Theoretical Computer Science 410 (2009) 4489–4503

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

A push–relabel approximation algorithm for approximating the minimum-degree MST problem and its generalization to matroids

Kamalika Chaudhuri [a,1], Satish Rao [b], Samantha Riesenfeld [b], Kunal Talwar [c,*,2]

[a] U.C. San Diego, United States
[b] U.C. Berkeley, United States
[c] Microsoft Research, Mountain View, CA, United States

ARTICLE INFO

Keywords:
Approximation algorithms
Push–relabel
Degree-bounded network design
Spanning trees

ABSTRACT

In the minimum-degree minimum spanning tree (MDMST) problem, we are given a graph G, and the goal is to find a minimum spanning tree (MST) T, such that the maximum degree of T is as small as possible. This problem is *NP*-hard and generalizes the Hamiltonian path problem. We give an algorithm that outputs an MST of degree at most $2\Delta_{\mathrm{OPT}}(G) + o(\Delta_{\mathrm{OPT}}(G))$, where $\Delta_{\mathrm{OPT}}(G)$ denotes the degree of the optimal tree. This result improves on a previous result of Fischer [T. Fischer, Optimizing the degree of minimum weight spanning trees. Technical Report 14853, Dept. of Computer Science, Cornell University, Ithaca, NY, 1993] that finds an MST of degree at most $b\Delta_{\mathrm{OPT}}(G) + \log_b n$, for any $b > 1$.

The MDMST problem is a special case of the following problem: given a k-ary hypergraph $G = (V, E)$ and weighted matroid M with E as its ground set, find a minimum-cost basis (MCB) T of M such that the degree of T in G is as small as possible. Our algorithm immediately generalizes to this problem, finding an MCB of degree at most $k^2\Delta_{\mathrm{OPT}}(G, M) + O(k\sqrt{k\Delta_{\mathrm{OPT}}(G, M)})$.

We use the push–relabel framework developed by Goldberg [A. V. Goldberg, A new max-flow algorithm, Technical Report MIT/LCS/TM-291, Massachusetts Institute of Technology, 1985 (Technical Report)] for the maximum-flow problem. To our knowledge, this is the first use of the push–relabel technique in an approximation algorithm for an *NP*-hard problem.

The MDMST problem is closely connected to the bounded-degree minimum spanning tree (BDMST) problem. Given a graph G and degree bound B on its nodes, the BDMST problem is to find a minimum cost spanning tree among the spanning trees with maximum degree B. Previous algorithms for this problem by Könemann and Ravi [J. Könemann, R. Ravi, A matter of degree: Improved approximation algorithms for degree-bounded minimum spanning trees, SIAM Journal on Computing 31(6) (2002) 1783–1793; J. Könemann, R. Ravi, Primal-dual meets local search: Approximating MST's with nonuniform degree bounds, in: Proceedings of the Thirty-Fifth ACM Symposium on Theory of Computing, 2003, pp. 389–395] and by Chaudhuri et al. [K. Chaudhuri, S. Rao, S. Riesenfeld, K. Talwar, What would Edmonds do? Augmenting paths and witnesses for bounded degree MSTs, in: Proceedings of APPROX/RANDOM, 2005, pp. 26–39] incur a near-logarithmic additive

* Corresponding author.
E-mail addresses: kamalika@soe.ucsd.edu (K. Chaudhuri), satishr@cs.berkeley.edu (S. Rao), samr@cs.berkeley.edu (S. Riesenfeld), kunal@microsoft.com (K. Talwar).

1 Research done at UC Berkeley.

2 Research partially done at UC Berkeley.

0304-3975/$ – see front matter
doi:10.1016/j.tcs.2009.07.029

error in the degree. We give the first BDMST algorithm that approximates both the degree and the cost to within a constant factor of the optimum. These results generalize to the case of nonuniform degree bounds.

1. Introduction

Given a weighted graph $G = (V, E, c)$, the minimum-degree minimum spanning tree (MDMST) problem is to find a minimum spanning tree (MST) of G that minimizes the maximum degree. This problem, even in the unweighted case, generalizes the Hamiltonian Path problem and is therefore *NP*-hard. In this paper we give a polynomial-time algorithm that, given a weighted graph G, outputs an MST of degree at most $2\Delta_{\mathrm{OPT}}(G) + O(\sqrt{\Delta_{\mathrm{OPT}}(G)})$, where $\Delta_{\mathrm{OPT}}(G)$ denotes the degree of an optimal solution. This is the first constant-factor approximation to this problem.

The MDMST problem requires us to optimize the degree in a graph G of a minimum-cost base in the graphical matroid of G. We consider a more general setting where the (hyper)graph and the matroid are not necessarily related. Given a k-ary hypergraph $G = (V, E)$ and a weighted matroid M with E as the ground set, the minimum-degree minimum-cost base (MDMCB) problem is to find a minimum-cost base T of M that minimizes the degree of T in G. (See Section 6.1 for a more complete definition.)

Unlike in the MDMST problem, where the vertices have a very specific relation to the elements of the matroid, this setting allows the nodes of the hypergraph to correspond to arbitrary subsets of the elements. The only restriction is that each element of the matroid occurs in at most k subsets. As a concrete example of the MDMCB problem, consider a network in which each link is controlled by a subset of a set of autonomous entities, with the restriction that no link is controlled by more than k entities. The goal is to build an MST of the network such that the maximum number of links controlled by a single entity is minimized. Other natural combinatorial optimization problems can also be formalized as instances of the MDMCB problem.

Our MDMST algorithm generalizes in a straightforward way to the MDMCB problem. Given a k-ary hypergraph $G = (V, E)$ and weighted matroid $M = (E, \mathcal{I}, c)$, it outputs an MCB of M that has degree in G at most $k^2\Delta_{\mathrm{OPT}}(G, M) + O(k^{\frac{3}{2}}\sqrt{\Delta_{\mathrm{OPT}}(G, M)})$, where $\Delta_{\mathrm{OPT}}(G, M)$ is the degree of an optimal solution.

The MDMST and MDMCB algorithms use the push–relabel framework invented by Goldberg [9] (and fully developed by Goldberg and Tarjan [10]) for the max-flow problem. To our knowledge, this work is the first use of the push–relabel technique in an approximation algorithm for an *NP*-hard problem.

Subsequent to the publication of a previous version of this work [4], Goemans [8] gave an algorithm for the MDMST problem that outputs an MST of degree at most $\Delta_{\mathrm{OPT}}(G) + 2$. Finally Singh and Lau [22] gave a MDMST algorithm that outputs a tree of degree at most $\Delta_{\mathrm{OPT}}(G) + 1$, which is optimal unless $P = NP$. Both these works are based on polyhedral techniques showing that extremal solutions to the natural linear-programming relaxation have particular structure. Using a lemma of Goemans [8], we show that running our push–relabel algorithm on the set of tight edges in an extremal solution gives a $(\Delta_{\mathrm{OPT}}(G) + 2)$-algorithm as well.

While these recent results dominate our results for the MDMST problem, the techniques developed in this paper may be of independent interest.

One of the motivations for studying the MDMST problem is its connection to the bounded-degree minimum spanning tree (BDMST) problem. Given a graph and upper bounds on the degrees of its nodes, the BDMST problem is to find a spanning tree of minimum cost, among the ones 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 [15]: 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 and MDMST problems. 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, [16], they use primal dual techniques and give similar results for nonuniform degree bounds.

Könemann and Ravi rely on an MDMST algorithm due to Fischer [6]. Given a graph G for which the MDMST solution is $\Delta_{\mathrm{OPT}}(G)$, Fischer's algorithm finds an MST of G of degree at most $b\Delta_{\mathrm{OPT}}(G) + \log_b n$ for any $b > 1$. In a recent paper, [3], the authors give an improved MDMST algorithm based on finding augmenting paths of swaps. The algorithm simultaneously enforces upper *and lower* bounds on degrees, which, by using linear programming duality and techniques of [15,5], is shown to result in an optimal-cost $(1,\ \frac{b}{2-b}B + O(\log_b n))$-approximation BDMST algorithm for any $b \in (1, 2)$. At the expense of quasipolynomial time, the authors [3] also give an algorithm that produces an MST with degree at most $\Delta_{\mathrm{OPT}}(G) + O(\frac{\log n}{\log\log n})$, leading to a quasipolynomial-time $(1,\ B + O(\frac{\log n}{\log\log n}))$-approximation algorithm for the BDMST problem.

The push–relabel MDMST algorithm in this paper also implies a polynomial-time $(1+\frac{1}{\beta},\ 2B(1+\beta)+O(\sqrt{B(1+\beta)}))$-approximation scheme for the BDMST problem, for any $\beta > 0$. Thus we give the first algorithm that approximates both degree and cost to within a constant factor of the optimum.

For example, for $B = 2$ (i.e. for the TSPP), all previous algorithms would produce a tree with near-logarithmic degree and cost within a constant factor of the optimum; our algorithm, in contrast, gives a tree of cost within a $(1+\epsilon)$-factor of the optimal solution 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 [11] gives a $\frac{3}{2}$-approximation to the TSPP based on Christofides' algorithm. The Euclidean version of the BDMST problem has also been widely studied. See, for example, [19,13,2,12].

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 nonuniform 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 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 [7] give a lovely algorithm for this problem that outputs an MST with degree $\Delta_{\mathrm{OPT}}(G)+1$. 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.

While Fischer's MDMST solution is locally optimal with respect to single edge-swaps, 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 [9] 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. We are intrigued by the possibility that the push–relabel framework may be extended to search what may appear to be complicated neighborhood structures for other optimization problems.

Independent of our work, Ravi and Singh [20] give an algorithm for the MDMST problem with an additive error of p, where p 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. The aforementioned works of Goemans [8] and Singh and Lau [22] also give algorithms for the BDMST problem that achieve additive errors of 2 and 1, respectively, while giving optimal cost.

These iterative rounding techniques have been further generalized to other degree constrained network design problems [1,14,17,18]. Notably, our results on MDMCB have been improved by Király, Lau and Singh [14], who give an algorithm that outputs an minimum-cost base which violates the degree constraint by an additive $(k-1)$, for k-ary hypergraphs.

1.1. Techniques

All known algorithms for the MDMST problem begin with an arbitrary MST T, and repeatedly update T by swapping 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 $T \cup \{e\}$. 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 $b\Delta_{\mathrm{OPT}}(G) + \log_b n$, for any $b > 1$, where $\Delta_{\mathrm{OPT}}(G)$ is the degree of the optimal MDMST solution for G. Moreover, this analysis is tight [3].

To illustrate the difficulty of the MDMST problem, we describe here a pathological MST T in a graph G (see Fig. 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 3. 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 [6,15]. Moreover, all the improving edges are incident on a single component of low-degree nodes; one can verify that the algorithm of Chaudhuri et al. [3] starting with this tree will not be able to improve the maximum degree. In 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 3, 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-

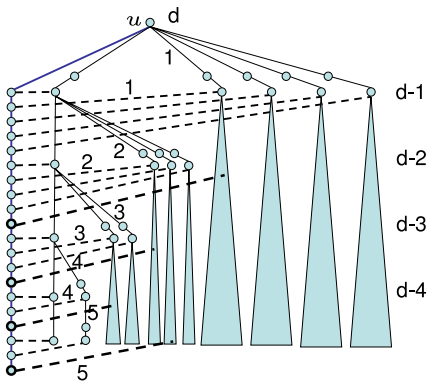

Fig. 1. Graph G and a locally optimal tree. The shaded triangles at each level represent subtrees identical to the explicit subtree (on the left) rooted at the same level. The bold nodes represent a path and the bold dotted edges correspond to a set of edges going to similar nodes in the subtrees denoted by shaded triangles.

d nodes, which cannot necessarily be improved independently since their improvements 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. While the algorithm may end up undoing a previous swap, the labels ensure that this process cannot continue indefinitely.

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 p^* such that the number of nodes with label at least p^* is not much larger than the number of nodes with label at least $p^* + 1$. We use feasibility to show that all nodes with labels p^* and higher must have high average degree in *any* MST, thus obtaining a lower bound on $\Delta_{\mathrm{OPT}}(G)$. This degree lower bound also holds for any fractional MST of the graph G.

Combining our MDMST algorithm with the cost-bounding techniques of Könemann and Ravi [15] gives our result for the BDMST problem (Section 7).

1.2. Organization of the paper

The rest of the paper is organized as follows. Section 2 defines the notation used throughout the paper, and Section 3 describes our push–relabel algorithm for the MDMST problem. We give two different (and incomparable) bounds on the performance of the algorithm in Sections 4 and 5. In Section 5.2, we use a result of Goemans to derive an algorithm with an additive error of 2. Section 6 generalizes our results to the MDMCB problem. Finally, in Section 7, we give our results for the BDMST problem.

2. Definitions and notation

For a graph $G = (V, E)$ and an MST T of G, the degree $\Delta(T)$ of T is defined to be the maximum over nodes u in V, of the degree of u in T. When T is obvious from context, we simply write its degree as Δ.

For a subset $F \subseteq E$ of edges and a subset $U \subseteq V$ of nodes, let $i_F(U)$ denote the set of edges in F that have both endpoints in U, and let $\vartheta_F(U)$ denote the set of edges in F incident on U, i.e. $i_F(U) = \{(u, v) \in F : u, v \in U\}$ and $\vartheta_F(U) = \{(u, v) \in F : u \in U \text{ or } v \in U\}$. Finally, $\delta_F(u)$ denotes the set of edges in F incident on a vertex u, i.e. $\delta_F(u) = \{e \in F : u \in e\}$.

Let $\mathbf{N}$ be the set of nonnegative integers. A *labeling* l of the nodes is a function $l : V \to \mathbf{N}$. For a labeling l and an integer p, let *level* p be defined as the set $\{v : l(v) = p\}$ of nodes that have label p, and let $W_p = \{v : l(v) \geq p\}$ be the set of nodes with labels at least p. For a real number $\mu \geq 1$, level p is called μ-*sparse* if $|W_p| \leq \mu |W_{p+1}|$.

Given an MST T, we define a *swap* in T 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 $T \cup \{e'\}$. Note that if (e, e') is a swap in T, then $T \setminus \{e\} \cup \{e'\}$ is also an MST of G. For a node u and a tree T, let S_u^T denote the set of swaps (e, e') in T such that e is incident on u and e' is not incident on u. We call a swap (e, e') in S_u^T *useful* for u because it can be used to decrease the degree of u; i.e. the degree of u in $T \setminus \{e\} \cup \{e'\}$ is one less than that in T.

Given a labeling l on V, we extend it to a labeling on E by defining $l(e) = \max\{l(u), l(v)\}$ for $e = (u, v)$. We say that a labeling l is *feasible* for a tree T if for all nodes $u \in V$, for every swap $(e, e') \in S_u^T$, $l(e) \leq l(e') + 1$. Given a labeling l and an MST T, a swap $(e, e') \in S_u^T$ is called *permissible* for u if $l(u) \geq l(e') + 1$. We show in Section 3.2 that our algorithm maintains a feasible labeling. Consequently, each permissible swap $(e, e') \in S_u^T$ satisfies $l(u) = l(e') + 1$.

We defer the definitions for the MDMCB problem to Section 6.1.

Algorithm PR-MDMST(G, μ)

$T \leftarrow$ arbitrary MST of G.
Repeat
  $\Delta \leftarrow$ maximum degree over nodes in T.
  Initialize labels to 0.
  Put excess of 1 on nodes with degree Δ. Put excess of 0 on all other nodes.
  Repeat
    $p \leftarrow$ lowest level that contains an overloaded node.
    Select the set $U_p \leftarrow$ overloaded nodes with label p.
    If there is a node $u \in U_p$ that has a permissible, useful swap (e, e') where $e = (u, v)$
      $T \leftarrow T \setminus \{e\} \cup \{e'\}$;
      Set excess on the endpoints of e to 0;
      For each endpoint of e' that has degree Δ.[3]
        set its excess to 1.
    else
      Relabel all nodes in U_p to $p + 1$.
  until there are no more overloaded nodes
    or there is a μ-sparse level.
until there is a μ-sparse level p^*.
Let $F \subset T$ be the edges in T *not* incident on W_{p^*+1}.
Output tree T and the pair $\mathcal{W}^{p^*} = (F, W_{p^*})$.

Fig. 2. A push–relabel algorithm for the MDMST problem.

3. 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.

3.1. The push–relabel MDMST algorithm

Starting with an arbitrary MST of the graph, our algorithm runs in phases. The idea is to reduce the maximum degree of the tree in each phase using a push–relabel technique. If we fail to make an improvement in some phase, we find 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, either we modify T_i to get T_{i+1} such that the maximum degree in T_{i+1} is less than Δ_i, or we output a certificate that Δ_i is close to optimal.

For a general phase of the algorithm, let T be the tree at the beginning of the phase, and let Δ be the degree of T. The algorithm maintains a labeling l. The algorithm maintains the invariant that the labeling l is feasible with respect to the current tree T. This notion of feasibility is crucial in establishing a lower bound on the optimal degree when the algorithm terminates.

In addition, each node is given an initial *excess*. The excess of a node is one if its degree is Δ, and zero otherwise. As we show in Lemma 5, the algorithm maintains the invariant that the degree of each node is at most Δ. For notational convenience, we call a node *overloaded* if it has positive excess.

The PR-MDMST algorithm takes as input a graph G and a real-valued parameter $\mu \geq 1$. The parameter μ determines the termination condition of the main loop of the algorithm. In Sections 4 and 5, we derive two different bounds on the approximation ratio of the algorithm; the parameter μ is chosen appropriately in the two cases.

We now describe a general phase of the algorithm. See Fig. 2 for a formal description. The phase proceeds as follows: The label $l(u)$ of each node u is initialized to zero. The excess of each node of degree Δ is initialized to one; the excess of every other node is initialized to zero.

Let p be the label of the lowest level containing overloaded nodes. If there is an overloaded node u in level p 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 v has positive excess, decrement its excess as well. 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 p has a permissible, useful swap, then *relabel* to $p + 1$ all overloaded nodes in level p. Repeat this loop until either there are no overloaded nodes or there is a μ-sparse level. Note that if the phase ends for the former reason, then the tree at the end of the phase has maximum degree at most $\Delta - 1$. In the latter case, we show that Δ is close to the optimal degree.

[3] We prove in Lemma 5 that the degree of any node never goes above Δ.

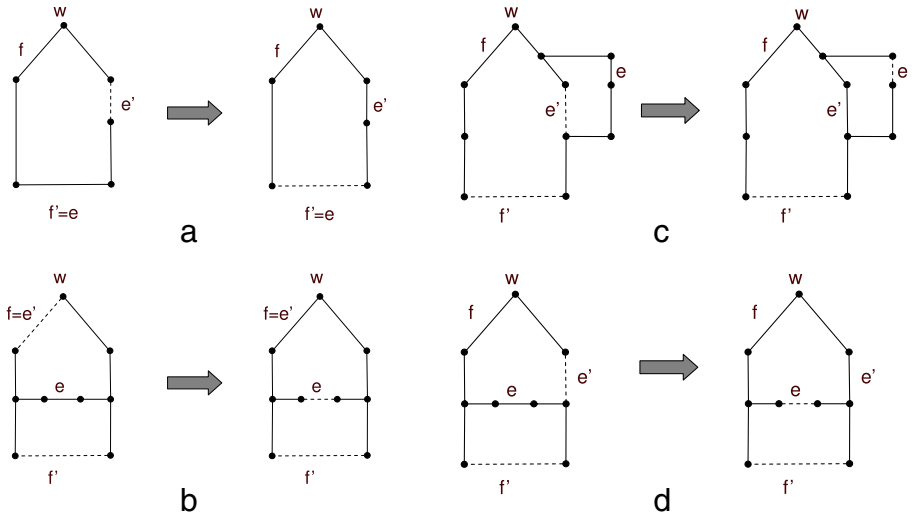

Fig. 3. Proof of Lemma 1.

If some node gets label n, there is guaranteed to be a 1-sparse level. Thus each node gets relabeled at most n times per phase, for any choice of the input parameter $\mu \geq 1$. The total number of iterations of the inner loop in any phase of the algorithm is therefore bounded by n^2. Since each phase (except the last) decreases the maximum degree of T by one, there are at most n phases. The algorithm therefore runs in polynomial time.

The algorithm outputs a tree T and a pair $\mathcal{W}^{p^*} = (F, X)$, where F is a forest on G and X is a subset of nodes. In the rest of this section, we show how to interpret (F, X) as a certificate that the degree of T is close to $\Delta_{\mathrm{OPT}}(G)$. We do this in two different ways, in Sections 4 and 5, leading to two incomparable bounds on the approximation ratio of the algorithm.

Remark: In Section 4, we set μ to a constant larger than 2. In this case, the number of relabels per node is bounded by $\log_2 n$, resulting in a faster algorithm.

3.2. Feasibility

We first prove a crucial lemma.

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

Proof. We prove this by induction on the number of iterations in a phase. At the beginning of any phase, all labels are zero, which is a feasible labeling. In one step of the algorithm, we either update a label or perform a permissible swap (e, e'). Since we increment the label of a node only when it has no permissible swaps, feasibility is maintained in the first case.

In the second case, since we change the structure of the tree, the set of available swaps may change. Consider a feasible swap (e, e'), where $e = (u, v)$ and $e' = (u', v')$, and the swap is permissible for u or v (or both). Let T be the tree before the (e, e') swap, and let T' be the tree after the swap. Consider a swap (f, f') in T'. To show that feasibility is maintained, we need to show that $l(f) \leq l(f') + 1$.

If the swap (f, f') already exists in T, feasibility holds inductively. However, the swap may have been missing in the tree T, but may appear in tree T' for one of the following three reasons.

- $f' \in T$ *and hence not available for the swap:* See Fig. 3(a). In this case, $f' = e$. The cycle formed by adding f' to T' includes e' and f, otherwise T already has a cycle. Moreover $c(f) = c(f') = c(e')$. Therefore (f, e') is a swap in T, and feasibility in T implies that $l(f) \leq l(e') + 1$. On the other hand, since (e, e') is a permissible swap in T, $l(e) \geq l(e') + 1$. Thus $l(f) \leq l(e') + 1 \leq l(e) = l(f')$.
- $f \notin T$ *and hence not available for the swap:* See Fig. 3(b). In this case, $f = e'$. As (e, e') is a swap in T and (e', f') is a swap in T', the cycle formed by adding f' to T includes e, and (e, f') is a valid swap in T. $l(e) \leq l(f') + 1$ by feasibility of T, and $l(e) = l(e') + 1$ by permissibility. Thus $l(f) = l(e') \leq l(f')$.
- $f \in T$ *and* $f' \notin T$*:* See Fig. 3(c) and 3(d). Since (e, e') is a swap in T, there is a unique cycle in $T \cup e'$ that contains e. If f does not lie on this cycle, as illustrated in Fig. 3(c), the swap (f, f') already exists in T and the claim holds by induction. Otherwise the cycle in $T \cup e'$ contains f, and the only structure in which this happens is illustrated in Fig. 3(d). Since e' is the missing edge in T of the cycle in $T \cup e'$, it must be the case that $c(e') \geq c(f) = c(f')$, or else $T \setminus \{f\} \cup \{e'\}$ would have strictly smaller cost than the MST T. Similarly, since f' is the missing edge of the cycle in $T \cup f'$, $c(f') \geq c(e) = c(e)$. Therefore $c(e) = c(e') = c(f) = c(f')$. Moreover, (f, e') and (e, f') are both available swaps in T. Thus $l(f) \leq l(e') + 1 = l(e) \leq l(f') + 1$.

We have shown that, in all cases, the swap (f, f') is feasible. Hence the induction holds. □

3.3. The witness

In this section, we introduce the notion of a *witness*. A witness is a combinatorial structure produced by our algorithms at termination to guarantee the near-optimality of the output. Our witness consists of a forest $F \subseteq E$ that is contained in some MST of G, along with a subset $X \subset V$. A pair (F, X) is a witness if it has the following property: For every MST T of G containing F, every edge in $T \setminus F$ is incident on X.

Lemma 2 ([6]). *Let $\mathcal{W} = (F, X)$ be a witness for a graph $G = (V, E)$ as defined above. Then any (fractional) minimum spanning tree of G has maximum degree at least $\frac{|V|-|F|-1}{|X|}$.*

Proof. Consider an MST T of G, and let T' be an MST containing F that has maximal intersection with T. By the exchange property, $T' \setminus F$ is contained in T. The witness property implies that every edge in $T' \setminus F$ is incident on X. Since there are $|V| - |F| - 1$ edges in $T' \setminus F$, the average degree of X in $T' \setminus F$, and therefore in T, is at least $\frac{|V|-|F|-1}{|X|}$. Since a fractional MST is a convex combination of integral ones, the claim follows. □

We now show that the pair (F, X) output by the algorithm PR-MDMST is a witness.

Lemma 3. *Let T be the MST of G and $l : V \to \mathbf{N}$ the labeling when the algorithm* PR-MDMST *terminates. For any integer p, let F be the subset of edges in T that are not incident on W_{p+1}, and let $X = W_p$. Then $\mathcal{W}^p = (F, X)$ is a witness.*

Proof. Assume the contrary, and let T' be an MST of G that contains F and also contains an edge $e' \notin F$ not incident on W_p. By the exchange property, there is an edge $e \in T \setminus F$ such that (e, e') is a swap in T. Since $e \in T \setminus F$, it is incident on W_{p+1} and thus $l(e) \geq p + 1$. On the other hand, e' is not incident on W_p and thus $l(e') \leq p - 1$. This, however, contradicts the feasibility of the labeling. □

3.4. Involuntary losses

Let p^* be the μ-sparse level used by the algorithm to compute a witness. From Lemma 2 and Lemma 3, it follows that any (fractional) MST of G has degree at least $\frac{|V|-|F|-1}{|W_{p^*}|}$. This ratio can be rewritten as $\frac{(|V|-|F|-1)}{|W_{p^*+1}|} \cdot \frac{|W_{p^*+1}|}{|W_{p^*}|}$. Note that the numerator of the first term is precisely the number of edges incident on W_{p^*+1} in T. The second term is bounded by $\frac{1}{\mu}$, where μ is the sparseness of level p^*. The next lemma follows immediately.

Lemma 4. *Let T be the MST of G, $l : V \to \mathbf{N}$ the labeling, and p^* the μ-sparse level when the algorithm* PR-MDMST *terminates. Let $\vartheta_T(W_{p^*+1})$ be the set of edges in T incident on W_{p^*+1}. Then any (fractional) MST of G has degree at least $\frac{1}{\mu} \cdot \frac{|\vartheta_T(W_{p^*+1})|}{|W_{p^*+1}|}$.*

Thus, to prove a lower bound on $\Delta_{\mathrm{OPT}}(G)$, we need to lower bound $|\vartheta_T(W_{p^*+1})|$. Towards this end, we distinguish between two different ways a node can lose degree during the course of the algorithm.

We say that a swap (e, e') executed by the algorithm *causes* a loss in degree to a node u if e is incident on u. A loss in degree to a node u that is caused by a swap (e, e') is called a *voluntary loss* if u is overloaded before the swap is executed; otherwise it is called an *involuntary loss*. By definition, voluntary losses do not decrease the degree of a node below $\Delta - 1$. Note that every swap (e, e') executed by the algorithm causes a voluntary loss to at least one endpoint of e (and an involuntary loss to at most one endpoint of e).

Suppose the algorithm terminates with a μ-sparse level in the Δ-phase. The last time it is relabeled, each node in W_{p^*+1} has degree at least Δ and is therefore overloaded. If each node in W_{p^*+1} suffered only voluntary losses in degree since its last relabeling, then its degree in T would be at least $\Delta - 1$. However, a node in W_{p^*+1} may also suffer from involuntarily losses, which may decrease its degree arbitrarily. Hence a node-by-node analysis is insufficient. To get a lower bound on the average degree of W_{p^*+1} in T, we instead bound the total number of involuntary losses to nodes in W_{p^*+1}. We do this in two different ways in the next two sections.

4. A constant-factor approximation

In this section, we show that the algorithm outputs a tree T of degree $\Delta \leq 2\Delta_{\mathrm{OPT}}(G) + O(\sqrt{\Delta})$. To bound the number of involuntary losses, we define a partitioning of the swaps executed by the algorithm into *cascades*. Each cascade can be charged to a relabel, which enables us to bound the number of involuntary losses to W_{p^*+1} in terms of the size of this set.

4.1. Cascades

Recall that, for an integer p, U_p is defined to be the set of overloaded nodes in level p. For the purpose of analysis, we introduce the notion of flagging a node. The flag indicates that the node has been relabeled but its excess has not been

removed. In addition, we give each node an *overloading-swap* field. The overloading-swap field of a node u points to the swap that put excess on it after its last relabel. We start with all the flags cleared and all overloading-swap fields set to null.

In each iteration of the Δ-phase of the algorithm, we find the lowest p such that U_p is non-empty, i.e. there is an overloaded node with label p. If we can find any swap (e, e') that is permissible and useful for a node in U_p, we execute the swap and clear flags (if set) on the endpoints of e. Moreover, for each endpoint u' of e' that is now overloaded, we set the overloading-swap field of u' to (e, e'). If there is no swap that is permissible and useful for any node in U_p, we increment the label, set the flag, and clear the overloading-swap field for each node in U_p.

The following lemma shows that no node has excess larger than one during the course of the algorithm, which implies, in particular, that the overloading-swap field is never overwritten before it is cleared to null.

Lemma 5. *During the Δ-phase, no node ever has degree more than Δ.*

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

We define the *label of a swap* (e, e') to be the label of e when the swap is executed. We call a swap a *root swap* if it is useful for a flagged node. Note that a flagged node has its overloading-swap field set to null. Let (e, e') be a non-root swap that occurs in the sequence of swaps executed by the algorithm. The swap (e, e') is executed in order to relieve the excess of an endpoint u of e. Let (f, f') be the swap pointed to by the overloading-swap field for node u when swap (e, e') is executed. Thus (f, f') is the last swap in the sequence that increases the degree of u and precedes (e, e'). We call (f, f') the *parent swap* of (e, e'). Note that the flagging procedure ensures that every non-root swap executed by the algorithm has a parent.

A label-p swap, by definition, reduces the degree of a node with label p. Since excess flows from a higher-labeled node to a lower-labeled node, the label of every non-root swap is strictly smaller than the label of its parent. The 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 define a *cascade* to be the set of swaps in a component of this Directed Acyclic Graph. In other words, a cascade corresponds to the set of swaps sharing the same root swap as an ancestor. Note that the cascades may be interleaved in the sequence of swaps executed by the algorithm. The *label of a cascade* is defined to be the label of the root swap in it.

Each swap is the parent of at most two swaps which each have a strictly smaller label. Thus it follows that:

Lemma 6. *A label-p cascade contains at most 2^{p-q} label-q swaps.*

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

Corollary 7. *A label-p cascade contains at most $(2^{p-q+1} - 1)$ involuntary losses to nodes with labels at least q.*

Proof. An involuntary loss to a label-r node must be caused by a swap with label at least r. The bound follows by summing the number of swaps with label r, for r between q and p. □

4.2. Computing the approximation ratio

Armed with the bound of Corollary 7, we now proceed to lower bound $\Delta_{\mathrm{OPT}}(G)$. Recall from Section 3.4 that it suffices to lower bound the average degree of W_{p^*+1} in T, where W_{p^*} is a μ-sparse level and T is the tree output by the algorithm.

Lemma 8. *Let p be an integer greater than p^*. Then $|W_p| > \mu |W_{p+1}|$.*

Proof. Each iteration of a phase of the algorithm decreases the size of at most one level p and increases the size of the level $p + 1$. Thus the only level that can go from not being μ-sparse to being μ-sparse is level p. Since the algorithm terminates as soon as it finds a μ-sparse level, it terminates with exactly one μ-sparse level. □

Lemma 9. *The number of involuntary losses to nodes in W_{p^*+1} is at most $2|W_{p^*+1}|(\frac{\mu}{\mu-2})$.*

Proof. Each involuntary loss to a node in W_{p^*+1} occurs in a cascade, and by Corollary 7, the number of involuntary losses to nodes in W_{p^*+1} in a label-p cascade is at most 2^{p-p^*}. The total number of involuntary losses to nodes in W_{p^*+1} during the course of the phase is at most

$$\begin{aligned}\sum_{p\geq p^*+1} 2^{p-p^*}|W_p| &< \sum_{p\geq p^*+1} 2^{p-p^*}\frac{|W_{p^*+1}|}{\mu^{p-p^*-1}}\\ &= 2\,|W_{p^*+1}| \sum_{p\geq p^*+1}\left(\frac{2}{\mu}\right)^{p-p^*-1}\\ &= 2|W_{p^*+1}|\left(\frac{\mu}{\mu-2}\right). \quad \square\end{aligned}$$

We are now ready to establish the approximation ratio of the algorithm.

Theorem 10. *Given a graph G and a constant $\mu > 2$, the* PR-MDMST *algorithm obtains in polynomial time an MST of degree Δ, where $\Delta \leq \mu\, \Delta_{\mathrm{OPT}}(G) + 2 + \frac{2\mu}{\mu-2}$.*

Proof. The PR-MDMST algorithm, when executed on graph G, terminates with a tree T and a pair $\mathcal{W}^{p^*} = (F, X)$. We now compute the number $\left|\vartheta_T(W_{p^*+1})\right|$ of edges incident on W_{p^*+1} in T. Each node in W_{p^*+1} has degree at least $(\Delta - 1)$ after its last voluntary loss, and it may then suffer some involuntary losses. Using the bound from Lemma 9, the sum of degrees of nodes in W_{p^*+1} in T is at least $(\Delta - 1 - \frac{2\mu}{\mu-2})|W_{p^*+1}|$. Since there are at most $\left|W_{p^*+1}\right| - 1$ edges in T that have both endpoints in W_{p^*+1}, the number of edges in T that are incident on W_{p^*+1} is at least $(\Delta - 2 - \frac{2\mu}{\mu-2})|W_{p^*+1}|$. Thus from Lemma 4,

$$\Delta_{\mathrm{OPT}}(G) \geq \left(\Delta - 2 - \frac{2\mu}{\mu-2}\right)\frac{1}{\mu}.$$

Rearranging, we get

$$\Delta \leq \mu\, \Delta_{\mathrm{OPT}}(G) + 2 + \frac{2\mu}{\mu-2}. \quad \square$$

Setting μ to be $2 + \frac{2}{\sqrt{\Delta_{\mathrm{OPT}}(G)}}$, we get

$$\Delta \leq 2\Delta_{\mathrm{OPT}}(G) + 4\sqrt{\Delta_{\mathrm{OPT}}(G)} + 4$$

Corollary 11. *Given a graph G, there is a polynomial-time algorithm that outputs an MST of degree Δ, where $\Delta \leq 2\Delta_{\mathrm{OPT}}(G) + O(\sqrt{\Delta_{\mathrm{OPT}}(G)})$.*

5. An additive approximation

In this section, we show a different upper bound on the performance of the algorithm that is better than the bound in the previous section if the graph G is *everywhere-sparse*, i.e. for every subset U of nodes, the induced subgraph on U is sparse.

More precisely, for a graph G, let the *local density* $s(G)$ be defined as the density of the densest subgraph of G: $s(G) = \max_{U \subset V}\left\{\frac{|i_E(U)|}{|U|}\right\}$. We show that on input G and $\mu = 1$, the PR-MDMST algorithm outputs an MST of degree at most $\Delta_{\mathrm{OPT}}(G) + s(G)$. In Section 5.2, we combine this result with a lemma from Goemans [8] to get a $(\Delta_{\mathrm{OPT}}(G) + 2)$-algorithm.

5.1. A density-based bound

Let T be the tree and $\mathcal{W}^{p^*}$ the witness output by the PR-MDMST algorithm on input G and $\mu = 1$. Let Δ be the degree of T. Since $\mu = 1$, the sets W_{p^*} and W_{p^*+1} must be equal, and hence, level p^* is empty when the algorithm terminates. The following lemma bounds $\left|\vartheta_T(W_{p^*+1})\right|$.

Lemma 12. *Let T be the MST, l the labeling, and p^* the empty level when the algorithm* PR-MDMST *terminates. Let $i_E(W_{p^*+1})$ be the set of edges in G that have both endpoints in W_{p^*+1}, and let $\vartheta_T(W_{p^*+1})$ be the set of edges in T that are incident on W_{p^*+1}. Then $\left|\vartheta_T(W_{p^*+1})\right| \geq (\Delta - 1)\left|W_{p^*+1}\right| + 1 - \left|i_E(W_{p^*+1})\right|$.*

Proof. For a node $u \in W_{p^*+1}$, let $\delta_T(u) \subset T$ be the set of edges in T incident on u, and let $\delta_L(u) \subset E$ be the set of edges that node u loses involuntarily after its last voluntary loss. Since each node has degree at least $\Delta - 1$ after its last voluntary loss, it follows that $|\delta_T(u)| \geq \Delta - 1 - |\delta_L(u) \setminus \delta_T(u)|$. Moreover, if v is the last node to be relabeled, $|\delta_T(v)| \geq \Delta$. Thus the sum of degrees of nodes in W_{p^*+1} in T is at least $(\Delta - 1)\left|W_{p^*+1}\right| + 1 - \sum_{u \in W_{p^*+1}} |\delta_L(u) \setminus \delta_T(u)|$.

An edge (u, v) may be in $\delta_L(u)$ or $\delta_L(v)$ but not both. It follows that $\sum_{u \in W_{p^*+1}} |\delta_L(u) \setminus \delta_T(u)| \leq \left|\cup_{u \in W_{p^*+1}} (\delta_L(u) \setminus \delta_T(u))\right|$. Recall that every involuntary loss to a node in W_{p^*+1} comes from an edge in $i_E(W_{p^*+1})$. Thus each edge in $\cup_{u \in W_{p^*+1}} (\delta_L(u) \setminus \delta_T(u))$ is in $i_E(W_{p^*+1}) \setminus i_T(W_{p^*+1})$.

An edge (u, v) in $i_T(W_{p^*+1})$ contributes to both $\delta_T(u)$ and $\delta_T(v)$. Thus the number of edges in T incident on W_{p^*+1} is

$$\begin{aligned}
\sum_{u \in W_{p^*+1}} |\delta_T(u)| - \left|i_T(W_{p^*+1})\right| &\geq (\Delta - 1)\left|W_{p^*+1}\right| + 1 - \left|\cup_{u \in W_{p^*+1}} (\delta_L(u) \setminus \delta_T(u))\right| - \left|i_T(W_{p^*+1})\right| \\
&\geq (\Delta - 1)\left|W_{p^*+1}\right| + 1 - \left|i_E(W_{p^*+1}) \setminus i_T(W_{p^*+1})\right| - \left|i_T(W_{p^*+1})\right| \\
&\geq (\Delta - 1)\left|W_{p^*+1}\right| + 1 - \left|i_E(W_{p^*+1})\right|. \quad \square
\end{aligned}$$

We now use the bound on $\left|\vartheta_T(W_{p^*+1})\right|$ in Lemma 12 to prove a lower bound on $\Delta_{\mathrm{OPT}}(G)$.

Theorem 13. *Let T and $\mathcal{W}^{p^*}$ be the output of the* PR-MDMST *algorithm given a graph G and $\mu = 1$. Then the degree Δ of T is bounded by $\Delta_{\mathrm{OPT}}(G) + \lceil s(G) \rceil$.*

Proof. By Lemma 12, $\left|\vartheta_T(W_{p^*+1})\right| \geq (\Delta - 1)\left|W_{p^*+1}\right| + 1 - \left|i_E(W_{p^*+1})\right|$. By the definition of local density, $\frac{\left|i_E(W_{p^*+1})\right|}{\left|W_{p^*+1}\right|} \leq s(G)$. Lemma 4 then implies that $\Delta_{\mathrm{OPT}}(G) \geq \Delta - s(G) - 1 + \frac{1}{\left|W_{p^*+1}\right|}$. Since $\Delta_{\mathrm{OPT}}(G)$ is an integer, we conclude that $\Delta_{\mathrm{OPT}}(G) \geq \Delta - \lceil s(G) \rceil$. □

5.2. An additive factor of 2

Goemans [8] shows that the support of the natural linear program from the MDMST problem is sparse. He considers the following linear program:

$$\min\ c(x) = \sum_e c_e x_e$$

subject to:

$$\begin{aligned} x(i_E(S)) &\leq |S| - 1 && S \subset V \\ x(E) &= |V| - 1 && \\ x(\delta_E(v)) &\leq k && v \in V \\ x_e &\geq 0 && e \in E \end{aligned}$$

The above linear program is feasible for an integer k if and only if $\Delta_{\mathrm{OPT}}(G) \leq k$. Let x^* be an optimal extreme-point solution to the above linear program for a graph G and for $k = \Delta_{\mathrm{OPT}}(G)$. Let E^* denote the support of x^*, i.e. $E^* = \{e \in E : x^*_e > 0\}$. Theorem 5 in Goemans [8] can be paraphrased as:

Theorem 14 (*Goemans [8, Theorem 5]*). *The local density of the graph $G^* = (V, E^*)$ is less than* 2.

We first argue that $\Delta_{\mathrm{OPT}}(G) = \Delta_{\mathrm{OPT}}(G^*)$. Since $x^*_e = 0$ for all $e \notin E^*$, it follows that x^* is a feasible solution to the linear program for $G^* = (V, E^*)$, and so $\Delta_{\mathrm{OPT}}(G^*) \leq \Delta_{\mathrm{OPT}}(G)$. Since G^* is a subgraph of G, $\Delta_{\mathrm{OPT}}(G^*) \geq \Delta_{\mathrm{OPT}}(G)$. We conclude that $\Delta_{\mathrm{OPT}}(G) = \Delta_{\mathrm{OPT}}(G^*)$.

Since the linear program can be solved efficiently (see, e.g., Schrijver [21], Section 40.3), we can compute the graph G^* in polynomial time. Our $(\Delta_{\mathrm{OPT}}(G) + 2)$-algorithm computes the graph G^* and then runs the algorithm PR-MDMST with input G^* and $\mu = 1$. By Theorem 13, the tree T output by the algorithm has degree most $\Delta_{\mathrm{OPT}}(G) + 2$. Theorem 15 summarizes.

Theorem 15. *Given a graph G, there is a polynomial-time algorithm that computes an MST of G with degree at most $\Delta_{\mathrm{OPT}}(G)+2$.*

6. Minimum-degree minimum-cost base in a matroid

In this section, we consider a generalization of the MDMST problem.

6.1. Definitions

Recall that a *matroid* is defined to be a pair $M = (E, \mathcal{I})$, where E is a ground set of elements and $\mathcal{I}$ is a family of *independent sets* such that (i) $\emptyset \in \mathcal{I}$, (ii) $A \in \mathcal{I}, B \subseteq A$ imply that $B \in \mathcal{I}$, and (iii) $A, B \in \mathcal{I}, |A| > |B|$ imply that there exists $e \in A \setminus B$ with $B \cup \{e\} \in \mathcal{I}$. A maximum-cardinality independent set of M is called a *base* of M.

Let $c : E \to \mathbb{R}^+$ be a non-negative cost function on the ground set of M. A base T of M that minimizes the cost $c(T) = \sum_{e \in T} c(e)$ is referred to as a *minimum-cost base* (MCB). Let $x^T \in \{0, 1\}^{|E|}$ be the incidence vector of an MCB T. We say a vector $x \in [0, 1]^{|E|}$ is a fractional MCB if it is the convex combination of incidence vectors of MCB's.

Recall that a hypergraph $G = (V, E)$ consists of a set of nodes V and hyperedges $E \subseteq 2^V$, i.e. each hyperedge is a subset of vertices. We say G is a k-ary hypergraph if the cardinality of each edge in E is at most k. If $u \in e$, we say that node u is an *endpoint* of e and that e is *incident* on u. For a subset $F \subseteq E$ of edges and a node $u \in V$, let $\delta_F(u)$ be the set $\{e \in F : u \in E\}$ of edges incident on u. We define the *degree of u* in F to be $|\delta_F(u)|$. The *degree of F* is then defined as the maximum over $u \in V$ of the degree of u in F, i.e. $\max_u |\{e \in F : u \in e\}|$. Further, for a subset of vertices $U \subseteq V$, let $i_F(U)$ and $\vartheta_F(U)$ denote the sets $\{e \in F : e \subset U\}$ and $\{e \in F : e \cap U \neq \emptyset\}$, respectively.

We can now formally define the minimum-degree minimum-cost base (MDMCB) problem. The input to the problem is a k-ary hypergraph $G = (V, E)$ and a matroid M with E as its ground set.

MDMCB problem: Given a k-ary hypergraph $G = (V, E)$ and a weighted matroid $M = (E, \mathcal{I}, c)$, find an MCB of M that has minimum degree in G.

Let $\Delta_{\mathrm{OPT}}(G, M)$ denote the optimal degree for an instance of MDMCB.

6.2. The MDMCB algorithm

Our algorithm for MDMCB is a generalization of the PR-MDMST algorithm. To define it formally, we first define some analogous concepts. Given an MCB T of a matroid M, a pair (e, e') of elements in E is a *swap* in T if $e \in T$, $e' \notin T$, and $T \setminus \{e\} \cup \{e'\}$ is also an MCB of M. The *label* and *excess* of a node in V can be defined exactly as in Section 3. A labeling $l : V \to \mathbf{N}$ is extended to a labeling on the ground set E as follows: for $e \in E$, $l(e) = \max_{u \in e} l(u)$.

For a node $u \in V$, we let S_u^T denote the set of swaps (e, e') in T such that e is incident on u but e' in not incident on u. We call a swap (e, e') in S_u^T *useful* for u because it can be used to decrease the degree of u. We say a swap (e, e') in T is *feasible* for a labeling l if $l(e) \leq l(e') + 1$. As in Section 3, a labeling $l : V \to \mathbf{N}$ is defined to be *feasible* for an MCB T if for all nodes $u \in V$, for all swaps $(e, e') \in S_u^T$, $l(e) \leq l(e') + 1$. Given a labeling l and an MCB T, a swap $(e, e') \in S_u^T$ is called *permissible* for u if $l(u) \geq l(e') + 1$.

Our PR-MDMCB algorithm is defined exactly as the push–relabel algorithm described in Fig. 2, except that it takes as input a k-ary hypergraph $G = (V, E)$ and a weighted matroid M on the ground set E, in addition to the parameter μ. To prove that the PR-MDMCB algorithm works correctly, we first show that feasibility is maintained in any iteration of the algorithm.

6.3. Feasibility

Lemma 16. *The* PR-MDMCB *algorithm always maintains a feasible labeling.*

Proof. As in Lemma 1, we show this by induction on the number of iterations in a phase. In the beginning of any phase of the algorithm, all nodes and hence all edges e have label 0, which is a feasible labeling. In one step of the algorithm, we either update a label or execute a permissible swap (e, e'). Since we increment the label of a node only when it has no permissible swaps, feasibility is maintained in the first case.

To prove that feasibility is maintained when a permissible swap is executed, we make use of the *rank* function $r : 2^E \to \mathbf{N}$ of the matroid M. For a subset $E' \subseteq \mathbf{E}$, the rank $r(E')$ is defined to be $\max_{F \subseteq E' : F \in \mathcal{I}} |F|$. For any subsets $A, B \subseteq E$, the rank function r satisfies (i) $r(A) \leq |A|$, (ii) $r(A) \leq r(B)$ if $A \subseteq B$, and (iii) $r(A \cup B) + r(A \cap B) \leq r(A) + r(B)$. Note that A is a base of M if and only if $r(A) = r(M)$.

Let T be the current MCB after executing a sequence of swaps, so that the labeling l is feasible for T. Let (e, e') be the permissible swap executed in the current iteration, and let (f, f') be a swap in $T' = T \setminus \{e\} \cup \{e'\}$. If (f, f') is a swap in T, then by the inductive condition, $l(f) \leq l(f') + 1$.

Thus it remains to consider a pair (f, f') that is a swap in T' but not in T. There are three cases:

- $f' \in T$ *and hence not available for the swap in* T*:* In this case, $e = f'$. We observe that $T \setminus \{f\} \cup \{e'\} = (T \setminus \{e\} \cup \{e'\}) \setminus \{f\} \cup \{f'\} = T' \setminus \{f\} \cup \{f'\}$, which is an MCB of M since (f, f') is a swap in T'. Thus the pair (f, e') is a swap in T. By feasibility of (f, e') and permissibility of (e, e'), we conclude that $l(f) \leq l(e') + 1 = l(e) = l(f')$.
- $f \notin T$ *and hence not available for the swap in* T*:* In this case $e' = f$. We observe that $T \setminus \{e\} \cup \{f'\} = (T \setminus \{e\} \cup \{e'\}) \setminus \{f\} \cup \{f'\} = T' \setminus \{f\} \cup \{f'\}$, which is an MCB of M since (f, f') is a swap in T'. Thus the pair (e, f') is a swap in T. By permissibility of (e, e') and feasibility of (e, f'), we conclude that $l(f) = l(e') = l(e) - 1 \leq l(f')$.
- $f \in T$ *and* $f' \notin T$*:* Note that since (f, f') is not a swap in T but is a swap in T', $T \setminus \{f\} \cup \{f'\}$ is not a base of M. Thus $r(T \setminus \{f\} \cup \{f'\}) = r(M) - 1$.

  We first claim that $T \setminus \{e\} \cup \{f'\}$ is a base of M. Suppose not. Then $r(T \setminus \{e\} \cup \{f'\}) = r(M) - 1$. By submodularity of the rank function, $r(T \setminus \{e, f\} \cup \{f'\}) + r(T \cup \{f'\}) \leq r(T \setminus \{e\} \cup \{f'\}) + r(T \setminus \{f\} \cup \{f'\})$. Thus $r(T \setminus \{e, f\} \cup \{f'\}) = r(M) - 2$ so that $r(T \setminus \{e, f\} \cup \{e', f'\}) \leq r(M) - 1$, which contradicts that fact that (f, f') is a swap in T'.

  We now argue that $T \setminus \{f\} \cup \{e'\}$ is also a base of M. Suppose not. Then $r(T \setminus \{f\} \cup \{e'\}) = r(M) - 1$. Once again submodularity tells us that $r(T \setminus \{f\} \cup \{e', f'\}) + r(T \setminus \{f\}) \leq r(T \setminus \{f\} \cup \{e'\}) + r(T \setminus \{f\} \cup \{f'\}) \leq 2r(M) - 2$. Thus $r(T \setminus \{f\} \cup \{e', f'\}) \leq r(M) - 1$. This then implies that $r(M) = r(T' \setminus \{f\} \cup \{f'\}) = r(T \setminus \{e, f\} \cup \{e', f'\}) \leq r(M) - 1$, which again contradicts the fact that (f, f') is a swap in T'.

  Since $T, T \setminus \{e\} \cup \{f'\}$, and $T \setminus \{f\} \cup \{e'\}$ are all bases of M, and T is an MCB, it follows that $c(f') \geq c(e)$ and $c(e') \geq c(f)$. Since $c(e) = c(e')$ and $c(f) = c(f')$, we conclude that $c(e) = c(f)$. Thus (e, f') and (f, e') are both swaps in T. By feasibility of these swaps and permissibility of (e, e'), we conclude that $l(f) \leq l(e') + 1 = l(e) \leq l(f') + 1$.

We have shown that, in all cases, the swap (f, f') is feasible. Hence the induction holds. □

6.4. The witness

The concept of the witness generalizes to the MDMCB problem. We define a *witness* to be a pair (F, X) where $F \subseteq E$ is a subset of some MCB and $X \subseteq V$ has the property that in any MCB T of M containing F, each edge in $T \setminus F$ is incident on X in G. The following lemma is analogous to Lemma 2 and follows from essentially the same arguments.

Lemma 17. *Let $\mathcal{W} = (F, X)$ be a witness as defined above for a hypergraph G and a matroid M. Then any (fractional) MCB of M has maximum degree at least $\frac{r(M)-|F|}{|X|}$ in G.*

6.5. A constant-factor approximation

We now give generalizations of other results from Sections 3 and 4. Generalizations of Lemmas 3–5 and 8 are immediate. The observation that each swap is a parent of at most k swaps leads to Lemma 18, which is an analogue of Lemma 6. Corollary 19 is analogous to Corollary 7.

Lemma 18. *A label-p cascade contains at most k^{p-q} label-q swaps.*

Corollary 19. *A label-p cascade contains at most $(k^{p-q+1} - 1)$ involuntary losses to nodes with labels at least q.*

Proof. From Lemma 18, the total number of swaps with label at least q in a label-p cascade is at most $\frac{k^{p-q+1}-1}{k-1}$. The corollary follows from the fact that each swap causes at most $(k-1)$ involuntary losses. □

Lemma 20. *The number of involuntary losses to nodes in W_{p^*+1} is at most $k|W_{p^*+1}|(\frac{\mu}{\mu-k})$.*

The proof of Lemma 20 is analogous to the proof of Lemma 9.

We conclude with an approximation guarantee for the PR-MDMCB algorithm:

Theorem 21. *Given a k-ary hypergraph $G = (V, E)$, a weighted matroid $M = (E, \mathcal{I}, c)$, and a real-valued parameter $\mu > k$, the* PR-MDMCB *algorithm obtains in polynomial time an MCB of degree Δ, where $\Delta \leq k\mu\, \Delta_{\mathrm{OPT}}(G, M) + 1 + \frac{k\mu}{\mu-k}$.*

Proof. On input(G, M, μ), the PR-MDMCB algorithm terminates with a maximum independent subset T and a pair $\mathcal{W}^{p^*} = (F, X)$. We now compute the number $\left|\vartheta_T(W_{p^*+1})\right|$ of edges in T incident on W_{p^*+1}. Each node in W_{p^*+1} has degree at least $(\Delta - 1)$ after its last voluntary loss, and it may then suffer some involuntary losses. Using the bound from Lemma 20, the total loss in degree to W_{p^*+1} from involuntary losses is $\frac{k\mu}{\mu-k}\left|W_{p^*+1}\right|$. The sum of degrees in T of nodes in W_{p^*+1} is therefore at least $(\Delta - 1 - \frac{k\mu}{\mu-k})\left|W_{p^*+1}\right|$. Since each edge is incident on at most k nodes, the number $\left|\vartheta_T(W_{p^*+1})\right|$ of edges in T incident on W_{p^*+1} is at least $(\frac{1}{k})$ times this sum of degrees. Thus from the generalization of Lemma 4,

$$\Delta_{\mathrm{OPT}}(M) \geq \left(\Delta - 1 - \frac{k\mu}{\mu - k}\right)\frac{1}{k\mu}.$$

Rearranging, we get

$$\Delta \leq k\mu\, \Delta_{\mathrm{OPT}}(M) + 1 + \frac{k\mu}{\mu - k}. \quad \square$$

Remark: The bound above does not strictly generalize the bound in Theorem 10 since we do not have the structure necessary to bound the number of edges internal to W_{p^*+1} as we did in the proof of Theorem 10. Instead we use a cruder argument to lower bound the cardinality of $\vartheta_T(W_{p^*+1})$.

Choosing $\mu = k + \sqrt{\frac{k}{\Delta_{\mathrm{OPT}}(G,M)}}$, we get the following bound.

Corollary 22. *Given a k-ary hypergraph $G = (V, E)$ and a matroid M on E, there is a polynomial-time algorithm that outputs an MCB of degree Δ, where $\Delta \leq k^2\Delta_{\mathrm{OPT}}(G, M) + 2k^{\frac{3}{2}}\sqrt{\Delta_{\mathrm{OPT}}(G, M)} + k + 1$.*

6.6. A density-based bound

In this section, we generalize the result of Section 5.1. We first define a notion of density for a hypergraph. Given a hypergraph $G = (V, E)$ and a subset $U \subseteq V$, we define the density of U to be the ratio $\frac{\sum_{e\in\vartheta_E(U)}(|e\cap U|-1)}{|U|}$. In other words, each edge e incident on U contributes $|e \cap U| - 1$ to the numerator. The *local density* $s(G)$ of a hypergraph G is then defined to be the maximum density of any subset of V, i.e. $s(G) = \max_{U\subseteq V} \frac{\sum_{e\in\vartheta_E(U)}(|e\cap U|-1)}{|U|}$.

Let T be the tree and $\mathcal{W}^{p^*}$ the witness output by the PR-MDMCB algorithm on input (G, M) and $\mu = 1$. Let Δ be the degree of T. Since $\mu = 1$, the sets W_{p^*} and W_{p^*+1} must be equal, and hence, level p^* is empty when the algorithm terminates. The following lemma is an analogue of Lemma 12.

Lemma 23. *Let T be the MCB, l the labeling, and p^* the empty level when the algorithm* PR-MDMCB *terminates. Let $\vartheta_T(W_{p^*+1})$ be the set of edges in T that are incident on W_{p^*+1}. Then $\left|\vartheta_T(W_{p^*+1})\right| \geq (\Delta-1)\left|W_{p^*+1}\right| + 1 - \sum_{u\in\vartheta_E(W_{p^*+1})}(\left|e\cap W_{p^*+1}\right|-1)$.*

Proof. For a node $u \in W_{p^*+1}$, let $\delta_T(u) \subset T$ be the set of edges in T incident on u, and let $\delta_L(u) \subset E$ be the set of edges that node u loses involuntarily after its last voluntary loss. Since each node has degree at least $\Delta-1$ after its last voluntary loss, it follows that $|\delta_T(u)| \geq \Delta - 1 - |\delta_L(u) \setminus \delta_T(u)|$. Moreover, if v is the last node to be relabeled, $|\delta_T(v)| \geq \Delta$. Thus the sum of degrees of nodes in W_{p^*+1} in T is at least $(\Delta-1)\left|W_{p^*+1}\right| + 1 - \sum_{u\in W_{p^*+1}} |\delta_L(u) \setminus \delta_T(u)|$.

An edge e may be in $\delta_L(u)$ for at most $\left|e\cap W_{p^*+1}\right| - 1$ nodes $u \in e$. It follows that

$$\sum_{u\in W_{p^*+1}} |\delta_L(u)\setminus\delta_T(u)| \leq \sum_{e\in\cup_{u\in W_{p^*+1}}(\delta_L(u)\setminus\delta_T(u))} (\left|e\cap W_{p^*+1}\right|-1).$$

Moreover, $\cup_{u\in W_{p^*+1}}(\delta_L(u)\setminus\delta_T(u))$ is a subset of $\vartheta_E(W_{p^*+1})\setminus\vartheta_T(W_{p^*+1})$.

An edge e in $\vartheta_T(W_{p^*+1})$ contributes to $\delta_T(u)$ for each endpoint $u \in e\cap W_{p^*+1}$, and thus is counted $\left|e\cap W_{p^*+1}\right|$ times in the sum of degrees of nodes in W_{p^*+1} in T. Thus the number $\left|\vartheta_T(W_{p^*+1})\right|$ of edges in T incident on W_{p^*+1} is

$$\begin{aligned}
&\sum_{u\in W_{p^*+1}} |\delta_T(u)| - \sum_{e\in\vartheta_T(W_{p^*+1})} (\left|e\cap W_{p^*+1}\right|-1)\\
&\quad\geq (\Delta-1)\left|W_{p^*+1}\right| + 1 - \sum_{u\in W_{p^*+1}} |\delta_L(u)\setminus\delta_T(u)| - \sum_{e\in\vartheta_T(W_{p^*+1})} (\left|e\cap W_{p^*+1}\right|-1)\\
&\quad\geq (\Delta-1)\left|W_{p^*+1}\right| + 1 - \sum_{u\in\vartheta_E(W_{p^*+1})\setminus\vartheta_T(W_{p^*+1})} (\left|e\cap W_{p^*+1}\right|-1) - \sum_{e\in\vartheta_T(W_{p^*+1})} (\left|e\cap W_{p^*+1}\right|-1)\\
&\quad= (\Delta-1)\left|W_{p^*+1}\right| + 1 - \sum_{u\in\vartheta_E(W_{p^*+1})} (\left|e\cap W_{p^*+1}\right|-1). \quad \square
\end{aligned}$$

We now use the above bound on $\left|\vartheta_T(W_{p^*+1})\right|$ to prove a lower bound on $\Delta_{\mathrm{OPT}}(G,M)$.

Theorem 24. *Let T and $\mathcal{W}^{p^*}$ be the output of the* PR-MDMCB *algorithm given a hypergraph G, a matroid M, and $\mu = 1$. Then the degree Δ of T is at most $\Delta_{\mathrm{OPT}}(G,M) + \lceil s(G)\rceil$.*

Proof. From Lemma 23, $\left|\vartheta_T(W_{p^*+1})\right| \geq (\Delta-1)\left|W_{p^*+1}\right| + 1 - \sum_{u\in\vartheta_E(W_{p^*+1})}(\left|e\cap W_{p^*+1}\right|-1)$. By the definition of local density, the ratio $\frac{\sum_{u\in\vartheta_E(W_{p^*+1})}(\left|e\cap W_{p^*+1}\right|-1)}{\left|W_{p^*+1}\right|}$ is at most $s(G)$. An analogue of Lemma 4 then implies that $\Delta_{\mathrm{OPT}}(G,M) \geq \Delta - s(G) - 1 + \frac{1}{\left|W_{p^*+1}\right|}$. Since $\Delta_{\mathrm{OPT}}(G,M)$ is an integer, we conclude that $\Delta_{\mathrm{OPT}}(G,M) \geq \Delta - \lceil s(G)\rceil$. $\square$

We note that we are not aware of an analogue of Theorem 14 for the MDMCB problem. Such an analogue would imply an additive approximation via Theorem 24.

7. Bounded-degree minimum spanning trees

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

Könemann and Ravi [15] show that an MDMST for a certain cost function is also a BDMST with related guarantees on degree and with cost within a constant factor of the optimum. For the sake of completeness, we present this argument below.

7.1. Linear programs for BDMST

An integer linear program for the BDMST problem is given by

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

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 and can be written as a convex combination of spanning trees of G. For a fractional tree T_f with edge incidence vector $x \in \text{SP}_G$, let $\deg_{T_f}(v) = \sum_{e \in \delta(v)} x_e$. Now we can write the linear program relaxation of (1) as

$$\min\{c(T_f) : T_f \in \text{SP}_G, \forall v \in V \ \deg_{T_f}(v) \leq B\}. \tag{2}$$

The approach used by Könemann and Ravi [15] is to take the Lagrangean dual of (2), given by

$$\max_{\lambda \geq 0} \min_{T_f \in \text{SP}_G} \{c(T_f) + \sum_{v \in V} \lambda_v(\deg_{T_f}(v) - B)\}. \tag{3}$$

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 $T_f^B \in \mathcal{O}^B$ minimizes

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

The optimal multipliers λ^B and set $\mathcal{O}^B$ of trees can be computed in polynomial time. The optimal value $\text{OPT}_{LD(B)}$ of this dual program (3) is a lower bound on OPT_B and a tight lower bound on the optimal value of the LP relaxation (2).

Following the analysis of [15], we 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}. An optimal solution to the linear program (2) is a fractional tree $T_{f\ \text{OPT}}^B = \sum_{T \in \mathcal{O}^B} \alpha_T T$ that is also an MST under the cost function c^{λ^B}.

Let $B^* = B(1 + \beta)$ for some $\beta > 0$, and let $T_f^{B^*} = \sum_{T \in \mathcal{O}^{B^*}} \alpha_T^* T$ be an optimal solution to the LP (2) for degree bound B^*. Since λ^{B^*} is a feasible solution for the dual LP (3), it is clear that

$$c(T_f^{B^*}) + \sum_{v \in V} \lambda_v^{B^*}(\deg_{T_f^{B^*}}(v) - B) \ \leq \ \text{OPT}_{LD(B)}. \tag{4}$$

Recall that we are guaranteed by complementary slackness conditions that if $\lambda_v^{B^*} > 0$ then $\deg_{T_f^{B^*}}(v) = B^*$. Using this fact along with (4), we get

$$\begin{aligned} \text{OPT}_{LD(B)} &\geq \textstyle\sum_{v \in V} \lambda_v^{B^*}(\deg_{T_f^{B^*}}(v) - B) \\ &= \textstyle\sum_{v \in V} \lambda_v^{B^*}(B^* - B) \\ &= \beta \textstyle\sum_{v \in V} \lambda_v^{B^*} B. \end{aligned} \tag{5}$$

Now let T be any MST of G under the cost function $c^{\lambda^{B^*}}$. Then we arrive at the cost bound of T in [15] as follows:

$$\begin{aligned} c(T) \ &= \ c^{\lambda^{B^*}}(T) - \sum_{v \in V} \lambda_v^{B^*} \deg_T(v) \\ &\leq \ c^{\lambda^{B^*}}(T_f^{B^*}) \\ &\leq \ \text{OPT}_{LD(B)} + B \sum_{v \in V} \lambda_v^{B^*} && \text{by (4)} \\ &\leq \ \left(1 + \frac{1}{\beta}\right) \cdot \text{OPT}_{LD(B)} && \text{by (5)}, \end{aligned}$$

from which we conclude Theorem 25.

Theorem 25 *([15]). Let T be an MST of G under the cost function $c^{\lambda^{B^*}}$ with $B^* = B(1 + \beta)$. Then $c(T) \leq \left(1 + \frac{1}{\beta}\right) \text{OPT}_{LD(B)}$.*

From Theorem 25 and Theorem 11, we derive the following theorem.

Theorem 26. *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 $\left(1 + \frac{1}{\beta}\right) \text{OPT}_{LD(B)}$.*

For example, for $\beta = 1$, we produce a spanning tree of cost at most twice the optimum and of degree at most $4B + 4\sqrt{2B} + O(1)$.

Acknowledgement

We would like to thank the anonymous referees for several helpful comments.

References

[1] N. Bansal, R. Khandekar, V. Nagarajan, Additive guarantees for degree bounded directed network design, in: STOC, 2008, pp. 769–778.
[2] T.M. Chan, Euclidean bounded-degree spanning tree ratios, in: Proceedings of the nineteenth Annual Symposium on Computational Geometry, ACM Press, 2003, pp. 11–19.
[3] K. Chaudhuri, S. Rao, S. Riesenfeld, K. Talwar, What would Edmonds do? Augmenting paths and witnesses for bounded degree MSTs, in: Proceedings of APPROX/RANDOM, 2005, pp. 26–39.
[4] K. Chaudhuri, S. Rao, S Riesenfeld, K. Talwar, A push-relabel algorithm for approximating degree-bounded spanning trees, in: Proceedings of ICALP, 2006, pp. 191–201.
[5] J. Edmonds, Maximum matching and a polyhedron with 0–1 vertices, Journal of Research National Bureau of Standards 69B (1965) 125–130.
[6] T. Fischer, Optimizing the degree of minimum weight spanning trees. Technical Report 14853, Dept. of Computer Science, Cornell University, Ithaca, NY, 1993.
[7] M. Fürer, B. Raghavachari, Approximating the minimum-degree Steiner tree to within one of optimal, Journal of Algorithms 17 (3) (1994) 409–423.
[8] M. Goemans, Minimum bounded-degree spanning trees, in: Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science, 2006, pp. 273–282.
[9] A.V. Goldberg, A new max-flow algorithm. Technical Report MIT/LCS/TM-291, Massachusetts Institute of Technology, 1985.
[10] A.V. Goldberg, R.E. Tarjan, A new approach to the maximum flow problem, in: Proceedings of the eighteenth Annual ACM symposium on Theory of Computing, ACM Press, 1986, pp. 136–146.
[11] J.A. Hoogeveen, Analysis of Christofides' heuristic: Some paths are more difficult than cycles, Operation Research Letters 10 (1991) 291–295.
[12] R. Jothi, B. Raghavachari, Degree-bounded minimum spanning trees, in: Proc. 16th Canadian Conf. on Computational Geometry, CCCG, 2004, pp. 192–195.
[13] S. Khuller, B. Raghavachari, N. Young, Low-degree spanning trees of small weight, SIAM Journal on Computing 25 (2) (1996) 355–368.
[14] T. Király, L.C. Lau, M. Singh, Degree bounded matroids and submodular flows, in: IPCO, in: Lecture Notes in Computer Science, vol. 5035, Springer, 2008, pp. 259–272.
[15] J. Könemann, R. Ravi, A matter of degree: Improved approximation algorithms for degree-bounded minimum spanning trees, SIAM Journal on Computing 31 (6) (2002) 1783–1793.
[16] J. Könemann, R. Ravi, Primal-dual meets local search: Approximating MST's with nonuniform degree bounds, in: Proceedings of the Thirty-Fifth ACM Symposium on Theory of Computing, 2003, pp. 389–395.
[17] L.C. Lau, J. Naor, M.R. Salavatipour, M. Singh, Survivable network design with degree or order constraints, in: STOC, 2007, pp. 651–660.
[18] L.C. Lau, M. Singh, Additive approximation for bounded degree survivable network design, in: STOC, 2008, pp. 759–768.
[19] C.H. Papadimitriou, U. Vazirani, On two geometric problems related to the traveling salesman problem, Journal of Algorithms 5 (1984) 231–246.
[20] R. Ravi, M. Singh, Delegate and conquer: An LP-based approximation algorithm for minimum degree MSTs, in: Proceedings of ICALP, 2006, pp. 169–180.
[21] A. Schrijver, Combinatorial optimization: Polyhedra and efficiency, 1988.
[22] M. Singh, L.C. Lau, Approximating minimum bounded degree spanning trees to within one of optimal, in: STOC, 2007, pp. 661–670.