A Push-Relabel Algorithm for Approximating Degree Bounded MSTs

Kamalika Chaudhuri[1], Satish Rao[1], Samantha Riesenfeld[1], and Kunal Talwar[2]

[1] U.C. Berkeley
[2] Microsoft Research Mountain View, CA
{kamalika, satishr, samr}@cs.berkeley.edu,
kunal@microsoft.com

Abstract. Given a graph G and degree bound B on its nodes, the bounded-degree minimum spanning tree (BDMST) problem is to find a minimum cost spanning tree among the spanning trees with maximum degree B. This bi-criteria optimization problem generalizes several combinatorial problems, including the Traveling Salesman Path Problem (TSPP).

An $(\alpha,\ f(B))$-approximation algorithm for the BDMST problem produces a spanning tree that has maximum degree $f(B)$ and cost within a factor α of the optimal cost. Könemann and Ravi [13,14] give a polynomial-time $(1+\frac{1}{\beta},\ bB(1+\beta)+\log_b n)$-approximation algorithm for any $b>1$, $\beta>0$. In a recent paper [2], Chaudhuri et al. improved these results with a $(1,\ bB+\sqrt{b}\log_b n)$-approximation for any $b>1$. In this paper, we present a $(1+\frac{1}{\beta},\ 2B(1+\beta)+o(B(1+\beta)))$-approximation polynomial-time algorithm. That is, we give the first algorithm that approximates both degree and cost to within a constant factor of the optimal. These results generalize to the case of non-uniform degree bounds.

The crux of our solution is an approximation algorithm for the related problem of finding a minimum spanning tree (MST) in which the maximum degree of the nodes is minimized, a problem we call the minimum-degree MST (MDMST) problem. Given a graph G for which the degree of the MDMST solution is Δ_{OPT}, our algorithm obtains in polynomial time an MST of G of degree at most $2\Delta_{\text{OPT}}+o(\Delta_{\text{OPT}})$. This result improves on a previous result of Fischer [4] that finds an MST of G of degree at most $b\Delta_{\text{OPT}}+\log_b n$ for any $b>1$, and on the improved quasipolynomial algorithm of [2].

Our algorithm uses the push-relabel framework developed by Goldberg [7] for the maximum flow problem. To our knowledge, this is the first instance of a push-relabel approximation algorithm for an NP-hard problem, and we believe these techniques may have larger impact. We note that for $B=2$, our algorithm gives a tree of cost within a $(1+\epsilon)$-factor of the optimal solution to TSPP and of maximum degree $O(\frac{1}{\epsilon})$ for any $\epsilon>0$, even on graphs *not* satisfying the triangle inequality.

1 Introduction

Given a graph and upper bounds on the degrees of its nodes, the bounded-degree minimum spanning tree (BDMST) problem is to find a minimum cost spanning tree among the spanning trees that obey the degree bounds. This bi-criteria optimization problem generalizes several combinatorial problems, including the Traveling Salesman Path Problem (TSPP), which corresponds to the case when degrees are restricted to 2 uniformly. Since we do not assume the triangle inequality, approximations for the BDMST problem must relax the degree constraint, unless P equals NP.

Let $c_{opt}(B)$ be the cost of an optimal solution to the BDMST problem, given input graph G and uniform degree bound B. We call a BDMST algorithm an $(\alpha,\ f(B))$-approximation algorithm if, given graph G and bound B, it produces a spanning tree that has cost at most $\alpha \cdot c_{opt}(B)$ and maximum degree $f(B)$. Könemann and Ravi give, to our knowledge, the first BDMST approximation scheme [13]: a polynomial-time $(1+\frac{1}{\beta},\ bB(1+\beta)+\log_b n)$-approximation algorithm for any $b > 1$, $\beta > 0$. They illustrate the close relationship between the BDMST problem and the problem of finding an MST in which the maximum degree of the nodes is minimized, a problem we call the minimum-degree MST (MDMST) problem. Using a novel cost-bounding technique based on Lagrangean duality, Könemann and Ravi show that the MDMST problem can essentially be used as a black box in an algorithm for the BDMST problem. In a subsequent paper [14], they use primal dual techniques and give similar results for nonuniform degree bounds.

The BDMST and MDMST problem are different generalizations of the same unweighted problem: given an unweighted graph $G = (V, E)$, find a spanning tree of G of minimum maximum degree. Fürer and Raghavachari [5] give a lovely algorithm for this problem that outputs an MST with degree $\Delta_{\mathrm{OPT}} + 1$. Their algorithm finds a sequence of swaps in a laminar family of subtrees of G such that the sequence results in an improvement to the degree of some high-degree node, without creating any new high-degree nodes. The laminar structure relies on the property that an edge $e \in E$ that is not in a spanning tree T can replace *any* tree edge on the induced cycle of $T \cup e$. This property is not maintained in weighted graphs because a non-tree edge can only replace other tree edges of equal cost. The structure of an improving sequence of swaps in a weighted graph can therefore be significantly more complicated.

Könemann and Ravi rely on an MDMST algorithm due to Fischer [4]. Given a graph G for which the MDMST solution is Δ_{OPT}, Fischer's algorithm finds an MST of G of degree at most $b\Delta_{\mathrm{OPT}} + \log_b n$ for any $b > 1$. In a recent paper [2], Chaudhuri et al. give an improved MDMST algorithm based on finding augmenting paths of swaps. The algorithm in [2] simultaneously enforces upper *and lower* bounds on degrees, which, by using linear programming duality and techniques of [13,3], is shown to result in an optimal-cost $(1,\ bB(1+\beta)+\log_b n)$-approximation BMDST algorithm for any $b > 1$. At the expense of quasipolynomial time, [2] also gives an algorithm that produces an MST with degree at most $\Delta_{\mathrm{OPT}} + O(\frac{\log n}{\log\log n})$, leading to a $(1,\ B+O(\frac{\log n}{\log\log n}))$-approximation for the BDMST problem (in quasipolynomial time).

In this paper, we present a polynomial-time BDMST algorithm which we show to be a $(1+\frac{1}{\beta},\ 2B(1+\beta)+o(B(1+\beta)))$-approximation scheme for any $\beta > 0$. That is, we give the first algorithm that approximates both degree and cost to within a constant factor of the optimal.

For example, for $B = 2$, all previous algorithms would produce a tree with near-logarithmic degree and cost within a constant factor of the optimal; our algorithm, in contrast, approximates both the degree and the cost to within a constant factor.

For the sake of a simpler exposition, we describe our BDMST results in the setting of uniform degree bounds. Our techniques imply analogous results even in the case of more general non-uniform degree bounds. Though our BDMST algorithm does not simultaneously enforce upper and lower degree bounds, our techniques here do apply to a version of the BDMST problem in which *lower* bounds on node degrees must be respected, which may be of independent interest.

The crux of our solution is an improved approximation algorithm for the MDMST problem that uses the push-relabel framework invented by Goldberg [7] for the max flow problem (and fully developed by Goldberg and Tarjan [8]). Given a graph G for which the degree of the MDMST solution is Δ_{OPT}, our algorithm obtains in polynomial time an MST of G of degree at most $2\Delta_{\mathrm{OPT}} + o(\Delta_{\mathrm{OPT}})$.

While Fischer's MDMST solution is locally optimal with respect to single edge swaps in the current tree, our algorithm explores a more general set of moves that may consist of long sequences of branching, interdependent changes to the tree. Surprisingly, the push-relabel framework can be delicately adapted to explore these sequences. The basic idea that we borrow from Goldberg [7] is to give each node a label and permit "excess" to flow from a higher labeled node to lower labeled nodes. Nodes are allowed to increase their label when they are unable to get rid of their excess. For max-flow, the excess was a preflow, while in our case, the excess refers to excess degree. To our knowledge, this is the first instance of a push-relabel approximation algorithm for an NP-hard problem and we are intrigued by the possibility that this framework may be extended to search what may appear to be complicated neighborhood structures for other optimization problems.

We note that for $B = 2$, our BDMST algorithm gives a tree of cost within a $(1+\epsilon)$-factor of the optimal solution to TSPP and of maximum degree $O(\frac{1}{\epsilon})$ for any $\epsilon > 0$. Our work does not assume the triangle inequality; when the triangle inequality holds, Hoogeveen [10] gives a $\frac{3}{2}$-approximation of TSPP based on Christofides' algorithm. The Euclidean version of the BDMST problem has also been widely studied. See, for example, [15,12,1,11].

Independent of our work, Ravi and Singh [16] give an algorithm for the MDMST problem with an additive error of k, where k is the number of distinct weight classes. We note that this bound is incomparable to the one presented here, and does not improve previous results for the BDMST problem. More recently, Goemans [6] has announced an algorithm for the BDMST problem with an additive error of 2.

1.1 Techniques

All known algorithms for the MDMST problem repeatedly swap a non-tree edge $e \in E$ with a tree edge $e' \in T$ of the same weight, where e' is on the induced cycle in $T \cup e$ in the current MST T. Fischer proceeds by executing any swap that improves a degree d node without introducing new degree d nodes, for selected high values of d. He shows that when the tree is locally optimal, the maximum degree of the tree is at most $b\Delta_{\mathrm{OPT}} + \log_b n$, for any $b > 1$, where Δ_{OPT} is the degree of the optimal MDMST solution. Moreover, as shown in [2], this analysis is tight.

To illustrate the difficulty of the MDMST problem, we next describe a pathological MST T in a graph G (see Figure 1): the tree T has a long path consisting of $O(n)$ nodes ending in a node u of degree d. The children of u each have degree $(d-1)$; the children of the degree $(d-1)$ nodes have degree $(d-2)$, and so on until we get to the leaves. Each edge on the path has cost ϵ, and an edge from a degree $(d-i+1)$ node to its degree $(d-i)$ child has cost i. In addition, each of the degree $(d-i)$ nodes has a cost-i edge to one of the nodes on the path. For some d with $d = O(\log n/\log\log n)$, the number of nodes in the graph is $O(n)$.

Note that an MST of G with optimal degree consists of the path along with the non-tree edges and has maximum degree three. On the other hand, every cost-neutral swap that improves the degree of a degree-$(d-i)$ node in the current tree increases the degree of a degree-$(d-i-1)$ node. Hence the tree T is locally optimal for the algorithms of [4,13]. Moreover, all the improving edges are incident on a single component of low degree nodes; one can verify that the algorithm of [2] starting with this tree will not be able to improve the maximum degree. In

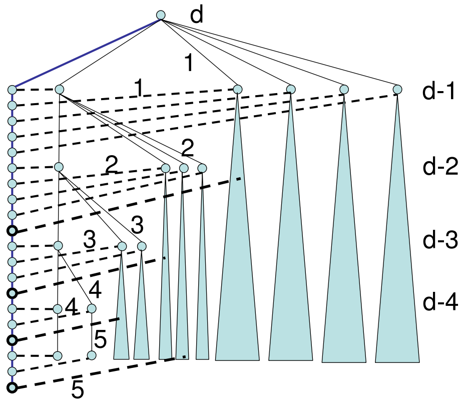

Fig. 1. Graph G and a locally optimal tree. The shaded triangles represent subtrees identical to the corresponding ones shown rooted at the same level. The bold nodes represent a path and the bold dotted edges correspond to a set of edges going to similar nodes in the subtrees denoted by shaded triangles.

fact, a slightly modified instance, G', where several of the non-tree edges are incident on the same node on the path, is not improvable beyond $O(d)$. Previous techniques do not discriminate between different nodes with degree less than $d-1$ and hence cannot distinguish between G and G'.

On the other hand, our MDMST algorithm, described in Section 2, may perform a swap that improves the degree of a degree d node by creating one or more new degree d nodes. In turn, it attempts to improve the degree of these new degree d nodes. We note that in the process the algorithm may end up undoing the original move. However, the labels ensure that this process cannot continue indefinitely. These two new degree d nodes cannot necessarily be improved independently since they may rely on the same edge or use edges that are incident to the same node. Moreover, this effect snowballs as more and more degree d nodes are created.

As previously mentioned, Goldberg's push-relabel framework helps us tame this beast of a process. A high degree node may only relieve a unit of excess degree using a non-tree edge that is incident to nodes of lower labels. Thus, while two high degree nodes may be created by a swap, at least they are guaranteed to have lower labels than the label of the node initiating the swap.

We define a notion of a *feasible* labeling and prove that our MDMST algorithm maintains one. During the course of the algorithm, there is eventually a label L such that the number of nodes with that label is not much larger than the number of nodes with label $L+1$. We use feasibility to show that all nodes with labels L and higher must have high average degree in *any* MST, thus obtaining a lower bound on Δ_{OPT}. This degree lower bound also holds for any fractional MST in the graph.

Combining our MDMST algorithm with the cost-bounding techniques of Könemann and Ravi [13] gives us our result for the BDMST problem.

2 Minimum-Degree MSTs

MDMST Problem: Given a weighted graph $G=(V,E,c)$, find an MST T of G such that $\max_{v\in V}\{\deg_T(v)\}$ is minimized.

2.1 The MDMST Algorithm

Our algorithm is based on the *push-relabel* scheme used in an efficient algorithm for the max flow problem [7,9]. Starting with an arbitrary MST of the graph, our algorithm runs in phases. The idea is to reduce the maximum degree in each phase using a push-relabel technique. If we fail to make an improvement at some phase, we get a set of nodes with high labels which serves as a certificate of near-optimality.

More formally, let Δ_i be the maximum degree of any node in the tree T_i at the beginning of phase i, also called the Δ_i-phase. During the Δ_i-phase, we either modify T_i to get T_{i+1} such that the maximum degree in T_{i+1} is less than Δ_i *or* we find a proof that $\Delta_i \leq 2\Delta_{\mathrm{OPT}} + O(\sqrt{\Delta_{\mathrm{OPT}}})$. (The constants hidden in the big-O notation are small—see Section 2.3.)

We now describe a general phase of the algorithm. Let T be the tree at the beginning of the current phase, and let Δ be the maximum degree over all nodes in T. Let $\mathbf{N}$ be the set of nonnegative integers. Given a *labeling* $l : V \rightarrow \mathbf{N}$, we extend the labeling to E by defining $l(e) = \max\{l(u), l(v)\}$ for $e = (u, v)$. The label of a node is a measure of its potential. In our algorithm, all nodes are initialized at the beginning of each phase to have label $l(v) = 0$. At any time, let *level* i be defined as the set of nodes that currently have label i.

In addition to being assigned a label, each node is given an initial *excess*. (Excess can also be formally defined as a function from V to $\mathbf{N}$.) At the beginning of the phase, each vertex with degree Δ is initialized to have an excess of 1; all other vertices initially have excess 0. We call a vertex that has positive excess *overloaded*.

We define a *swap* to be a pair of edges (e, e') such that $e \in T$, $e' \notin T$, $c(e) = c(e')$, and e lies on the unique cycle of $T \cup e'$. For a node u and a tree T, let S_u^T denote the set of swaps (e, e') such that e is incident on u and e' is not incident on u. We call a swap in S_u^T *useful* for u because it can be used to decrease the degree of u. We say that a labeling l is *feasible* for a tree T if for all nodes $u \in V$, for every swap $(e, e') \in S_u^T$, $l(e) \leq l(e') + 1$. A swap $(e, e') \in S_u^T$ is called *permissible* for u if $l(u) \geq l(e') + 1$. This notion of feasibility is crucial in establishing a lower bound on the optimal degree (and hence proving an approximation guarantee) when the algorithm terminates.

The current phase proceeds as follows: Let L be the label of the lowest level containing overloaded nodes. If there is an overloaded node u in level L that has a permissible, useful swap $(e, e') \in S_u^T$, modify T by deleting $e = (u, v)$ and adding $e' = (u', v')$. Then decrease the excess on u by one. If u' now has degree Δ or more, add one to its excess; if v' has degree Δ or more, add one to its excess. If no overloaded node in level L has a permissible, useful swap, then *relabel* to $L + 1$ all overloaded nodes in level L. Repeat this loop.

The phase ends either when no node is overloaded, or when there is an overloaded node with label $\log_2 n$. Note that if the phase ends for the former reason, then the tree at the end of the phase has maximum degree at most $\Delta - 1$. See Figure 2 for a formal algorithm.

Since each node is relabeled at most $\log_2 n$ times, the number of iterations in any phase of the algorithm is bounded by $n^2 \log_2 n$. Hence the algorithm runs in polynomial time.

In the next few sections, we build the tools used to argue that when a phase ends with some overloaded node in level $\log_2 n$, the algorithm produces a *witness* to the fact that $\Delta \leq 2\Delta_{\mathrm{OPT}} + O(\sqrt{\Delta_{\mathrm{OPT}}})$.

The proof of the following lemma is omitted from the extended abstract.

Lemma 1. *The algorithm always maintains a feasible labeling.*

```
Algorithm push_relabel_MDMST

$T \leftarrow$ arbitrary MST of $G$.
While witness not found do
    $\Delta \leftarrow$ maximum degree over nodes in $T$.
    Initialize labels to zero; put excess of 1 on nodes with degree $\Delta$.
    Repeat
        $L \leftarrow$ lowest level that contains overloaded nodes.
        $U_L \leftarrow$ overloaded nodes with label $L$.
        If there is a node $u \in U_L$ that has a permissible, useful swap $(e, e')$ where $e = (u, v)$
            $T \leftarrow T \setminus \{e\} \cup \{e'\}$;
            Set excess on $u$ and $v$ to 0;
            If an endpoint of $e'$ has degree $\Delta$ or more
                set its excess to 1.
        else
            Relabel all nodes in $U_L$ to $L + 1$.
    until there is an overloaded node with label at least $\log_2 n$
      or there are no more overloaded nodes.
    If some node has label $\log_2 n$
        Pick $L$ such that level $L$ is the highest sparse level.
        Let $W$ be the set of nodes with label strictly higher than $L$.
        Let $W'$ be the set of nodes with label higher than or equal to $L$.
        Output tree $T$ and witness $(W, W')$.
endwhile.
```

Fig. 2. An algorithm for the MDMST problem

2.2 Cascades and Involuntary Losses

For an integer L, let V_L be level L, i.e. the set of nodes with label L, and let U_L be the set of overloaded nodes in V_L. For convenience, we imagine placing flags on nodes when we relabel them. We start with all the pending flags cleared.

In each iteration of the algorithm, we find the lowest L such that U_L is non-empty, i.e. there are some overloaded nodes with label L. If we can find any swap (e, e') that is permissible and useful for a node in U_L, we execute the swap and clear pending flags (if set) on the endpoints of e. Let us call this a label-L swap. If no such swaps exist, we raise the label of all nodes in U_L by one and set their pending flags.

Lemma 2. *During the Δ-phase, no node ever has degree more than Δ.*

Proof. We use induction on the number of swaps. In the beginning of the phase, the maximum degree is Δ. Any swap (e, e') decreases the degree of a node in U_L and adds at most one to the degree of a node with strictly lower label. By choice of L, all nodes with lower labels had degree at most $\Delta - 1$ before the swap. Since a swap adds at most one to any vertex degree, the induction holds. The lemma follows.

Consider a swap (e, e'), where $e = (u, v)$ and $e' = (u', v')$, that is useful for an overloaded node u. If v is *not* overloaded, then we say that the swap (e, e') *causes* v an *involuntary loss* in degree.

We call a swap a *root swap* if it is a useful swap for a node with its pending flag set. Let (e, e') be a non-root swap that occurs in the sequence of swaps made by the algorithm. The swap (e, e') was performed in order to decrease the degree of node u (where $e = (u, v)$). There is a unique swap (f, f') in the sequence that most recently (before the (e, e') swap was done) increased the degree of u (so u is an endpoint of edge f'). We say that swap (e, e') can *blame* swap (f, f'), and we call (f, f') the *parent swap* of (e, e'). Recall that a label-L swap reduces the degree of a node with label L, and note that every non-root swap has a label strictly smaller than its parent. Moreover each swap is the parent of at most two other swaps. This parent relation naturally defines a directed graph on the set of swaps, each component of which is an in-tree rooted at one of the root swaps. We call the set of swaps in a component a *cascade*. In other words, a cascade corresponds to the set of swaps sharing a single root swap as an ancestor. Note that one cascade does not necessarily finish before another begins. The label of the cascade is defined to be the label of the root swap in it.

As noted above, each swap has at most two children and they have a strictly smaller label. Thus it follows that:

Lemma 3. *A label-i cascade contains at most 2^{i-j} label-j swaps.*

We say that the cascade *contains* an involuntary loss if some swap in the cascade causes it. Since each swap causes at most one involuntary loss, the lemma above implies:

Corollary 4. *A label-i cascade contains at most $2^{i-j+1} - 1$ involuntary losses to nodes with labels at least j.*

Proof. An involuntary loss to a label-k node must be caused by a swap with label k or higher.

2.3 Obtaining the Witness

We now show that when the algorithm terminates, we can find a combinatorial structure, that we call the witness, which establishes the near optimality of the final tree. Our witness consists of a partition $\mathcal{C} = \{W, C_1, \ldots, C_k\}$ and a subset $W' \subset V$ such that $W \subseteq W'$. Call an edge e MST-worthy if there is some minimum spanning tree of G that contains e. The witness has the following property: any MST-worthy edge $e = (u, v)$ leaving C_i has at least one endpoint in W', i.e. for any MST-worthy edge $(u, v) : u \in C_i, v \notin C_i$, $|\{u, v\} \cap W'| \geq 1$. The following lemma, essentially contained in [4] shows that a witness establishes a lower bound on the minimum degree of any MST.

Lemma 5. [4] *Let $\mathcal{W} = \{\mathcal{C}, W'\}$ be a witness defined as above. Then any minimum spanning tree of G has maximum degree at least $(k + |W| - 1)/|W'|$.*

Proof. Consider any MST T of G and let G' be the graph formed by shrinking each of the sets $C_1, \ldots, C_k$ to a single node. T must contain a spanning tree

T' of G' and hence must have at least $k + |W| - 1$ edges from G'. Moreover, each such edge is MST-worthy and hence has at least one endpoint in W'. The average degree of W' is thus at least $(k + |W| - 1)/|W'|$.

Let the Δ-phase be the last phase of the final iteration of the algorithm. It ends with an overloaded node at level $l = \log_2 n$. Let $W_L = \cup_{i \geq L} V_i$ be the set of nodes with label at least L and $s_L = |W_L|$ be the final number of nodes in W_L. Fix a constant $c \geq 2$. We search top-down for the highest level labeled $l - j$, for some j, $0 \leq j < l$, such that $s_{l-(j+1)} < c \cdot s_{l-j}$. We call level $l - (j + 1)$ *sparse.* Since $l \geq \log_c n$, such a sparse level must exist for any $c \geq 2$. Then for every level i such that $i \geq g$, it is true that $s_i \geq c \cdot s_{i+1}$.

We first show that the average degree of W_g is high.

Lemma 6. *The average degree of W_g in the final tree is at least $\Delta - 1 - \frac{2c}{c-2}$.*

Proof. Each node enters the set W_g with degree Δ, after which it may lose at most one degree from a useful swap and it may suffer some involuntary losses. Thus the total degree of W_g is at least $(\Delta - 1)|W_g|$ minus the number of involuntary losses to W_g.

Each involuntary loss to W_g occurs in a cascade and by Corollary 4, the number of involuntary losses to W_g in a label-i cascade is at most 2^{i-g+1}. Recall that the root swap of a cascade is useful to a pending node that moved up one label. The total number of involuntary losses to W_g during the course of the phase is at most

$$\begin{aligned}
\sum_{i \geq g} 2^{i-g+1} |W_i| &= \sum_{i \geq g} 2^{i-g+1} s_i \\
&\leq \sum_{i \geq g} 2^{i-g+1} \frac{s_g}{c^{i-g}} \\
&\leq 2 s_g \sum_{i \geq g} \left(\frac{2}{c}\right)^{i-g} \\
&\leq 2|W_g|\left(\frac{c}{c-2}\right)
\end{aligned}$$

Thus the average degree Δ_{av} of W_g is at least $\Delta - 1 - 2\frac{c}{c-2}$.

Our witness is now constructed as follows: W' is the set W_{g-1} and W is W_g. $C_1, \ldots, C_k$ are the components formed by deleting W_g from the final tree. The feasibility of the labeling implies that this indeed is a witness. Moreover, k is at least $\Delta_{av}|W_g| - 2(|W_g| - 1)$, where Δ_{av} is the average degree of W_g in T. Lemma 5 then implies that the maximum degree of any MST must be at least

$$\left(\Delta - 2 - \frac{2c}{c-2}\right) \frac{|W|}{|W'|} = \left(\Delta - 2 - \frac{2c}{c-2}\right) \left(\frac{|W_g|}{|W_{g-1}|}\right) \geq \frac{\Delta}{c} - \frac{2}{c-2} - 2$$

Rearranging, we get

$$\Delta \leq c(\Delta_{\mathrm{OPT}} + 2 + \frac{2}{c-2})$$

Setting c to be $2 + \frac{2}{\sqrt{\Delta_{\text{OPT}}}}$, we get

$$\Delta \leq 2\Delta_{\text{OPT}} + 4\sqrt{\Delta_{\text{OPT}}} + 6 + \frac{4}{\sqrt{\Delta_{\text{OPT}}}}$$

Theorem 7 summarizes the results of Section 2.

Theorem 7. *Given a graph G, the* `push_relabel_MDMST` *algorithm obtains in polynomial time an MST of degree Δ, where $\Delta \leq 2\Delta_{\text{OPT}} + O(\sqrt{\Delta_{\text{OPT}}})$.*

This algorithm along with Lagrangean relaxation techniques from [13] gives a bi-criteria approximation for the Bounded-degree MST problem. We omit the proof from this extended abstract.

Theorem 8. *For any $\beta > 0$, there is a polynomial-time algorithm that, given a graph G and degree bound B, computes a spanning tree T with maximum degree at most $2\,(1+\beta)\,B + O(\sqrt{(1+\beta)\,B})$ and cost at most $\left(1+\frac{1}{\beta}\right)\text{OPT}_{LD(B)}$.*

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. K. Chaudhuri, S. Rao, S. Riesenfeld, and K. Talwar. What would edmonds do? augmenting paths and witnesses for bounded degree msts. In *Proceedings of APPROX/RANDOM*, 2005.
3. J. Edmonds. Maximum matching and a polyhedron with 0–1 vertices. *Journal of Research National Bureau of Standards*, 69B:125–130, 1965.
4. T. Fischer. Optimizing the degree of minimum weight spanning trees. Technical Report 14853, Dept of Computer Science, Cornell University, Ithaca, NY, 1993.
5. 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.
6. M. Goemans. Personal communication, 2006.
7. A. V. Golberg. A new max-flow algorithm. Technical Report MIT/LCS/TM-291, Massachussets Institute of Technology, 1985. Technical Report.
8. A. V. Goldberg and R. E. Tarjan. A new approach to the maximum flow problem. In *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pages 136–146. ACM Press, 1986.
9. A. V. Goldberg and R. E. Tarjan. A new approach to the maximum-flow problem. *J. ACM*, 35(4):921–940, 1988.
10. J. A. Hoogeveen. Analysis of christofides' heuristic: Some paths are more difficult than cycles. *Operation Research Letters*, 10:291– 295, 1991.
11. R. Jothi and B. Raghavachari. Degree-bounded minimum spanning trees. In *Proc. 16th Canadian Conf. on Computational Geometry (CCCG)*, 2004.
12. S. Khuller, B. Raghavachari, and N. Young. Low-degree spanning trees of small weight. *SIAM J. Comput.*, 25(2):355–368, 1996.
13. J. Könemann and R. Ravi. A matter of degree: Improved approximation algorithms for degree-bounded minimum spanning trees. *SIAM Journal on Computing*, 31(6):1783–1793, Dec. 2002.

14. J. Könemann and R. Ravi. Primal-dual meets local search: approximating MST's with nonuniform degree bounds. In ACM, editor, *Proceedings of the Thirty-Fifth ACM Symposium on Theory of Computing, San Diego, CA, USA, June 9–11, 2003*, pages 389–395, New York, NY, USA, 2003. ACM Press.
15. C. H. Papadimitriou and U. Vazirani. On two geometric problems related to the traveling salesman problem. *J. Algorithms*, 5:231–246, 1984.
16. R. Ravi and M. Singh. Delegate and conquer: An LP-based approximation algorithm for minimum degree msts. In *Proceedings of ICALP*, 2006.