

The BDMST problem, [14] also has several applications in efficient network routing protocols. For example, in multicast networks, one wishes to build a spanning tree that minimizes the total cost of the network as well as the maximum work done by (that is, the degree of) any vertex. The BDMST problem is a natural formulation of the duelling constraints.

For the sake of simplicity, we focus on a version of the problem in which each node has the same degree bound B , though our results generalize to the case of nonuniform degree bounds. We refer to this case as the uniform version of the problem. Bicriteria approximation algorithms for this problem were first given by Ravi et al. [14, 15]. The best previous results are by Könemann and Ravi [10, 11]. They give an algorithm that, for a graph with a BDMST of maximum degree B and cost C , and for any $w > 1$, finds a spanning tree with maximum degree $\frac{w}{w-1}bB + \log_b n$, for any constant b , and cost wC . While this expression yields a variety of tradeoffs, one can observe that the Könemann-Ravi algorithm always either violates the degree constraints or exceeds the optimal cost by a factor of two.

We give a polynomial-time algorithm that removes the error in the cost completely and also a quasi-polynomial-time algorithm that both obtains a solution of optimal cost and significantly improves the error in the degree bounds. Specifically, we give a polynomial-time algorithm that, given a graph with a BDMST of optimal cost C and maximum degree B , finds a spanning tree of cost exactly C and maximum degree $bB + 2(b+1)\log_b n$ for any constant b . The main technical contribution here is the use of a subroutine that finds minimum cost spanning trees while enforcing both lower and upper bounds on degrees. This allows for more natural and better performing cost-bounding techniques, which we discuss below. We also give a quasi-polynomial-time algorithm that finds a spanning tree of maximum degree $B + O(\frac{\log n}{\log \log n})$ and optimal cost C . If the degree is $\omega(\log n / \log \log n)$, our algorithm gives a $1 + o(1)$ approximation in the degree while achieving optimal cost. The main technical contribution here is the use of augmenting paths to find low-degree minimum spanning trees, where Könemann and Ravi[10] used a local optimization procedure to solve this subproblem. We discuss this technique further below as well.

In a recent paper, Könemann and Ravi[11] improve their algorithms both in terms of implementation and applicability by providing an algorithm that can deal with nonuniform degree bounds and does not rely on explicitly solving a linear program. Our polynomial-time algorithm works as does theirs for nonuniform bounds, except we get cost optimality. Our quasi-polynomial-time algorithm extends easily to nonuniform degree bounds since our error is additive, rather than multiplicative, for this version of the algorithm. The algorithms that we give for enforcing lower-bound degree constraints require new ideas and may also be of independent interest. Our results extend to a more general version of the BDMST problem, in which upper bounds *and lower bounds* on the degree of vertices are given.

Previous Techniques

Fürer and Raghavachari[6] give a beautiful algorithm for un-weighted graphs that finds a spanning tree of degree $\Delta + 1$ in any graph that has a spanning tree

of optimal degree Δ . This is optimal, in the sense that an exact algorithm would give an algorithm for Hamiltonian path. Their algorithm takes any spanning tree and relieves the degree of a high-degree node by an alternating sequence of edge additions and deletions. While there is a node in the current tree of degree at least $\Delta + 1$, their algorithm either finds such a way to lower its degree or certifies that the maximum degree of *any* spanning tree must be at least Δ . The certificate consists of a set of nodes S whose removal leaves at least $\Delta|S|$ connected components in the graph. This implies that the average degree of the set S must be at least Δ . This algorithm is reminiscent of Edmonds' algorithm for unweighted matching [3].

For the weighted version of the BDMST problem, we consult a different algorithm of Edmonds[2] that computes *weighted* matchings. This algorithm can be viewed as first finding a solution to the dual of the matching problem: it finds an assignment of penalties to nodes that lengthens adjacent edges and then finds a maximal packing of balls around subsets of nodes of the graph. Once the dual is known, one can ignore the values of the weights on the edges and rely solely on an un-weighted matching algorithm.

Könemann and Ravi begin along Edmond's path by examining a linear programming relaxation for the BDMST problem and its dual¹. Unfortunately, in this case, the combinatorial subroutine must produce a low-degree *minimum cost* spanning tree in the graph where the edge costs have been modified by the dual potentials. To find an MST in the modified graph that has (approximately) minimum degree over all MSTs, Ravi and Könemann use an algorithm of Fischer. For a graph with an MST of degree Δ , Fischer's algorithm provides an MST of degree at most $b\Delta + \log_b n$ for any $b > 1$. They show how to get a good solution by solving a linear program with a relaxed (by a factor of $(w/w - 1)$) degree constraint and then applying Fischer's algorithm to the dual.

Our Results and Techniques

As mentioned above, Könemann and Ravi's methods introduce a tradeoff between degree and cost that always creates a constant-factor error somewhere. In this paper, we remedy this problem by following along the lines of Edmonds' approach to weighted matchings. We show that any MST in the modified graph in which the nodes with dual penalties have (relatively) high degree has low cost in the original graph. Thus, instead of just finding an MST of low maximum degree, we also wish to enforce that the nodes with penalties have high degree. This entails developing algorithms and certificates for enforcing degree lower bounds, in addition to our algorithms for enforcing degree upper bounds. Moreover, we have to combine the algorithms to simultaneously enforce the high- and low-degree constraints.

In addition to improving the cost of the solution, we improve the degree-bounds at the cost of a quasi-polynomial running time. Our approach relies on

¹ As in the case of Edmonds' non-bipartite matching algorithm, the linear program and its dual are of exponential size, though their solutions can be found in polynomial time.

our analogy with bipartite matching. As a guide to our approach, consider the problem on bipartite graphs of assigning each node on the “left” to a node on the “right” while minimizing the maximum degree of any node on the right. This problem can naturally be solved using matching algorithms. Consider instead finding a locally optimal solution where for each node on the left, all of its neighbors have degree within 1 of each other. One can then prove that no node on the right has degree higher than $b\Delta + \log_b n$ where Δ is the optimum value. The proof uses Hall’s lower bound to show that any breadth first search in the graph of matched edges is shallow, and the depth of such a tree bounds the maximum degree of this graph. This is the framework of Fischer’s algorithm for finding a minimum cost spanning tree of degree $b\Delta + \log_b n$.

In the case of the bipartite graph problem above, an augmenting or alternating path algorithm for matching gives a much better solution than the locally greedy algorithm. Based on this insight, we use augmenting paths to address Fischer’s problem of finding a MST of minimum maximum degree. By doing so, we are able to find a minimum cost tree of degree at most $\Delta + O(\frac{\log n}{\log \log n})$. In contrast to Fisher’s local (or single-swap) approach, our algorithm finds a sequence of moves to decrease the degree of one high degree node. This introduces significant complications since changing the structure of the tree also changes which swaps are legal.

We also remark that our augmenting path algorithms, though based on the most powerful methods in matching, fall short of the $\Delta + 1$ that Furer and Raghavachari obtain. While we make some progress by removing all the leading constants, and by bounding the cost using degree lower bounds, $\Delta + 1$ is a tantalizing target.

2 Bounded Degree Minimum Spanning Trees

BDMST Problem: Given a weighted graph $G = (V, E, c)$, $c : E \rightarrow R^+$, and a positive integer $B \geq 2$, find the minimum cost spanning tree in the set $\{T : \forall v \in V, \text{deg}_T(v) \leq B\}$.

Konemann and Ravi [11] show that an MST of minimum degree for a certain cost function is also a BDMST with analogous guarantees on degree and with cost within a constant factor of the optimal. We show in this section that an algorithm which obtains an MST with guarantees on both maximum and minimum degrees can be used to produce a BDMST with optimal cost and analogous degree bounds. We present an algorithm in Section 6 that obtains such guarantees.

Linear Programs for BDMST: An integer linear program for the BDMST problem is given by:

$$\begin{aligned} \text{OPT}_B = \min \quad & \sum_{e \in E} c_e x_e \\ \text{s.t.} \quad & \sum_{e \in \delta(v)} x_e \leq B \quad \forall v \in V; x \in \text{SP}_G; x_e \in \{0, 1\} \end{aligned} \tag{1}$$

where $\delta(v)$ is the set of edges of E that are incident to v , and SP_G is the convex hull of edge-incidence vectors of spanning trees of G . A tree defined by a vector $x \in \text{SP}_G$, the entries of which are not necessarily all integer, is called a *fractional* spanning tree. It can be written as a convex combination of spanning trees of G . For a fractional tree T with edge incidence vector $x \in \text{SP}_G$, let $\deg_T(v) = \sum_{e \in \delta(v)} x_e$. We can write the linear program relaxation of (1) as $\min\{c(T) : T \in \text{SP}_G, \forall v \in V \deg_T(v) \leq B\}$. The approach used by Könemann and Ravi [11] is to take the Lagrangean dual of this LP, given by: $\max_{\lambda \geq 0} \min_{T \in \text{SP}_G} \{c(T) + \sum_{v \in V} \lambda_v (\deg_T(v) - B)\}$. We can think of the optimal solution to the dual as a vector λ^B of Lagrangean multipliers on the nodes and a set \mathcal{O}^B of optimal trees, such that every tree $T^B \in \mathcal{O}^B$ minimizes $c(T^B) + \sum_{v \in V} \lambda_v^B (\deg_{T^B}(v) - B)$. The set of multipliers λ^B and \mathcal{O}^B can be computed in polynomial time. The optimal value $\text{OPT}_{LD(B)}$ of this dual program is a lower bound on OPT_B and a tight lower bound on the optimal value of the LP relaxation.

Following the analysis of [11], let us define a new cost function c^{λ^B} , where $c^{\lambda^B}(e) = c_e + \lambda_u^B + \lambda_v^B$ for an edge $e = (u, v)$. Since $B \sum_{v \in V} \lambda_v^B$ is constant for a fixed choice of λ^B , every tree in \mathcal{O}^B is an MST of G under the cost function c^{λ^B} . The optimal solution to the linear program is a *fractional MST* $T_B^f = \sum_{T \in \mathcal{O}^B} \alpha_T T$. Note that the degree of v in T_B^f is $\deg_{T_B^f}(v) = \sum_T \alpha_T \deg_T(v)$.

Let $L_B = \{v : \lambda_v^B > 0\}$ be the set of nodes with positive dual variables in the optimal solution. Complementary slackness conditions applied to the optimal solutions of the linear program and its dual imply the following claim.

Lemma 1 *For all $v \in V$, $\deg_{T_B^f}(v) \leq B$, and for all $v \in L_B$, $\deg_{T_B^f}(v) = B$.*

So we are given the existence of the fractional MST T^f (under the cost function c^λ) that meets both upper and lower degree bounds (i.e. no node has degree more than B and every node in L_B has degree exactly B). Our approach to solving the BMST problem is to find an *integral* MST, under a slightly different cost function, that meets both the upper and lower degree bounds approximately. Then we use the dual program to show that this tree does not cost too much more than T^f .

Let $B^* = B + \omega$, for some $\omega > 0$ to be specified later. Let $T^{B^*} \in \mathcal{O}^{B^*}$, so T^{B^*} is an MST under the cost function $c^{\lambda^{B^*}}$. Since λ^{B^*} is a feasible solution for the dual LP, it is clear that $c(T^{B^*}) + \sum_{v \in V} \lambda_v^{B^*} (\deg_{T^{B^*}}(v) - B) \leq \text{OPT}_{LD(B)}$. Further, if T^{B^*} has the property that for every node $v \in L_{B^*}$, $\deg_{T^{B^*}}(v)$ is at least $B = B^* - \omega$, the second term in the above expression is non-negative and hence $c(T^{B^*})$ is at most $\text{OPT}_{LD(B)}$.

Lemma 2 *Let T be an MST of G under the cost function $c^{\lambda^{B^*}}$ such that for every $v \in L_{B^*}$ $\deg_T(v) \geq B$. Then $c(T) \leq \text{OPT}_{LD(B)}$.*

MSTs with Maximum and Minimum Degree Bounds

In Section 2, we observed that the problem of finding a BDMST can be reduced to the problem of finding an actual MST, under the cost function c^λ , which satisfies the upper and lower bounds on the degrees. Our approach is to start with an

arbitrary MST $T \in \mathcal{O}^{B^*}$ and try to decrease its maximum degree. So that we can bound the cost of T with respect to the optimal BDMST, we simultaneously increase to a relatively high degree of at least B the degree of each node in $L_{B^*} = \{v \in V : \lambda_v^{B^*} > 0\}$. We first consider meeting these dueling constraints one at a time. In Section 4 we give an algorithm for finding an MST in which the maximum degree bounds are approximately met. Our algorithm produces a witness \mathcal{W} that certifies that the tree we find has near-optimal maximum degree. In Section 5, we show how to find an MST in which a specified subset L of the nodes meet minimum degree bounds approximately. Again, the algorithm produces a witness \mathcal{W}_L that certifies near-optimality. Finally, in Section 6 we give an algorithm for achieving these goals simultaneously and an analogous witness.

3 Minimum Spanning Trees with Degree Bounds

MSTDB Problem: Given a graph $G = (V, E)$ with costs on edges, a degree upper bound B_H , a set of vertices L and a degree lower bound B_L , find, if one exists, a minimum spanning tree of G such that no vertex has degree more than B_H and all vertices in L have degree at least B_L .

Note that an MST with these degree bounds may not exist in the graph. If this happens, the algorithm is expected to provide a combinatorial proof of non-existence.

We present two algorithms for this problem. The first one, which is based on Fischer’s algorithm for finding a minimum degree MST of a graph, runs in polynomial time and finds an MST such that (a) every vertex has degree at most $bB_H + \log_b n$ and (b) all vertices in L have degree at least $B_L/b - \log_b n$. If it fails, it finds a combinatorial witness to show that there exists no MST of the graph in which all vertices have degree at most B_H and the vertices in L have degree at least B_L . The second algorithm we present finds a MST with two properties: (a) every vertex has degree $B_H + O(\frac{\log n}{\log \log n})$ or less and (b) all vertices in L have degree $B_L - O(\frac{\log n}{\log \log n})$ or more. If it fails, it provides a combinatorial witness to show that there exists no MST of G in which all vertices have degree at most B_H and the vertices in L have degree at least B_L . Our second algorithm runs in quasi-polynomial time.

Recall from Section 2 that our BDMST algorithm runs MSTDB with the following inputs: the cost function is $c'_{uv} = c_{uv} + \lambda_u + \lambda_v$, the degree upper bound B_H is B , the degree lower bound B_L is B , and L , the set of vertices on which the degree lower bounds should be enforced, is the set of vertices for which $\lambda_v > 0$. Lemma 1 guarantees that there always exists a fractional MST for this cost function in which the maximum degree of any vertex is B and in which all vertices in L have degree exactly B . The combinatorial witnesses produced by the algorithm certify non-existence of fractional trees, as well as integral trees, with the same degree bounds. It follows that the output of the first algorithm is an MST of maximum degree $bB + \log_b n$, where $b > 1$ is a constant, in which the minimum degree of L is also $B/b - \log_b n$, and the output of the second algorithm

is an MST with maximum degree $B + O(\frac{\log n}{\log \log n})$ in which the minimum degree of L is $B - O(\frac{\log n}{\log \log n})$. Otherwise the witnesses produced by the algorithms provide a contradiction to Lemma 1.

Starting with an arbitrary MST, both of our MSTDB algorithms proceed in phases of improvement. Each phase is essentially a combination of two algorithms: MAXDMST and MINDMST. MAXDMST is essentially a subroutine that either decreases the number of “high degree” vertices or produces a witness to show that the maximum degree of the current tree is close to optimal. Similarly, MINDMST decreases the number of nodes in L which are of “low degree”. If it fails, it finds a witness to guarantee that the minimum degree of L in the current tree is close to optimal.

In our first MSTDB algorithm, MAXDMST is essentially Fischer’s algorithm, and MINDMST is a symmetric version of Fischer’s algorithm that finds an MST in which a certain set of nodes have maximum minimum degree. Since these algorithms are largely based on Fischer’s algorithm, we do not provide any more details on them in this paper. Instead we focus for the next two sections on our second MSTDB algorithm which uses an approach based on augmenting paths.

4 Minimum Spanning Trees with Degree Upper Bounds

Let $d_{\max}(T)$ denote the maximum degree over all vertices in the current MST T , and let $d_{\min}^L(T)$ denote the minimum degree over all vertices in L in T . In this section, we describe the MAXDMST algorithm, which solves the following problem.

Min Max Degree MST: Given a graph G and a degree bound $d > 0$, find an MST T of G such that $d_{\max}(T) \leq d$.

The main idea behind MAXDMST is to find a series of *edge swaps* that decreases in a controlled manner the degree of some high-degree vertex in the tree. We say that an edge e in tree T can be *swapped* for edge e' if the following hold: (a) $e' \notin T$, and (b) the unique cycle in $T \cup e'$ contains e . Performing the swap (e, e') involves removing e from T and adding e' to produce another tree T' . A swap (e, e') is called *feasible* if the edges e and e' have the same weight. Note that if T is an MST, then the tree T' produced after a feasible swap on T is an MST as well. MAXDMST attempts to find an augmenting path of swaps that decreases the degree of a high-degree vertex by one without creating any more vertices of the same or higher degree.

An important part of our algorithm is that when it terminates, it produces a proof that the resulting tree T has close-to-optimal degree. We call such a combinatorial proof a *witness*. The basic underlying structure of the witness produced by MAXDMST is a partition of the nodes in G into a center set W and k other sets C_1, C_2, \dots, C_k called *clusters*. The partition has the property that there are no tree edges between clusters. The use of this partition as a witness is based on the following idea: if there is no feasible swap involving an inter-cluster edge and an edge in T adjacent to a node in W , then *any* MST

must connect the clusters C_1, C_2, \dots, C_k using edges adjacent to W . If it is also true that k is large, then the average degree of W in any MST must be high. Our version of the proof of the following lemma appears in the full version of the paper.

Lemma 3 ([6]) *Let V be partitioned into sets of nodes W, C_1, C_2, \dots, C_k . If for any edge e between two clusters C_i and C_j , there is no MST on G containing e , then for any MST T of G , $d_{\max}(T) \geq \lceil \frac{|W|+k-1}{|W|} \rceil$.*

Definitions and Notation

Before describing our algorithm formally, we introduce some definitions and notation. Let T be the MST of G at the beginning of the current phase of the algorithm. We use W to refer generally to the center set, and W_i to denote the center set at some specific iteration i of this phase. For a node $u \in W$, the clusters connected to u by tree edges are called the *children clusters* of u . For any d , we denote by $S_{\geq d}$ the set of vertices that have degree d or more in T .

Let $\tau \subseteq T$ be a Steiner tree on the nodes of W_0 (the Steiner nodes are the nodes in $V - W_0$). We say that we *freeze* the edges of τ incident on W , meaning that the edges of τ that are incident on W are not allowed to be removed by any swap. Freezing these edges ensures that the edge swaps made at the end of the algorithm result in a tree (and not a graph containing a cycle).

Given T and a partition of the nodes into W and a set \mathcal{C} of clusters, we call a non-tree inter-cluster edge e' *feasible with respect to W* if there exists a feasible swap (e, e') such that e is incident on W and e is not frozen. A feasible swap $(e = (u, v), e' = (u', v'))$ is called *good* if (a) $u \in W$ (b) $v \notin W$, (c) e is not frozen, and (d) e' is an inter-cluster edge. The algorithm uses good swaps to decrease the degrees of high-degree vertices in W .

The MAXDMST Algorithm

Given G and the current MST T on G , we begin each phase of the algorithm by choosing a degree d such that $|S_{\geq d-1}| \leq \frac{\log n}{\log \log n} |S_{\geq d}|$.

Note that this inequality must be true for some d lying between $d_{\max}(T)$ and $d_{\max}(T) - \frac{\log n}{\log \log n}$. For the duration of this phase, we maintain a center set W of vertices, and a set \mathcal{C} of clusters. The initial center set W_0 is $S_{\geq d-1}$, and the initial clusters are the connected components created by deleting W_0 from T . In general, though, a cluster is not necessarily internally connected. Consider the Steiner tree $\tau \subseteq T$ on W_0 , and freeze the edges of τ incident on W_0 . In each iteration i , we find a good swap $(e = (u, v), e' = (u', v'))$ where $u \in W_i$ and $v \notin W_i$. We then take the following steps:

1. Remove u from W_i .
2. Form a new cluster C_u by merging u with the cluster containing u' , the cluster containing v' , and all the children clusters of u .
3. Consider each feasible edge (u, w) such that w is a node that has been removed from the witness at some prior stage, and merge C_u with the cluster currently containing w .

Step 3 ensures that we never try to use a feasible edge between two nodes which initially have degree $d - 1$ in T . This process is repeated until we either remove a vertex u^* with degree d or higher from W , or we run out of good swaps. Either event marks the end of the current phase.

Lemmas 4 and 5 show that in the first case, we can find a sequence of edge swaps to get a tree T' from T such that the degree of u^* decreases by one and no vertex has its degree raised to more than $d - 1$. In the case that we run out of good swaps, Lemma 6 and Theorem 9 shows that we can interpret W_i as a witness to the fact that the maximum degree of T is close to optimal.

Lemma 4 *Suppose we remove a vertex v from the center set W_i in some iteration i . Then there is a sequence of feasible edge swaps which (a) decreases the degree of v by one and (b) does not increase the degree of any other node in T to d .*

Note that since each of $((u_i, v_i), (u'_i, v'_i))$ is a feasible swap when discovered, performing any one of them does not introduce a cycle in the tree. Yet it is not obvious that we can perform all of them together without disrupting the tree structure. Lemma 5 shows that the graph produced after performing all these edge swaps is still a tree. The frozen edges of the Steiner tree play a crucial role in its proof. Proofs from Lemmas 4 and 5 are omitted from this extended abstract.

Lemma 5 *The graph produced after performing all the edge swaps in Lemma 4 is a tree.*

The MAXDMST Witness

The actual witness \mathcal{W} produced by the algorithm is slightly more complex than the one described at the beginning of Section 4. A witness \mathcal{W} actually consists of a partition of the nodes into W and a set of clusters C_1, C_2, \dots, C_k , and a set of edges R such that at least one endpoint of each edge in R is in W . The witness \mathcal{W} must have the property that any MST that contains all of the edges of R cannot contain an inter-cluster edge. Now if the size of R is small and k is large, then any MST must use a large number of edges incident to W to connect the clusters. The proof of Lemma 6 is straightforward and is omitted from this extended abstract.

Lemma 6 (Witness to high optimal degree) *A high-degree witness $\mathcal{W} = (W, R, \{C_1, C_2, \dots, C_k\})$ certifies that any MST of G has maximum degree at least $\lceil \frac{|W| + k - 2|R| - 1}{|W|} \rceil$.*

Now we interpret the structure produced by running the MAXDMST algorithm as a witness. Let R^* be the subset of frozen edges that have at least one endpoint incident on W^* , where W^* is the center set when the algorithm terminates. Let $C_1^*, C_2^*, \dots, C_k^*$ be the clusters and T^* the tree when the algorithm terminates.

Lemma 7 $\mathcal{W}^* = (W^*, R^*, \{C_1^*, C_2^*, \dots, C_k^*\})$ *is a high-degree witness.*

Lemma 8 *At most $2|W_0|$ edges are frozen by the algorithm, where W_0 is our initial witness.*

Theorem 9 *If the MAXDMST algorithm halts with an MST of maximum degree d , then $\Delta_{\text{OPT}} \geq d - 2 - O(\frac{\log n}{\log \log n})$.*

The proofs of Lemma 8, Lemma 7 and Theorem 9 are omitted from this extended abstract.

5 Minimum Spanning Trees with Degree Lower Bounds

Max Min Degree MST: Given a subset L of the nodes of G and a minimum degree bound $d > 0$, find an MST T of G such that $d_{\min}^L(T) \geq d$.

In this section, we consider the problem defined above. Note that this is an NP-hard decision problem, and our algorithm solves a gap version of it. Much like the algorithm of the previous section, our algorithm either finds a tree which respects the degree bounds up to an additive error of $\frac{\log n}{\log \log n}$ or gives a combinatorial proof showing that the decision problem is a no instance, i.e. there is no MST in G satisfying the degree constraints. While the algorithm and the witness here are somewhat analogous to those in the previous section, they are not symmetric, and new ideas are required for this case.

We first introduce the kind of witness we use for proving upper bounds on the minimum degree in L . A witness \mathcal{W}_L consists of a set W of nodes in L , a set of clusters C_1, C_2, \dots, C_k, D , that form a partition of $V \setminus W$, and a set R of inter-cluster edges. In any MST T of G , for all i , $1 \leq i \leq k$ the cluster C_i must be internally connected. For any MST T of G containing the edges of R , it must also be true that for all $v \in D$, there is some i , $1 \leq i \leq k$ such that there is a path from v to C_i in the forest created by deleting W from T .

Lemma 10 (Witness to low optimal degree) *A witness*

$\mathcal{W}_L = (W, R, \{C_1, \dots, C_k\}, D)$ *certifies that any MST of G has a node in L with degree at most $\lfloor \frac{2|W| + |R| + k - 2}{|W|} \rfloor$.*

The proof of Lemma 10 is omitted from this extended abstract. Since it bounds the average degree of L , Corollary 11 follows.

Corollary 11 *Given a witness \mathcal{W}_L as in the previous lemma, it holds that for any fractional MST T^f with edge incidence vector x , the minimum fractional degree in L , i.e. $\min_{v \in L} \sum_{e \in \delta(v)} x_e$, is at most $\frac{2|W| + |R| + k - 2}{|W|}$.*

Recall that $d_{\min}^L(T) = \min_{v \in L} \deg_T(v)$ is the minimum degree over all nodes in L in tree T , and let Δ_{OPT}^L be the maximum possible value of $d_{\min}^L(T)$ attainable by any MST; the algorithm we describe in this section finds an MST in which $d_{\min}^L(T)$ is at least $\Delta_{\text{OPT}}^L - O(\frac{\log n}{\log \log n})$.

We now introduce the following notation: For a tree T , we denote by $S_{\leq d}^L$ the subset of nodes in L which have degree d or less in T .

Definition 1 We call an edge e dotted with respect to an MST T and a set W , if e is a non-tree edge incident to a node in W and it is feasible with respect to some edge e' in T such that e' is not incident to a node in W .

If the set W has low degree, we are interested in dotted edges with respect to W , since swapping in these edges and removing the edges they are feasible to, increases the degree of W .

The MINDMST Algorithm

Like the algorithm in Section 4, at the beginning of each phase, we pick a degree d such that in the current tree, $|S_{\leq d+1}| \leq \frac{\log n}{\log \log n} |S_{\leq d}|$.

Note that this inequality must be true for some d lying between $d_{\min}^L(T)$ and $d_{\min}^L(T) + \frac{\log n}{\log \log n}$. The aim is to improve the degree of some vertex of degree d or less without introducing more vertices of the same or lower degree. We begin with the center set $W_0 = S_{\leq d+1}^L$. Our initial clusters are the components obtained when we remove W_0 from T . Note that every node in L outside W_0 belongs to a cluster and has degree at least $d + 2$.

We maintain a set of special nodes N which is also initialized to $S_{\leq d+1}^L$. Consider the Steiner tree induced on N by the given tree T . We freeze the Steiner tree edges adjacent to N — that is, we disallow the algorithm from swapping out these edges— and put them in F . (Note that freezing edges reduces the set of dotted edges that can be used.)

In each iteration i , we look for a dotted edge $e = (u, v)$ adjacent to a node u in the witness W_i , and an intra-cluster tree edge $e' = (u', v')$ with which it can be swapped. If we find such a pair, we take the following steps:

1. Remove u from W_i .
2. Form a new cluster C_u by merging u along with all the children clusters of u , except for those which would involve a merge along an edge between u and a $(d + 1)$ -degree node.
3. Split the cluster containing (u', v') along e' into two clusters $C_{u'}$ and $C_{v'}$.
4. Let u' be the endpoint of e' that has a tree path to u . Let w be the first node in N on the u' - u path. Add u' to N . Freeze the edges on this path that are incident on u' and on w , and add the new frozen edges to F . The u' - v path in T is called the *tail* of this swap.

Step 1 allows us to delete an edge incident on u at a later step. Steps 2 and 3 are needed to ensure that we never swap out more than one edge incident on any node. Finally, step 4 ensures that the sequence of swaps found by the end of the phase can be executed concurrently without resulting in non-tree structures.

We repeat the above process until we either remove a node of degree d or less from W , or there are no useful intra-cluster edges left. Either event ends the current phase. In the former case, Lemma 12 shows that we can find an improvement. Once again the frozen edges play a crucial role in the proof, which appears in the full version of the paper.

In the latter case, Theorem 13 shows that we can turn the resulting structure into a witness \mathcal{W}_L^* certifying that the tree has close to the optimal degree.

Lemma 12 *If we remove a vertex u from W , we can find a sequence of edge swaps which increases the degree of u by 1. Moreover, performing all these swaps results in a tree without introducing any new nodes of degree d or less.*

Theorem 13 *If the MINDMST algorithm halts with a tree with minimum degree d for a node in L , then $\Delta_{\text{OPT}}^L \leq d + 4 + O(\frac{\log n}{\log \log n})$.*

Proof. We begin by showing how to convert the structure produced by the MINDMST algorithm into a witness of the type in Lemma 10. Suppose the algorithm begins a phase with an MST T , chooses a degree d , sets $W_0 = S_{\leq d+1}^L$, but then cannot improve the degree of any vertex in L of degree d or lower. It terminates after $t > 0$ iterations with $W_t \supseteq S_{\leq d}^L$ as the center set and $C_1, C_2, \dots, C_{k'}$, as the clusters.

Let $W^* = W_t$, and let I be the set of inter-cluster edges with respect to the clusters $C_1, \dots, C_{k'}$. Recall that F is the set of edges frozen by the algorithm. Let $R^* = F \cup I$. We now describe how to modify the above clusters to get our witness. Let \mathcal{C}^* be a collection of clusters, and let D^* be a collection of nodes. We begin with \mathcal{C}^* and D^* empty. For $1 \leq i \leq k'$, let $C = C_i$ and do the following:

If cluster C is internally connected in every MST, then add C to \mathcal{C}^* . If not, let T' be an MST of G in which C is not internally connected, and let (u, v) be an edge of $T|_C \setminus T'|_C$.

Consider the clusters C_u and C_v that would result from splitting C about the edge (u, v) . If both C_u and C_v have edges in T to W_t , then split C and recurse on each cluster C_u and C_v . If only one of the two, say C_u has a tree edge to W_t , then split C , recurse on C_u , and add the vertices in C_v to D^* .

The process terminates when every node outside W^* either belongs to D^* or to a cluster C that has been put in \mathcal{C}^* . Let us label the clusters in \mathcal{C}^* by $C_1^*, C_2^*, \dots, C_k^*$, $k \geq k'$. Note that by construction, every vertex in D^* has a path in $T - W^*$ to a cluster in \mathcal{C}^* .

Lemma 14 $\mathcal{W}_L^* = (W^*, R^*, \mathcal{C}^* = \{C_1^*, \dots, C_k^*\}, D^*)$ is a low-degree witness.

The proof of Lemma 14 is omitted due to space constraints. To finish the proof of Theorem 13, we compute the bound that we get by using Lemma 10. First we count the number of clusters created by the algorithm. The number of clusters is the number of components formed by W_t from T , of which there are at most $(d + 1)|W_t|$, plus the number of splits. There are at most $2|W_0 - W_t|$ additional clusters created by the splits during the algorithm. To see this, note that there are a total of $|W_0 - W_t|$ splits caused by finding a useable dotted edge, and each split creates one new cluster by splitting along the intra-cluster edge (u', v') involved in the swap. There are also some clusters split by edges adjacent to two $(d + 1)$ -degree nodes, but there can be at most $|W_0 - W_t|$ of these.

The number of clusters after the algorithm stops is thus at most $(d + 1)|W_t| + 2|W_0 - W_t|$. In the process of breaking apart clusters with more than one edge to W_t , at most $|W_t|$ more splits may happen. Thus the total number k of clusters in \mathcal{C}^* is at most $(d + 2)|W_t| + 2|W_0 - W_t|$.

There are at most $2|W_0| + 2|W_0 - W_t|$ edges in F , and at most $2|W_0 - W_t|$ inter-cluster edges with respect to the clusters $C_1, \dots, C_{k'}$ when the algorithm

terminates. Therefore $|R^*| \leq 2|W_0| + 4|W_0 - W_t|$. According to Lemma 10, the witness \mathcal{W}_L^* gives an upper bound of

$$\frac{2|W^*| + |R^*| + k - 2}{|W^*|} \leq \frac{(d+4)|W_t| + 6|W_0 - W_t| + 2|W_0| - 2}{|W_t|}$$

on the minimum degree of at least one node in L . Using the fact that $|S_{\leq d+1}| \leq \frac{\log n}{\log \log n} |S_{\leq d}|$, we conclude that $\Delta_{\text{OPT}}^L \leq d + 4 + O(\frac{\log n}{\log \log n})$.

6 Combining Upper and Lower Bounds

Now we describe our MSTDB algorithm more formally. Recall from Section 3 that our MSTDB algorithm works in phases. Each phase employs algorithms MAXDMST and MINDMST to improve either a high degree vertex or a low degree vertex in L ; when both improvements fail, their failure is justified by two combinatorial witnesses.

Let us now look into a phase of the algorithm in more detail. Each phase of MSTDB begins by picking a d such that

$$|S_{\leq B_L - d + 1}| \leq \frac{\log n}{\log \log n} |S_{\leq B_L - d}| \quad \text{and} \quad |S_{\geq B_H + d - 1}| \leq \frac{\log n}{\log \log n} |S_{\geq B_H + d}|.$$

It is easy to show that one can always find such a d between 0 and $2\frac{\log n}{\log \log n}$.

For the rest of the phase, vertices with degree $B_H + d$ or more are considered “high degree” vertices and those in L with degree $B_L - d$ are considered “low degree”. We employ MAXDMST and MINDMST to reduce the degree of a high degree vertex and increase the degree of a low degree vertex respectively.

Before calling MAXDMST and MINDMST we need to ensure that the improvements they perform do not interfere. For this purpose, we look at all edges e such that one end point of e lies in $S_{\leq B_L - d + 1}$ and the other in $S_{\geq B_H + d - 1}$. Note that e might be a tree edge or a non-tree edge. We freeze all such edges, which means that we neither add any of the non-tree edges nor delete any of the tree edges while executing algorithms MAXDMST or MINDMST. Lemma 15 guarantees that this is enough to make the algorithm terminate. The proof of Lemma 15 is omitted from the extended abstract. Theorem 16 shows that when both MAXDMST and MINDMST fail with two combinatorial witnesses, at least one of the witnesses is good. The proof will appear in the full version of the paper.

Lemma 15 *Algorithm MSTDB terminates.*

Theorem 16 *If MSTDB fails, then in every MST T of the graph, either:*

- (1) *The maximum degree of T is more than B_H , or*
- (2) *Some node in L has degree less than B_L .*

References

1. T. M. Chan. Euclidean bounded-degree spanning tree ratios. In *Proceedings of the nineteenth annual symposium on Computational geometry*, pages 11–19. ACM Press, 2003.

2. J. Edmonds. Maximum matching and a polyhedron with 0–1 vertices. *Journal of Research National Bureau of Standards*, 69B:125–130, 1965.
3. J. Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.
4. S. Even and R. E. Tarjan. Network flow and testing graph connectivity. *SIAM Journal on Computing*, 4(4):507–518, Dec. 1975.
5. T. Fischer. Optimizing the degree of minimum weight spanning trees. Technical Report 14853, Dept of Computer Science, Cornell University, Ithaca, NY, 1993.
6. M. Fürer and B. Raghavachari. Approximating the minimum-degree Steiner tree to within one of optimal. *Journal of Algorithms*, 17(3):409–423, Nov. 1994.
7. J. Hopcroft and R. Karp. An $n^{5/2}$ algorithm for maximum matching in bipartite graphs. *SIAM Journal on Computing*, 2:225–231, 1973.
8. R. Jothi and B. Raghavachari. Degree-bounded minimum spanning trees. In *Proc. 16th Canadian Conf. on Computational Geometry (CCCG)*, 2004.
9. S. Khuller, B. Raghavachari, and N. Young. Low-degree spanning trees of small weight. *SIAM J. Comput.*, 25(2):355–368, 1996.
10. J. Könemann and R. Ravi. A matter of degree: improved approximation algorithms for degree-bounded minimum spanning trees. In *Proceedings of ACM STOC, 2000*.
11. J. Könemann and R. Ravi. Primal-dual meets local search: approximating MST’s with nonuniform degree bounds. In ACM, editor, *Proceedings ACM STOC, 2003*.
12. R. Krishnan and B. Raghavachari. The directed minimum degree spanning tree problem. In *FSTTCS*, pages 232–243, 2001.
13. C. H. Papadimitriou and U. Vazirani. On two geometric problems related to the traveling salesman problem. *J. Algorithms*, 5:231–246, 1984.
14. R. Ravi, M. V. Marathe, S. S. Ravi, D. J. Rosenkrantz, and H. B. Hunt, III. Many birds with one stone: multi-objective approximation algorithms. In *Proceedings of ACM STOC, 1993*.
15. R. Ravi, M. V. Marathe, S. S. Ravi, D. J. Rosenkrantz, and H. B. Hunt, III. Approximation algorithms for degree-constrained minimum-cost network-design problems. *Algorithmica*, 31, 2001.